Reading Between the Lines: Measuring Hyper-Partisanship in the News with Deep Learning

Stanford CS224N Custom Project. Mentor: Swastika Dutta

Katherine Crandell
Department of Computer Science
Stanford University
kwc@stanford.edu

Andy Huynh
Department of Computer Science
Stanford University
avhuynh@stanford.edu

David Wang
Department of Computer Science
Stanford University
davidw23@stanford.edu

Abstract

Hyper-partisanship in the news and on social media has contributed to the general decline of democracy in the United States over the past decade [1]. Building on the results of the SemEval 2019 hyper-partisanship classification task [2], we built an ensemble model consisting of 3 transformer architectures (XLM-RoBERTa, XLNet, BigBird) to classify news articles as Right, Center-Right, Center, Center-Left, and Left. Individually, the BigBird model outperformed the other two with an accuracy of 0.665. Our final ensemble model was able to achieve an accuracy of 0.608. We then used a fine-tuned model to classify a selection of articles from the Stanford Daily and the Stanford Review.

1 Background

1.1 Hyper-partisanship in the News

A hyper-partisan news article describes a sharply polarized situation where the contents of the article are in fierce disagreement with the opposing side, often including misinformation and inflammatory language. With the development of social media, these articles can be shared broadly, deepening political divides and spreading harmful content.[1] Classification algorithms, like the models we utilized, can help detect and flag articles for hyper-partisanship to prevent its spread, creating a better environment in online spaces. In our increasingly polarized political climate, providing an objective standard which consumers can ground themselves with could be an extremely valuable tool.

1.2 SemEval 2019 Task 4

Our project was inspired by the SemEval 2019 research paper. In 2019 as a part of the annual International Workshop on Semantic Evaluation (SemEval), researchers around the world competed to see who could achieve the best scores on 11 different NLP tasks.[2] For task 4, researchers were challenged to develop models which could identify if a news article was likely to be "hyper-partisan" or not. The motivation behind this task is that an effective NLP model could be used to filter out partisan news online which is less interested in objectivity and more interested in appealing to partisanship for political expediency.

The submissions for this competition were a very diverse range of models, but across the board most models found it helpful to learn word embeddings using Glove, Word2Vec, or ELMo. Beyond this commonality, there was not much shared between teams. Several teams used traditional neural networks, others used CNNs and LSTMs, and a few teams also used transformers with BERT pretraining. Since researchers were tasked with a simple classification problem, they used common

classification metrics like accuracy, precision, recall, and F-1 score which are various ratios of true and false positives as well as true and false negatives.

There were 2 datasets available: by article and by publisher. The former consists of 645 articles and labeled by expert, while the latter consists of 600,000 and labeled by publisher. For the by article training dataset, although it consisted of far fewer articles, the classifiers trained on this dataset achieved far better results (20% higher accuracy) than those that were trained on the by publisher dataset. The authors were surprised by this results, because they had hoped that the power of big data might overcome the limited labeling scheme they used for the larger dataset.

Overall, hyper-partisan news could be detected with over 80% accuracy. The winning team learned sentence based ELMo embeddings which were fed through a traditional CNN. Another high performing team simply used a random forest classifier, which is an ensemble of an even more traditional classifier, the decision tree. Interestingly, after receiving the submissions the authors then created a couple of models which consisted of ensembles of the top performing models. The resulting model outperformed any single model on its own.

Since BERT was relatively new, very few teams used transformers in the SemEval 2019 competition. Those that did, did not perform particularly well in comparison with the rest of the competition. Because of this, we hoped to improve upon their results by using newer state of the art pretrained transformer models and ensembling the resulting models together.

1.3 Pretrained Transformer Models

A transformer model is a neural network that learns from context from sequential data utilizing self attention while masking tokens to enable next sentence prediction. Today, they are by far the most popular paradigm in NLP, and have shown to perform well on a variety of tasks when finetuned from a pretrained model. As such, many different pretrained transformers architectures exist and each has their strengths and weaknesses. The three popular architectures that we decided to finetune and ensemble are XLM-RoBERTa, XLNet, and BigBird.

XLM-RoBERTa was pretrained on a vast corpus of text data (2.5TB of filtered data that was scraped from the web) in a self-supervised manner. Compared to the base BERT model, RoBERTa pre-trains over a longer period of time and on longer sentences while removing the next sequence classification task and dynamically changing the masking pattern[3]. In this way, RoBERTa greatly improves the performance that BERT achieved, without making changes to the actual model architecture.

