Uncovering Bias: Leveraging Deep Learning for Political News Classification

Darshan Chudiwal, dsc5636@psu.edu,
Yash Priyadarshi *yvp5218@psu.edu*

1. Problem Definition

The problem which has been picked for this task is the classification of news articles based on their political bias. The problem of classifying news articles according to their political bias is an impactful one which combines technical possibilities with societal relevance. It takes care of growing concern of media bias which influences the public's opinion, their trust in journalism and democratic processes, making it a critical area for transparency and awareness.

This problem leverages advanced NLP techniques like pretrained Word2Vec and deep learning models like LSTM and attention mechanisms to analyse and categorize the data effectively. The problem has real-world applications like building tools for balanced news aggregation, promoting media literacy and monitoring media bias trends. It also offers research opportunities at the intersection of computer science, political science and media studies. Nevertheless, challenges like presence of bias, data quality and ethical considerations must be handled to ensure that the model is fair, accurate and responsibly deployed. Overall, this problem statement is both well suited for modern AI solutions and meaningful, plus it offers significant societal benefits while advancing in the field of NLP.

2. Dataset Overview

The dataset used in this project consists of news articles labelled according to their political biasness. It provides a base for training and evaluating the bias classification model. The dataset has been picked from Kaggle under the name of Bias of US News Media Houses.

2.1. Column Description

The articles are have been sourced from various media outlets, ensuring diverse representation of political perspectives. Let us also have a discussion on the columns present in the dataset. The first column is an unnamed column which is used for index identification, next we have topic, which is the category of the article. Next we have source which is the media outlet or publisher. Next is bias which is the political bias label (e.g., left, centre, right). Next are URL, title, date of publication, author and content of the articles. Next column is content_original which has the pre-processed version of the content. Next is the source_url which is the base URL of the source, the bias_text which has the description of the bias label and lastly we have the ID column which stores a unique identifier for an article. The bias label in the dataset is essential for supervised learning which enables the model to learn the patterns and relations between the text and its corresponding bias category.

2.2. Dataset Split

The dataset has been split into training and validation sets using an 80-20 ratio, where 80% of the data is used for training the model and the remaining 20% is reserved for validation. The dataset contains 26,950 articles with 21272 articles allocated to the training set and 5318 articles to the validation set. This balanced split ensures that the model is exposed to a wide range of examples during training while reserving sufficient size for validation.

