Breaking a Fake News Classifier

ECS 170 Final Project

By: Matthias Gabel, Kaylee Christianson, Josh Vu,
Ananya Deepak, Kasey Baysa, Hanlin Chen, Chloe Lai

June 27, 2024

1 Introduction

Fake news has always posed a threat to society due to its ability to spread misinformation, erode trust, and instigate polarization. While there are dedicated communities working to dissolve the spread of misinformation through fact checking resources, the sheer volume of articles present on the internet makes the task of discriminating what is real and fake extremely difficult, therefore the implementation of a fake news detector employing AI and NLP (natural language processing) techniques would be extremely beneficial to the cause. Even if existing models have excellent performance, it is imperative to scrutinize and analyze these models given the impact fake news has on society.

This project will evaluate the model's generalizability, which is defined as the model's ability to maintain high accuracy with new data or changed parameters. We use this definition of generalizability to evaluate if the model has found a robust and generalizable pattern in the fake news or if the model has overfitted and simply memorized the training data.

2 Background

2.1 What is Fake News?

The term 'fake news' is a catch all term popularized by the 2016 US Presidential election. Given 'fake news' is a super charged term from political discourse, it is no surprise that its definition is highly complex. Linguistic literature takes a fine-grained approach to this definition, defining 'fake news' as an article containing incorrect information with authorial intent to maliciously deceive audiences. This is distinct from an article that is written in good faith, but happens to contain incorrect information. (Grieve & Woodfield 2023).

Fake news always peaks whenever there is public disarray due to any sort of nationwide/worldwide issue. Because these events typically cause high levels of uncertainty, people are more prone to believing in anything they see due to lack of information. For example, when measels, mumps, and rubella were present in the United States, a 1998 paper made a false claim that the vaccine meant to protect against all caused autism in children (Caceres et al., 2022). This false claim about the vaccine meant that parents were less likely to vaccinate their children, leading to more children losing their lives to those diseases.

2.2 Binary Classifiers

A binary classifier is any system that takes some sort of input, and outputs one of two choices. Some fake news classifiers fall under this category as they mark articles as either 'reliable' or 'fake'.

Based on linguistic literature and reported methods of tagging, this approach is significantly less fine grained. Notably, a Kaggle data set by Emine Bozkuş states that fake articles were "collected from unreliable websites that were flagged by Politifact (a fact-checking organization in the USA) and Wikipedia" (Bozkus, 2022). While the sites themselves are tagged as unreliable by human fact checking sources, this may present as problematic as the entire website, rather than individual articles being considered as unreliable.

Binary classifiers never check for partials; it will always be one choice or the other. Sometimes, articles may only contain partially false information, such as one paragraph having inaccurate information. If a binary fake news classifier parses an article, the output will always be 'reliable' or 'fake', and will not consider something like "partially fake". Due to this nature, it's imperative to ensure classifiers are able to be as accurate as possible with both the articles it's trained on, as well as any new articles inputed for classification.

3 Methodology

3.1 Model Architecture

We will analyze the performance of a fake news detector by Abdeladim Fadheli and Adrien Payong. This model uses the bert-base-uncased transformer model. BERT is a text classifier that uses both past and future text information to predict a word given its surrounding context. BERT is trained on English Wikipedia and BookCorpus, a database of unpublished books. It uses two types of methods to train itself–"Masked Language Modeling" and "Next Sentence Prediction." In the Masked Language Modeling task, the model randomly masks a small percentage (∼15%) of texts. BERT then has to predict what texts were most likely used in the masked spot. In the Next Sentence Prediction task, the model concatenates two sentences as inputs and has to predict whether the two sentences are cohesive and relevant to each other or not.

The bert-base-uncased version of BERT modifies BERT to read news articles and predict whether an article is reliable or unreliable (binary coding). The "base" indicates a BERT model that includes 110 million trainable parameters and 12 transformer block layers, and the "uncased" indicates that text capitalization is disregarded (ex, "HELLO" == "hello").

3.2 Training Methods and Parameters of the Model

The model uses a Fake News data set by William Lifferth. This data is stripped of all 'null' values and tokenized using BertTokenizer. After being tokenized, the data is converted into a Pytorch dataset so it can be used in our model's class parameters. The model utilizes an open-source transformer model class from Hugging Face, which utilizes gradient descent and the AdamW optimizer function for training parameters. AdamW disregards weight decay in the optimization process. The transformer class takes in:

- `model`: The type of model architecture to use (ex: bert-based-uncased)
- `args`: Training arguments for the model
- `train_dataset`: Pytorch training dataset
- `eval_dataset`: Pytorch validation set
- `compute_metrics`: Function that computes metrics. Our function utilizes scikit functions that compute accuracy, F1, and AUROC.