While BERT learns structure by masking tokens and predicting masked tokens independent of each other, XLNet utilizes permutation based language modeling to shuffle the sequence of tokens and uses auto-regressive language modeling to estimate the probability distribution over all permutations in both forward and backward directions. Permutation based language modeling is shown in Figure 1. By training on different permutations of input sentences, the model is exposed to a wider variety of input sequences and can perform auto-regressive predictions better. This allows the model to capture bi-directional information without masking tokens, preventing the corruption of input data. This method prevents pretrain-finetune discrepancy and outperforms BERT in 20 tasks[5].

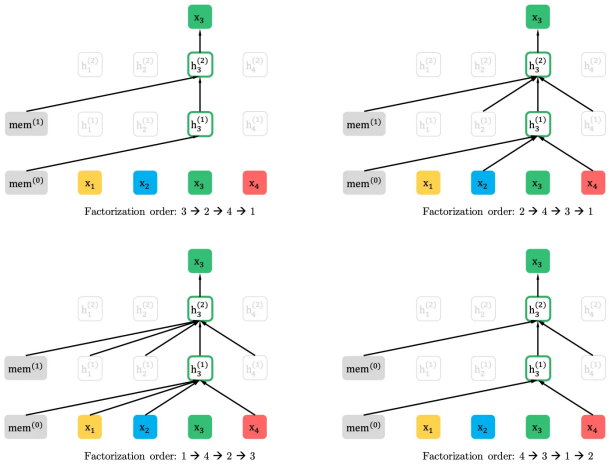
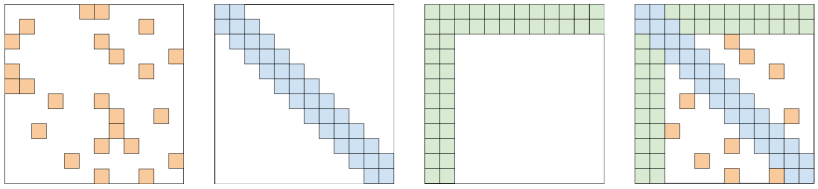


Figure 1: Figure illustrating the permutation language modeling objective for predicting x_3 given the same input sequence but different permutation orders. [4]

Compared to BERT, BigBird uses sparse attention which applies attention to each token as opposed to each sequence [6]. Figure 2 demonstrates the differences in attention mechanisms that are combined into the BigBird



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 2: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with r = 2, (b) sliding window attention with w = 3 (c) global attention with g = 2. (d) the combined BIGBIRD model. [6]

model. Using the BigBird model, queries attend to random keys (as shown in figure 2a), to tokens to the left and right of its location (shown in figure 2b), and to global tokens (as shown in figure 2c). BigBird can also process longer sequences and utilized longer masked language model pre-training than BERT.

In summary, all of our chosen models improve on the base BERT performance, and ensembling them should in theory give us even better performance. Specifically, RoBERTa varies the masking location, XLNet utilizes permutation based modeling to avoid pretain-finetune discrepancy, and BigBird combines multiple forms of attention to better learn dependencies between words.

2 Dataset

All of our models were trained on the HuggingFace dataset of news articles labeled for hyper-partisan content which is a publically available version of the dataset used at SemEval [7]. There are two sub-sets of data contained in this dataset: a small set of just 645 hand labeled articles and a much larger set of 600,000 articles labeled based on their publisher.

The features of the by publisher dataset of 600,000 articles are:

- text: a string feature.
- title: a string feature.
- hyperpartisan: a bool feature.
- url: a string feature.
- published_at: a string feature describing the date that an article was published.
- bias: a classification label, with possible values including right (0), right-center (1), least (2), left-center (3), left (4).

Articles with a bias of 0 or 4 had a were labeled as hyper-partisan, while those with bias of 1, 2, or 3 were not labeled hyper-partisan. For example, articles posted by *dailywire.com* were labeled as hyper-partisan with a bias of 0, indicating right leaning content, whereas articles posted by *forwardprogressive.com* were labeled as hyper-partisan with a bias of 4, indicating left leaning content. The dataset is balanced so that half of the articles are hyper-partisan and the other half are not. Of the hyper-partisan articles, or non-hyperpartisan articles, again half of them contain views that lean right while the other half lean left on the political spectrum. In our models, we did not use the title, url, or published_at features.

The other subset of 645 articles that were labeled by hand had the following features:

- text: a string feature.
- title: a string feature.
- hyperpartisan: a bool feature.
- url: a string feature.
- published_at: a string feature.