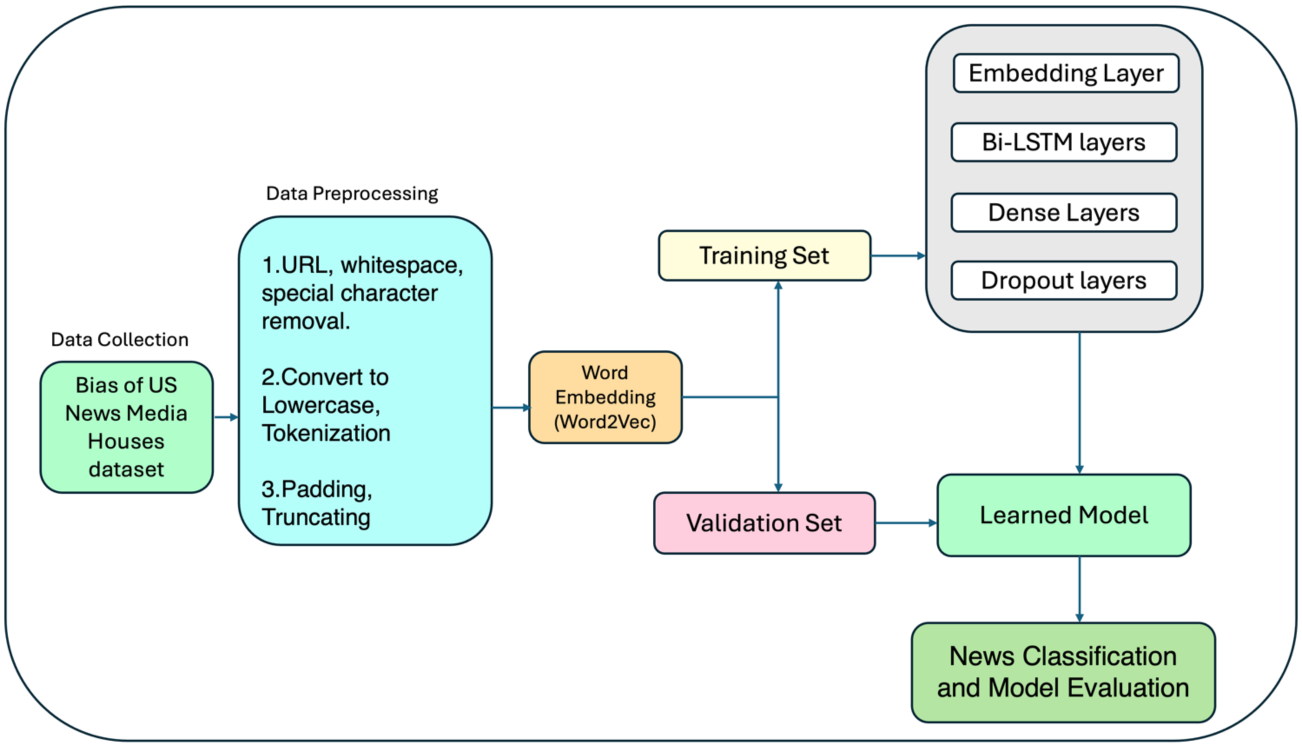
2.3. Data Preprocessing

Before we send the data to our deep learning model, we need to clean and standardize the data for which several pre-processing steps has been applied. For text cleaning, we have removed URLs, whitespaces and special characters. We have also converted all text to lowercase, tokenized them, added padding on shorter sequences and truncated longer ones.

2.4. Dataset Challenges

Author bias in annotations is a big challenge when labelling datasets, specifically for tasks like classification of political bias in news articles. This occurs because of annotator's personal opinions, beliefs or cultural backgrounds and can influence how they interpret the articles. An article that person A thinks is neutral can be interpreted as biased by person B depending on their political views.

This subjectivity leads to inconsistent labels which affects the quality of the dataset and the performance of the model trained on it. When author bias pops into annotation, it can cause problems like inconsistent labels, skewed data and poor model performance.


Data Collection
Bias of US News Media Houses dataset
Data Preprocessing
1.URL, whitespace, special character removal.
2.Convert to Lowercase, Tokenization
3.Padding, Truncating
Word Embedding (Word2Vec)
Training Set
Validation Set
Embedding Layer
Bi-LSTM layers
Dense Layers
Dropout layers
Learned Model
News Classification and Model Evaluation
Figure 1: Architectural overview of our implementation.

Different annotators could label the same article differently which makes the dataset unreliable. If annotators lean towards a particular political perspective, then the dataset becomes imbalanced and favours that specific perspective. Lastly, a model trained on biased and inconsistent data might not generalize well which leads to inaccurate predictions.

Let us also discuss how these issues can be mitigated. Annotators can be provided with detailed instructions and instances to label the articles objectively. Combined judgement of several people over a common article can be jotted down to reduce individual biases. By having annotators from diverse backgrounds and political views, we can get a balanced data without personal biases. By addressing the author bias, the dataset becomes more reliable, model learns better and makes better predictions.

3. Approach

We have implemented a deep learning model to classify the bias of US news media articles using their titles and content, which uses pre-trained word embeddings and RNN. For data preparation, we load the data from the excel file in which the dataset is present and clean it by removing the URLs, special characters and excessive whitespaces, which ensures that our final input data is standardized.

3.1. Word Embeddings, Tokenization and Vocabulary

With respect to word embedding and tokenization, we are using the Word2Vec embedding technique to convert the words into numerical vectors and capture the semantic relationships. The text is then tokenized by splitting based on the whitespaces and padded or truncated to a fixed length for proper processing.