And for the `args`, it's initialized with a separate class that takes:

- `num_train_epochs`: Total number of epochs to train with
- `per_device_train_batch_size`: Total number of samples to train with per batch
- `per_device_eval_batch_size`: Total number of samples to evaluate with per batch
- `logging_steps=200`: Number of steps to log current metrics and training progress
- `learning_rate`: Rate at which the model moves toward the optimal value for a loss function

For the sake of time efficiency and limited computational power, the model uses 1 training epoch, as well as a 'batch size for evaluation' equal to 5 and 'batch size per device' equal to 5 for parameter training. Metrics were saved every 200 steps to track how the model's performance would change over time. The learning rate will start at 0.00005, however will later be increased for testing.

3.3 Tests of Generalizability

A fake news detector should be able to detect any fake news article, even if it's not originally from the training or test data in the scaffolding. To know if the model is generalizable enough to feel confident in predictions, we will run a series of tests to see how easily the model will overfit. Overfitting will tell us if the model is too accustomed to its training data and if it responds well to new data. Because one method may not be enough information to indicate overfitting, we tested by training with smaller sample sizes, larger learning rates, and different datasets that don't come from the scaffolding.

3.3.1 Sample Size

The original model is trained with approximately 20,000 samples, so we will try runs that train with 10,000, 5,000, 1,000, and 100 samples. Testing with smaller sample sizes is one way of checking for overfitting. If model performance suffers with smaller training sets, then the model would likely fit too close on the training set and possibly be sensitive to noise within the data. This is a method for us to see if the model is simply memorizing the training data or actually learning the linguistic patterns that separate the real articles from the fake.

3.3.2 Learning Rate/Step Size

We tested on the default learning rate of 0.00005, then tried increasing the learning rates of 0.0005 and 0.005, which are already considerably bigger than 0.00005. If the model performs poorly with larger learning rates, it means it struggles to converge to an optimal solution and will likely cause oscillating performance. This is another way for us to see if the model simply trains very closely on specific strings or is capable of recognizing patterns in data that deem an article "false". If we observe oscillation, it suggests that writing styles may be too complex and that it can't pick up notable linguistic patterns in broad texts.

3.3.3 Testing on an Alternative Dataset and AI Generated Articles

For this method, we'll use whichever model yields the best performance. Testing the best model with different datasets lets us measure its ability to make predictions with articles not found in the original scaffolding. While the original scaffolding includes a test dataset, it was created by the same author, meaning they could be articles that use similar writing styles to the training data. To determine the generalizability of the model, we tested the model with an alternative dataset by Sameer Patel that contains articles marked as real and fake. One of the tests consisted of 5000 real articles and 5000 fake articles to determine the F1 and AUROC. The other tests consisted of either real or fake articles (10,000 samples each) to directly monitor how often thee model can actually make a correect prediction. While these homogeneous tests cannot calculate the F1 and AUROC, they are necessary to determine how well the model performs at detecting if an article is real or if an article is fake.

Lastly, we created a test data set of AI-generated articles as a separate measure. This was done in hopes of finding insight about how the model is successfully identifying fake news. Using ChatGPT 3.5, 3 new test data sets were created:

1. **Trigrams**: AI-generated articles created using the most prevalent trigrams found within the reliable article training dataset (see Figure 1).

2. **Reliable News Writing Style**: AI-generated articles emulating the writing style characteristic of an article in the reliable article training dataset (common words in 'reliable' articles shown in Figure 2).

3. **BBC News Writing Style**: AI-generated articles emulating the writing style of a BBC News article (FitzGerald, 2024)

The topics in the original training and testing datasets were all similar (see Figure 2). The AI generated dataset has more subject variety because we provided the AI with a subject (i.e politics, tech, etc) and prompted it to write about any news in that sector. Each method produced 50 articles, for a total of 150 AI-generated articles.