These articles were read and manually labeled by 3 annotators. For this sub-set, articles are not labeled only labeled as hyper-partisan or not and no bias score is given. This makes the labels on this subset more reliable than the generated labels for the 600,000 articles, since the bypublisher dataset assumes that a given platform only produces one type of article. While this makes it possible to label a much larger collection of articles automatically, it does make for less accurate labels.

Because of the differences between the bypublisher and byarticle dataset, we decided to use the bypublisher dataset for training and validation, and the byarticle dataset as our test dataset. This allowed us to do our fine-tune training on as large of a corpus as possible, but test our models and evaluate them realistically with the more accurate hand labelled dataset. To do so, we considered scores of 0 or 4 to be hyper-partisan and scores 1-3 to not be hyper-partisanship.

2.1 Data Preprocessing

The text of articles in this dataset contained HTML tags and formatting. HTML tags don't contribute to our understanding of language and would not be helpful for our classification task, so we had to pre-process the text data to clean it. We utilized the trifilatura python package to clean the data. [8] When we created our baseline, we had not decided to remove HTML tags, so the RoBERTa model was trained on article text that contained HTML tags, while the other two models were not.

3 Methods

Our main approach for this project involved creating an ensemble model of state-of-the-art pretrained transformer models for our classification task. To do this, we first obtained each of these models from the HuggingFace Transformers library, and then finetuned the models for sequence classification on the SemEval dataset. After fine tuning and achieving decent performance, we combined all of our models in to an ensemble model and evaluated the performance of the ensemble as a whole. In the following subsections we will discuss both the individual fine-tuned models and the ensemble model in more detail.

3.1 Finetuning

Using the HuggingFace Transformers library, we fine tuned three transformer models: XLM-Roberta, XLNet, and BigBird. We trained each model on the SemEval labeled by publisher training set, evaluated on the SemEval labeled by publisher validation set, and tested on the SemEval labeled by article dataset. Because this was an incredibly large corpus of text, we thought the models would learn a lot about the features of hyper-partisan text. The specific experimental details of the training will be elaborated on in a later section. We trained our model on the bias label of each article.

In order to prepare the models for predicting bias, we needed to add a linear neural network classifier on top of the sequence outputs the base models produce. To do this, we used hugging face's AutoModelForSequenceClassification, which automatically generates the required neural network with the desired dimensions going from sequence output size to number of labels. The AutoModel uses cross-entropy loss on the 5 labels, and allow gradients to backpropagate all the way through the entire network when optimizing.

We used a learning rate of 0.00002. We trained the RoBERTa model for 5 epochs, XLNet model for 3 epochs, and BigBird model for 4 epochs. We evaluated model performance on the test set after each epoch, and saved a checkpoint so that we could use the highest performing checkpoint. We achieved a training loss of 0.03 for the RoBERTa model, 0.07 for XLNet, and 0.02 for BigBird. Training on the labeled by publisher subset took over 24 hours for all models.

3.2 Creating an ensemble model

After finetuning our models individually, we combined the best checkpoint from three models using a regular dense neural network with one hidden layer. Our highest performing BigBird model was the epoch 4 model, whereas for XLNet and RoBERTa it was the epoch 1 model. As shown in Fig. 3, we do this by concatenating the sequence output from each of the individual models which were all of size 768 for a total input layer size of 2304. We pass it through two hidden layers, with size 1200 and 600, then use a final dense layer from 600 to 5 to calculate the logits. When training the

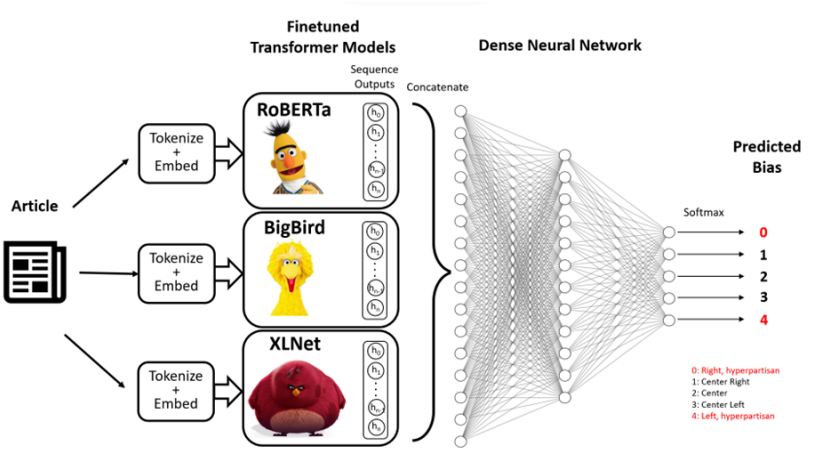


Figure 3: A schematic of our ensemble model architecture showing the pipeline from raw text to predicted bias.