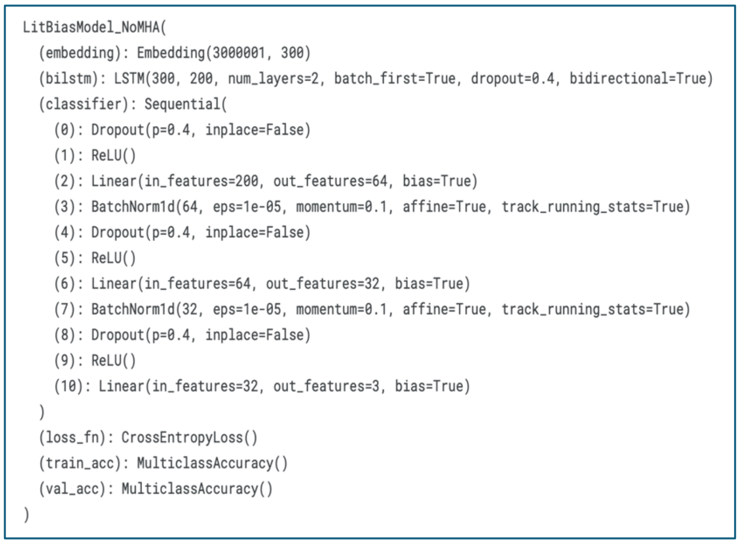
We are using a pre-trained Word2Vec model from "*/kaggle/input/googlenewsvectorsnegative300/GoogleNews-vectors-negative300.bin*". We are building the embedding matrix by mapping the words to indices and reserving the index 0 for unknown tokens. The embedding matrix which is initialized with zeros is then being filled with Word2Vec model i.e. `torch.tensor(w2v_model[word])`.

3.2. Model Architecture

Our model uses a frozen Word2Vec embedding layer to preserve the pre-trained word representations. A shared bi-directional LSTM layer is used to process both, the title and body text and captures sequential dependencies in both directions. The hidden states are mean pooled and averaged for title and body, and then passed on through a feed-forward network for classification.

Let us also take a look into some of the key components of the model. Using the `LitBiasModel_NoMHA` class as shown in Fig. which implements PyTorch's Lightning module we have incorporated several layers –

- **Embedding layer** – The model uses a pre-trained Word2vec embeddings (GoogleNews-vectors-negative300.bin), which loads the embedding and stores them in the embedding matrix. Each word is mapped to 300 dimensional vector space.
- **Bidirectional LSTM** – This enables the model to process text in both forward and backward directions, which captures context from past as well as future words in the sentence, while improving its ability to understand long term dependencies in the text.
- **Mean Pooling and aggregation** – The hidden states from all LSTM time steps are not discarded after processing and instead of just using the final LSTM output, a mean pooling operation is applied which takes the average of all the hidden states across the sequence and provides a compressed representation of the entire review.
- **Classifier** – A feedforward network with layers including dropout, ReLU activations, batch normalization and linear transformations has been used for classification, with cross entropy as the loss function.



```
LitBiasModel_NoMHA(
  (embedding): Embedding(3000001, 300)
  (bilstm): LSTM(300, 200, num_layers=2, batch_first=True, dropout=0.4, bidirectional=True)
  (classifier): Sequential(
    (0): Dropout(p=0.4, inplace=False)
    (1): ReLU()
    (2): Linear(in_features=200, out_features=64, bias=True)
    (3): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): Dropout(p=0.4, inplace=False)
    (5): ReLU()
    (6): Linear(in_features=64, out_features=32, bias=True)
    (7): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): Dropout(p=0.4, inplace=False)
    (9): ReLU()
    (10): Linear(in_features=32, out_features=3, bias=True)
  )
  (loss_fn): CrossEntropyLoss()
  (train_acc): MulticlassAccuracy()
  (val_acc): MulticlassAccuracy()
)
```

Figure 2: Bi-directional LSTM model

3.3. Training and Evaluation

We have trained our model using PyTorch Lightning (a high level library for Deep learning), which ensures scalability and logging. The model is evaluated on a validation set over metrics like confusion matrix, classification reports, ROC curves, precision recall curves, and calibration curves for a thorough performance analysis.

3.3 Optimization Details

For model optimization, we employed the AdamW optimizer with an initial learning rate of 1e-3. We applied a dropout rate of 0.4 throughout the network to help mitigate overfitting. Although the training was set to run for up to 100 epochs, an early stopping mechanism monitored via validation loss with a patience of 5 epochs, terminated training at around 20 epochs, as the validation performance plateaued while training accuracy continued to improve. These experiments were executed on Kaggle using the two NVIDIA T4 GPUs provided by the platform. Note that no fixed random seed was set for these runs, which may introduce some variability in results across different runs.