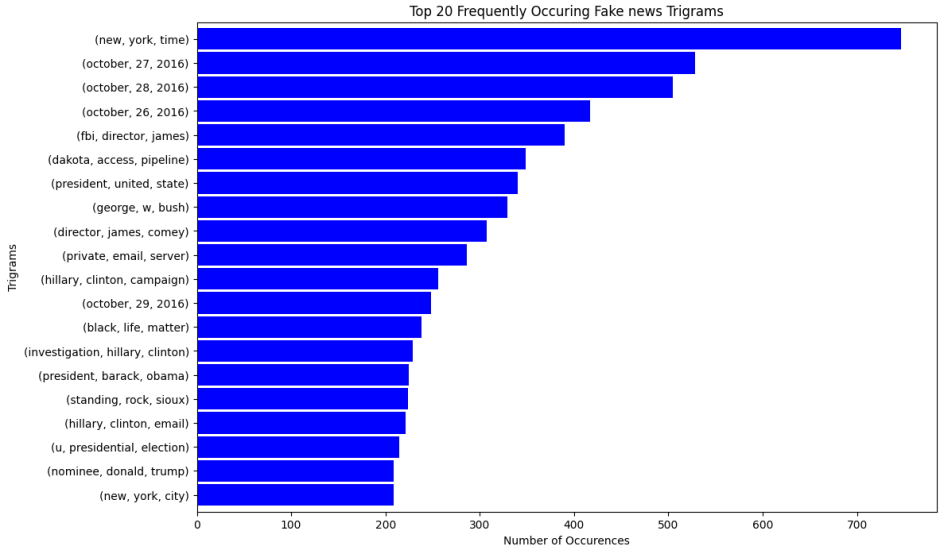


Figure 1: The top 20 Frequently Occurring Fake News Trigrams from articles marked as "reliable" in our dataset.

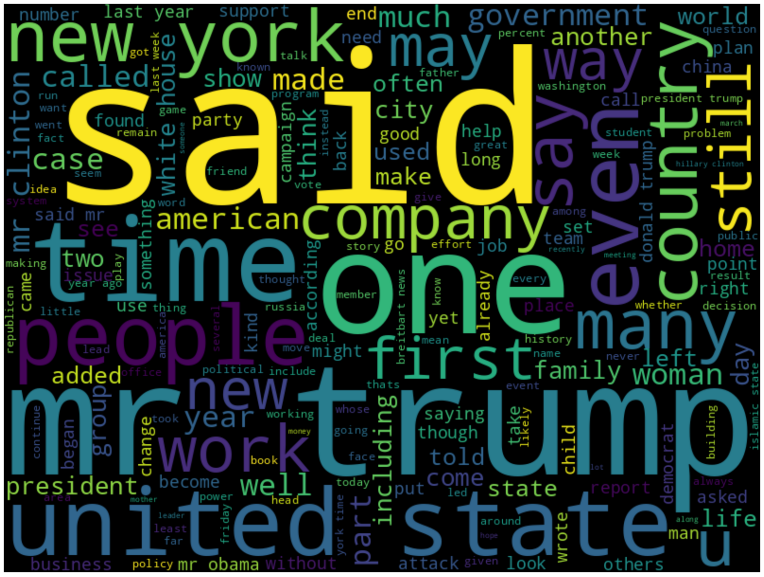


Figure 2: A WordCloud representing the most frequent words present in real articles, where the larger the word the more common it is.

3.4 Metrics and Data Collection

We chose three metrics to measure model performace: accuracy, F1 Score, and Area under the Receiver Operating Characteristic Curve (AUROC). For every 200 steps in training, the model calculates the current accuracy, F1 score, and AUROC score using functions from Python's scikit library for each metric. At the end of each training session, a table is generated (see Table 1) which is how we create line graphs using matplotlib.

At the end of each training session, the Hugging Face transformer library allows us to use a function .evaluate() which retrieves the average value for each metric for the model.

Table 1: Sample Training Data for each Metric returned after some training session.

Step Size	Training Loss	Validation Loss	Accuracy	F1	AUROC
200	0.368	0.034	0.994	0.993	0.999
400	0.048	0.015	0.998	0.997	0.999
600	0.013	0.021	0.997	0.997	0.999
800	0.046	0.146	0.973	0.968	0.999
1000	0.030	0.021	0.995	0.994	0.999
1200	0.010	0.011	0.999	0.998	0.999
1400	0.007	0.013	0.998	0.998	0.999
1600	0.000	0.014	0.998	0.998	0.999
1800	0.020	0.013	0.998	0.998	0.999
2000	0.026	0.008	0.999	0.999	1.000
2200	0.007	0.012	0.998	0.998	1.000
2400	0.000	0.009	0.999	0.999	1.000
2600	0.008	0.015	0.998	0.998	1.000
2800	0.016	0.014	0.998	0.998	1.000

3.4.1 Accuracy

Accuracy measures how well a model can correctly classify articles as real or fake. It's also a way to benchmark how confident we are in predictions with inputted articles after training has completed. Accuracy can be calculated with:

$$Accuracy = \frac{\#\ of\ correct\ predictions}{length\ of\ data}$$

3.4.2 F1 Score

For a classifier model, two important metrics to measure performance are precision and recall. Precision measures what proportion of positive predictions were true positives (samples marked 'reliable', and were actually reliable). while recall measures what proportion of actual positives were coded as true positives. Precision and recall are almost identical in their calculations:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Because both are quite similar, we chose to measure F1 scores because the F1 score combines precision and recall. This allows for a more concise and general measure of performance. Using the equations for precision and recall, the F1 score is then calculated as:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

3.4.3 AUROC Curve

The AUROC tells us how correct our model can actually distinguish what's reliable or fake. A Receiver Operating Characteristic (ROC) curve is a method of visualizing the rate of how well a model classifies true positives and true negatives, whereas AUROC gives us a more summarized version of ROC curves across all its thresholds (Google, 2022). While accuracy and F1 can measure overall performance, the AUROC measures the ability the model has to make distinguish samples. A high AUROC indicates excellent resolution, an AUROC at 50% is the same as someone making random guesses, and an area near zero means the model makes practically all of its decisions incorrectly.