ensemble model, we chose to keep the parameters in the transformer models frozen, only fine tuning the classifier network. We did this in part due to computational limitations, and in part because we wanted to preserve the performance of the underlying models such that we had a true ensemble model as opposed to a new model architecture. We used the following hyper-parameters during training:

Table 1: Hyperparameters

Hyperparameter	Value
Batch-size	256
Learning Rate	0.003
Betas	0.9 and 0.999
Dropout rate	0.01
Weight Decay	0
Learning Rate Decay	0

We tried several variations of these parameters and hyperparameters and this selection had the best possible performance.

3.3 Evaluation Metrics

To quantitatively evaluate each of our models, we used the same common classification metrics, as used in Kiesel *et al.* 2019. [2]. These include:

- Accuracy

$$\text{Accuracy} = \frac{TN + TP}{TP + FP + TN + FN}$$

- Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall

$$\text{Recall} = \frac{TP}{TP + FN}$$

- F-1 Score

$$\text{F-1} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

We computed these metrics for all models on both the test, labeled by article, data and the training data. Since the training data is a multi-label classification problem as opposed to a binary classification problem, the metrics were computed all against one for each label and then averaged.

4 Results

4.1 Models' Performance

For our baseline, we wanted to see what accuracy we could achieve without any hyperparameter tuning and without ensembling our model. We trained our baseline on the article text before stripping it of the HTML tags. The results are tabulated below:

Baseline Results				
	Accuracy	Precision	Recall	F-1
Train	0.993	0.993	0.993	0.993
Test	0.606	0.700	0.606	0.605

Clearly our baseline was already able to achieve very high train results, and better than random test accuracy as well as decent precision. These baseline results show the power of the models we chose, that even with very minimal hyperparameter tuning they are able to achieve great results. After this initial test, we then trained our other models in full and achieved the following results:

All Model Train Results				
	Accuracy	Precision	Recall	F-1
RoBERTa	0.993	0.993	0.993	0.993
XLNet	0.981	0.981	0.981	0.981
BigBird	0.996	0.996	0.996	0.996
Ensemble	0.994	0.994	0.994	0.994

All Model Test Results				
	Accuracy	Precision	Recall	F-1
RoBERTa	0.606	0.700	0.606	0.605
XLNet	0.637	0.725	0.637	0.639
BigBird	0.665	0.740	0.665	0.668
Ensemble	0.608	0.598	0.608	0.602

Our model is clearly overfitting to our training dataset. Even after the first epoch, most of our models were able to achieve greater than 90% accuracy on the training set, and on the order of 60% accuracy on the test set. Moreover, for every model except BigBird, additional training past epoch 1 only seemed to decrease the test accuracy. Since the by publisher dataset contains < 100 publishers, it would be easy for the model to learn which publishers post hyper-partisan content as opposed to recognizing the hyper-partisan content of the article text itself. This likely contributed to the poor model test results.

Disappointingly, we can also see that our best ensemble model was unable to achieve results that were better than any model individually. Even the lowest performing pretrained transformer model, RoBERTa was able to outperform the ensemble model on the test set. We think that we likely needed to optimize the parameters of the classifier layer more to achieve better results, or experimented with slightly stronger regularization such as weight decay to avoid the overfitting behavior.

4.2 Classifying Articles from the Stanford Daily and the Stanford Review

The Stanford Daily is the biggest student run newspaper on campus that issues factual reporting of events on campus, as well as opinion pieces, satire, and content reviews. The Stanford Review is a smaller student run newspaper that aims "to present alternative views on a wide range of issues, create a forum for rational debate on campus, and challenge those who disagree to participate"[9].

From our experimental results, its clear that of the four different models we trained, BigBird had the highest performance. We decided to evaluate some hand picked articles from the Stanford Daily and Stanford Review using our finetuned BigBird model. In total, we picked 13 articles from the Stanford Daily and 13 articles from the Stanford Review covering the same events.