4. Experiments and Analysis

We ran some ablation experiments examining our proposed model and it's underlying components.

4.1 Early Stopping Experiment

For the early stopping experiment, we set up PyTorch Lightning's EarlyStopping callback to monitor the validation loss (with a patience of 5 epochs) and a ModelCheckpoint callback to monitor "Epoch_Val_Acc." Although, we allowed training for up to 100 epochs, early stopping kicked in around epoch 20 once the validation loss plateaued. Although our training accuracy continued to improve, the validation accuracy plateaued or only fluctuated slightly. This plateau in validation performance triggered the early stopping mechanism. There's a noticeable gap between the training and validation accuracies. While training loss kept decreasing, the validation loss did not show significant improvement beyond 20 epochs—indicating that further training might lead to overfitting. This experiment provided clear evidence for the optimal stopping point and helped ensure that the model's generalization was preserved. Furthermore, the early stopping outcome suggests that extending training further under the current hyperparameters might not yield genuine improvements on unseen data. Thus, we next vary the dropout rates.

4.2 Dropout Study

We have evaluated the effects of different dropout rates [0.2, 0.4, 0.5, and 0.6] over 20 epochs –

- As observed in Figure 3 and Figure 4, **at 0.2 dropout**, both training and validation accuracies steadily improved (with training accuracy moving from ~42% to ~90% and validation accuracy from

~34% to ~76%), accompanied by a corresponding decrease in loss.

- At 0.4 dropout, the validation accuracy still reached around 76%, which is comparable to the 0.2 setting; however, the training accuracy was a bit lower, as expected with a stronger regularizer.
- At 0.5 and 0.6 dropout, we observed overall lower training accuracy and more fluctuations in validation accuracy. This suggests that excessive dropout might be causing underfitting or slowing convergence.

These results show that moderate dropout (between 0.2 and 0.4) provides a good balance between reducing overfitting and allowing the network to learn effectively. Although higher dropout rates degrade the training signal, the overall impact of changing dropout within this range wasn't dramatic. Thus, we pick a moderate dropout of 0.4 for all our experiments.

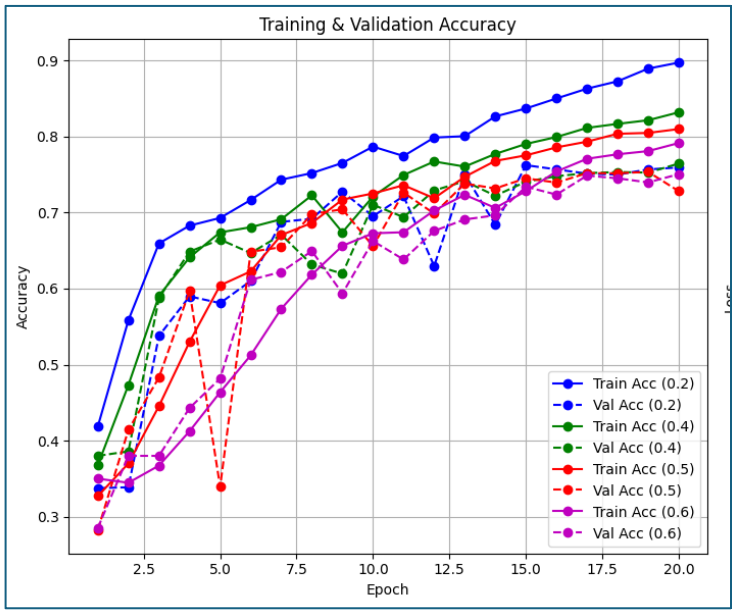


Figure 3: Classification Accuracy for various dropout values

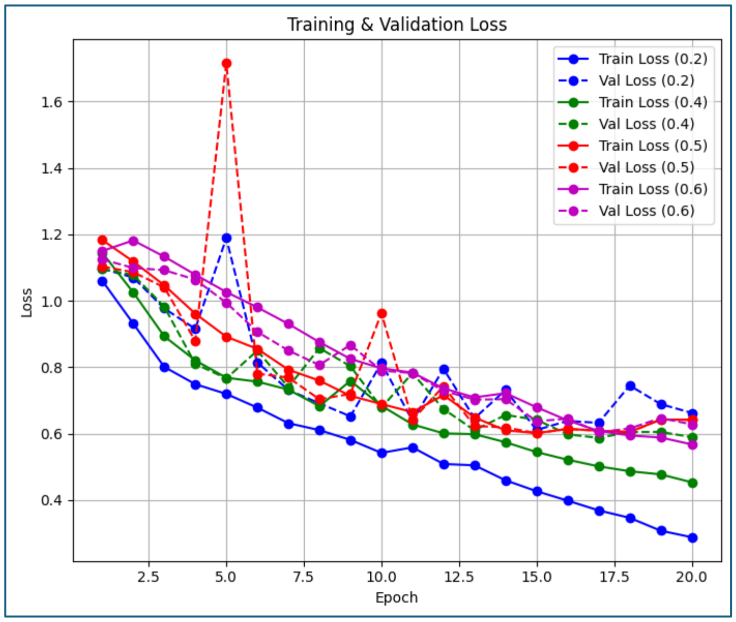


Figure 4: Loss for various dropout values

4.3 Number of BiLSTM Layers

We compared the performance of models using one versus two layers of bidirectional LSTM (with a dropout rate of 0.4) to understand the impact of network depth of the bidirectional LSTM layers:

The single-layer model's validation accuracy peaked around 70–73%, whereas the two-layer model reached around 75–76% as depicted in Figure 4. This suggests

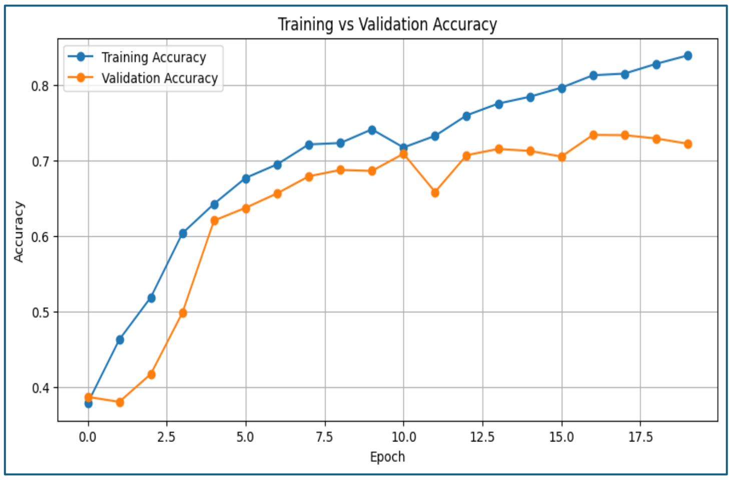


Figure 5: Classification accuracy for model with single bidirectional lstm layer.

that the additional layer provides extra capacity to capture nuanced features. With only one layer, the model has fewer parameters and is less capable of capturing intricate details, which likely explains its slightly lower accuracy. Despite having fewer parameters, the single-layer model follows a similar training pattern—a rapid increase in accuracy, a slight dip, then partial recovery. This behaviour may be influenced by the 0.4 dropout rate, which can slow convergence. A single-layer BiLSTM might be acceptable if we were to require a lighter model or faster inference. However, for achieving the highest possible validation accuracy, we choose to stick to the two-layer model.

5. Results

Let us now have a look at the results of how our model predicted. Our goal was to categorize the news articles based on their political bias into three classes which are class 0, class 1 and class 2, representing left-leaned, neutral and right leaned. The results include overall metrics, the classification report and insights from various visualizations which when integrated provides a comprehensive view of our model's performance.

With respect to the overall metrics, we achieved a **validation accuracy** of 0.7484 (74.84%), a validation loss of 0.6209, a **training accuracy** of 0.8135 (81.35%) and a training loss of 0.4914. Now we can see that the training accuracy is higher at 81.35% which indicates that the model performs better on training data than on unseen data. This suggests that there is overfitting, and although our model may learn the training data well but struggles to generalize on new articles.