BigBird Classification Results			
Event	Date	Daily	Review
Removing Home Equity from Financial Aid	Jan 2019	3	2
Student Senate on Free Speech	May 2019	4	4
Ben Shapiro Talk	Oct 2019	4	0
Mike Pompeo Talk	Jan 2020	4	4
Condoleeza Rice Protest	Mar 2020	4	0
George Shultz Passes	Feb 2021	3	0
Laptops Stolen	Feb 2021	1	0
36 Sports Strong Rally	Apr 2021	3	0
Free Palestine Rally	May 2021	3	0
TDX Death Updates	May 2021	4	0
Noose on Campus	Nov 2021	4	4
Views of Kyle Rittenhouse	Dec 2021	3	0
Investigation of Bias Against Men	Dec 2022	1	0
Mean		3.231	0.769

Table 2: Results for the 26 Stanford Daily and Stanford Review articles

These results reveal a couple of interesting insights into our transformer model. Clearly the model is capable of learning some information about bias, as expected the majority of labels for the Stanford Review name it as right leaning, and the majority of labels for the Daily are either Left or Center Left. Our model also predicts that the Stanford Review is more hyperpartisan, since all but one of the articles was labeled hyperpartisan. However, there are a couple of notable outliers where our model predicts that Stanford Review article as left biased instead of right biased. On the other hand, the Stanford Daily has an even mix of articles that are predicted to be hyperpartisan, and articles that are not. While there are a few articles there are labelled as right leaning, this is not too surprising and is within the bounds of reason. These classifications support our initial hypothesis that the Stanford Daily is a left leaning publication and the Stanford Review is a right leaning publication.

On this small dataset of 13 articles, the mean score of the Stanford Daily is 3.231, closest to Center-Left and the mean score the Stanford Review is 0.768, closest to Center-Right. Interestingly, these scores are equidistant from 2, which is the score for Center.

5 Discussion

Because our model was trained on the data labeled by publisher, our model was obviously able to fit the by publisher data well, and in terms of training metrics was able to outperform the best models from the SemEval 2019 competition on every metric. However, because we did not use any data labeled by article in our training or evaluating, we performed poorly on our test set when classifying articles labeled by article. Even though the by publisher dataset is incredibly large, it likely generalizes across entire publishers which can have a diverse range of article bias, leading to the mis-classification of very informative articles. The model would learn a lot from an unbiased article that comes from an outlet that primarily publishes hyper-partisan content, or by contrast, a partisan article that was published by a generally unbiased publisher. The by publisher dataset cannot capture these intricacies, leading to our poor performance. The fact that there is such a large discrepancy makes us suspect that all of our models were overfitting to the training dataset. We suspect that our models became quite good at identifying the publisher as a proxy for the desired bias label, but never actually learned very much about what features in the language actually suggested hyperpartisanship. To test this, we could have further finetuned our models to predict the publisher, and if we succeed even while freezing gradients in the transformers and only allowing updates in the linear classifier built on top, then we could be confident this is the case.

We also found that our ensemble model performed marginally worse than the BigBird model did individually. While in theory it should have been possible for our classifier to learn to simply pass on BigBird's outputs and ignore the other two models, the addition of the bad data from the other two models made our ensemble model worse, even after experimenting with tuning hyperparameters and hidden layer sizes. Additionally, by only ensembling over only transformers, we did not diversify the model types to capitalize on other strengths from other models, like CNNS or RNNS.

6 Conclusions

In this paper, we explored the classification of hyper-partisanship in news articles using an ensemble model of various transformer models. Our model's performance shows how even though pre-trained transformers are powerful tools in natural language processing, they cannot alone achieve high accuracy on a classification task. Despite the challenges, our best model was able to achieve a final classification accuracy of 67% and we were able to use the model to analyze the potential biases in two sources of relevant news sources for Stanford students. From initial analysis, our model indicates that the Stanford Daily is left-leaning and the Stanford Review right-leaning, and the Stanford Review publishes more hyper-partisan articles.

Our study provides a foundation for future work in this area, including the development of more sophisticated models, incorporating additional features, and exploring the effectiveness of alternative data pre-processing techniques. Specifically this could entail ensembling our model with a successful model from the SemEval 2019 competition, introducing features like word embeddings similar to how successful teams did, or acquiring more hand labeled data to incorporate into our training.

References

[1] Lee Drutman. We need political parties. but their rabid partisanship could destroy americannbsp;democracy., Sep 2017.

[2] Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. SemEval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[4] Aishwarya V Srinivasan. Xlnet explained in simple termsnbsp;!!, Aug 2019.

[5] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[6] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.

[7] Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, David Corney, Payam Adineh, Benno Stein, and Martin Potthast. Data for pan at semeval 2019 task 4: Hyperpartisan news detection. 2019.

[8] Adrien Barbaresi. Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131. Association for Computational Linguistics, 2021.

[9] Stanford Review. About us, Jan 2023.