5.1 Classification Report

The classification report provides precision, recall, F1-score and support for each class.

Class	Precision	Recall	F1-Score	Support
0	0.8325	0.6702	0.7426	1795
1	0.8586	0.6225	0.7217	1502
2	0.6616	0.9114	0.7667	2021
Accuracy			0.7484	5318
Macro Avg	0.7842	0.7347	0.7437	5318
Weighted Avg	0.7749	0.7484	0.7459	5318

Table 1: Classification Report

Precision indicates how well the model predicts a certain class element and recall tells us the how well the model identifies actual class elements of a particular class. From the classification report above, we can say the **precision**, **recall**, **F1-score** and support values for all three classes. For Class 0, we have an F1 score of 0.7426 indicating a balanced measure of precision and recall and having a moderate performance. For class 1, we can see a slightly lower F1-score than class 0, reflecting the lower recall, and Class 2 has the highest F1-score driven by the high recall. The **macro average** treats all classes equally, while the **weighted average** accounts for class imbalance. From the above report, we can see that class 2 has the most samples followed by Class 0 and then Class 1.

5.2. Confusion Matrix

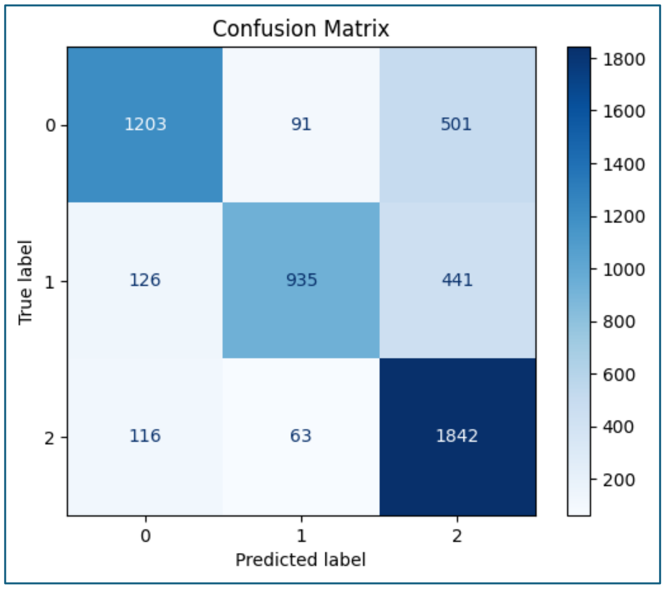


Figure 6: Confusion Matrix for the model

From the confusion matrix, we can see the number of correct and incorrect predictions. Class 0 has 1203 (out of 1795) correct predictions, but has 501 misclassified predictions as Class 2 which gives rise to its lower recall of 0.6702. For Class 1, we see a total of 935 correct predictions (out of 1502), with 441 misclassified as class 2, explaining its low recall of 0.6225. For Class 2, we see a 1842 correct classifications (out of 2021) with few misclassifications (116 as Class 0 and 63 as Class 1), which is aligning with its high recall of 0.9114 but lower precision due to the false positives from other classes. Lastly, the matrix highlights a tendency to misclassify Classes 0 and 1 as Class 2, likely due to Class 2's larger sample size (2,021 vs. 1,795 and 1,502).

5.3. ROC Curves (One-vs-Rest)

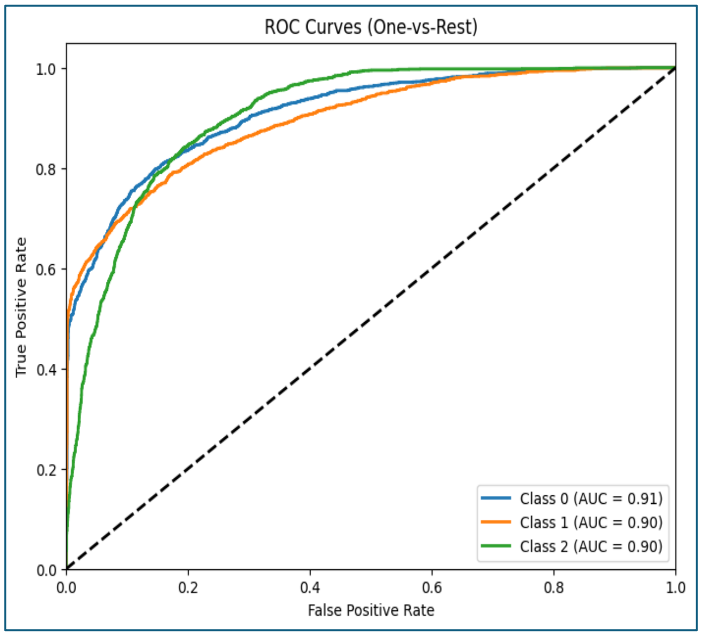


Figure 7: Receiver Operating Characteristic curve for our model.

A high value of area under curve (AUC), indicates better model performance and suggests the model's ability to differentiate between classes. From the above plot we can see that all three classes have high AUC scores (close to 1), indicating strong discriminatory power (can classify better) for each class when treated as a positive class against the others. Also, Class 0 slightly outperforms others, which reflects its higher precision.

5.4. Precision-Recall-Curves (One-vs-Rest)

From the PR curve pasted along-size, we can see the average precision (AP) scores for all three classes. The scores are high which shows that the model maintains a good precision across various recall levels. Class 0's higher AP aligns with its higher precision, while Class 2's lower precision is offset due to its high recall.

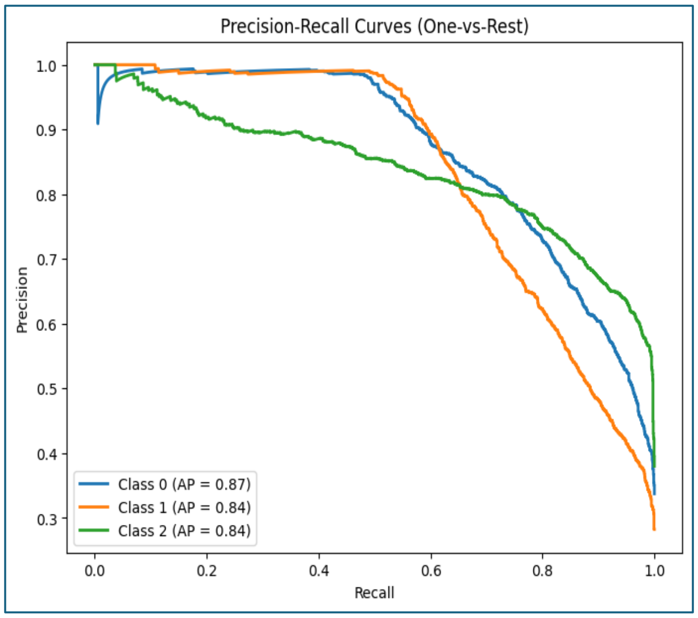


Figure 8: Precision-recall (PR) curve for our model WRT difference classes.

5.5. Training vs Validation Accuracy

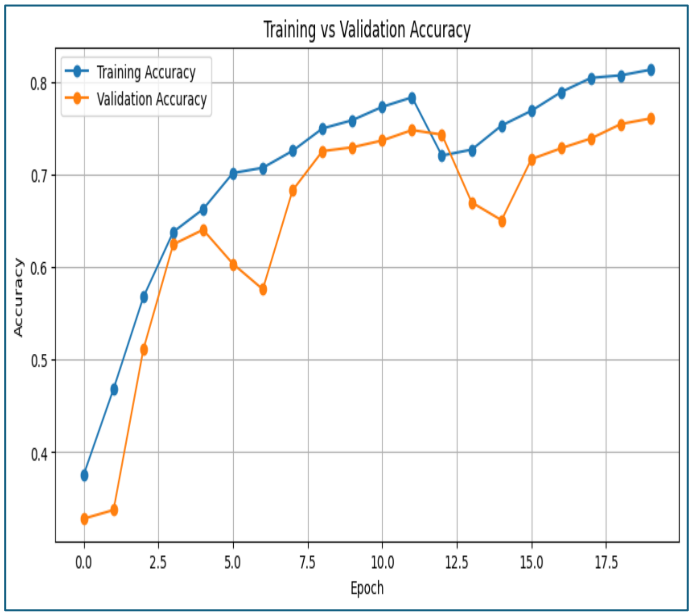


Figure 9: Training vs Validation accuracy for the model.

The training accuracy rises steadily to 0.8 by epoch 18, while the validation accuracy peaks at 0.75 around epoch 10, then dips and stabilizes between 0.7 – 0.75 at the end. The gap after epoch highlights the overfitting of the model.

5.6. Training vs Validation Loss

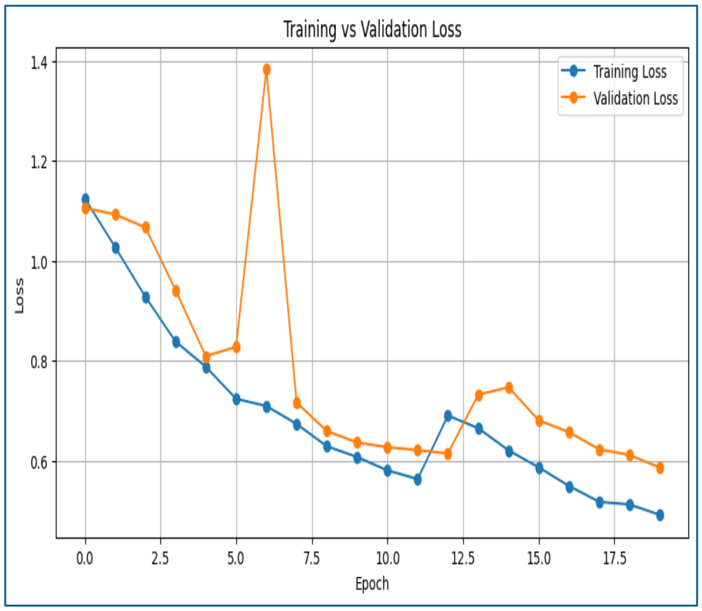


Figure 10: Training vs Validation Loss for the model.

From the above plot, we can see that the training loss decreases smoothly to 0.4 by epoch 18. Also, the validation loss drops initially but spikes to 1.4 at epoch 5, and then stabilizes at 0.6 towards the end. The persistent gap from the training loss reinforces overfitting.

6. Lessons Learned

We explore a complex issue and explore the potential of using NLP as a tool of political analysis.

6.1. Data Quality and Annotation Changes

One of the biggest challenges was the significant impact of data quality and labelling political bias. This challenge in particular highlighted the need for a larger human
We learned that even subtle inconsistencies can influence model training and evaluation, leading to overfitting or skewed predictions. Future work should consider strategies like ensemble labeling or incorporating annotators with diverse backgrounds to achieve more balanced and reliable labels.

6.2. Model Architecture

The experiments revealed a trade-off between model complexity and generalization. While the two-layer BiLSTM model improved validation accuracy compared to a single-layer approach, it also exposed the model's tendency to overfit—evidenced by the widening gap between training and validation metrics. This observation highlights the critical need for effective regularization techniques and more robust validation strategies. Our early stopping experiment was particularly instructive, demonstrating that monitoring validation loss can prevent unnecessary training and potential overfitting, though it also prompts us to explore additional techniques.

6.3. Role of Hyperparameter Tuning

Our dropout study was instrumental in understanding how different regularization strengths affect learning. We observed that moderate dropout rates (around 0.2 to 0.4) provided a balance between maintaining a strong training signal and reducing overfitting, while too aggressive dropout led to underfitting.

6.4. Experimentation and Evaluation Metrics

A comprehensive evaluation using metrics like precision, recall, F1-scores, and ROC/PR curves proved essential for a nuanced understanding of the model's performance. The varying performance across classes show the need for class-specific adjustments and the potential benefits of using cost-sensitive learning strategies.

7. References

1) G. Xu, Y. Meng, X. Qiu, Z. Yu, X. Wu. Sentiment Analysis of Comment Texts Based on BiLSTM, IEEE Access, 2019.
2) U. B. Mahadevaswamy, Swathi P. Sentiment Analysis using Bidirectional LSTM Network. Elsevier, 2023.
3) Z. Cui, R. Ke, Z. Pu, Y. Wang. Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction, arXiv.org, 2018.