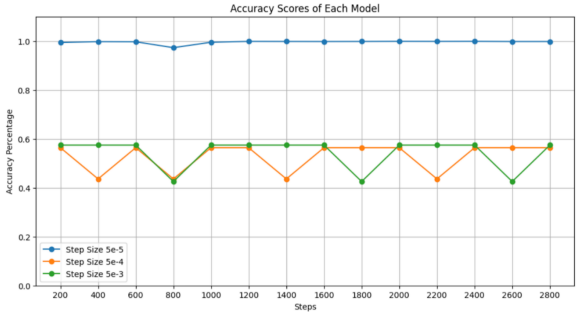
4 Results

4.1 Model Performance With Various Learning Rates

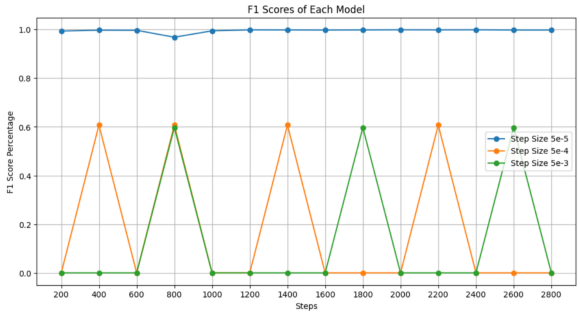
With the learning rate = 0.00005, the model excels in all metrics, with each being greater than 99%. Training and validation losses are also extremely low being no greater than 0.05.

When the learning rate is increased, there is a considerable loss of stability in the model's performance. Accuracy scores drop to approximately 50% and F1 scores plummet to 0, suggesting the model can't confidently classify true positives. The AUROC scores also show that with a 0.0005 learning rate, the model makes mostly incorrect decisions, and at 0.005, it's only as good as guessing (see Table 2). Models with increased learning rates consistently oscillate and minimally improve across 2800 steps. This erratic behavior is consistent with the training and validation loss graphs in Figure 4.

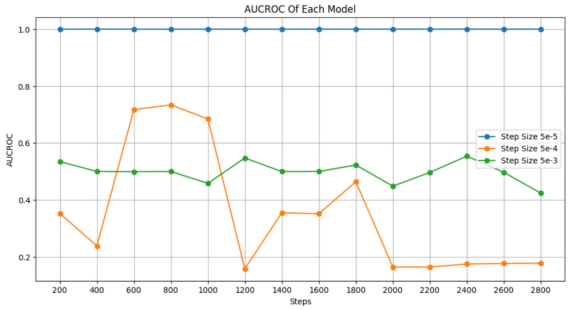
Models with larger learning rates will converge on a result much quicker, updating their weights more significantly, resulting in the instability. These results show us that the model is sensitive to the data and susceptible to overstepping the most optimal point in the gradient descent function, leading to oscillations observed in the graphs.



(a) Accuracy Scores during training.



(b) F1 Scores during training.

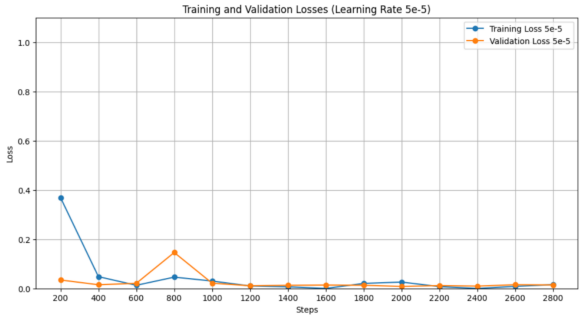


(c) AUROC Scores during training.

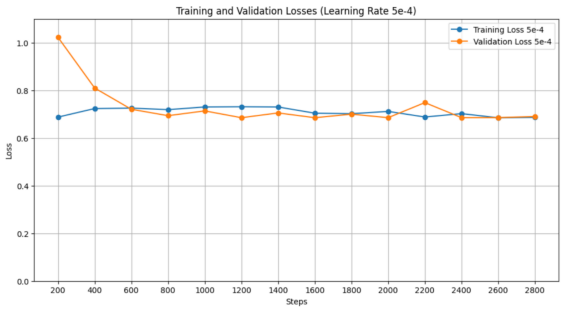
Figure 3: Metrics of each model trained with learning rates $5.e-5$, $5.e-4$ and $5.e-3$.

Table 2: Model Evaluation Metrics with Different Learning Rates from Hugging Face's '.evaluate()' Function.

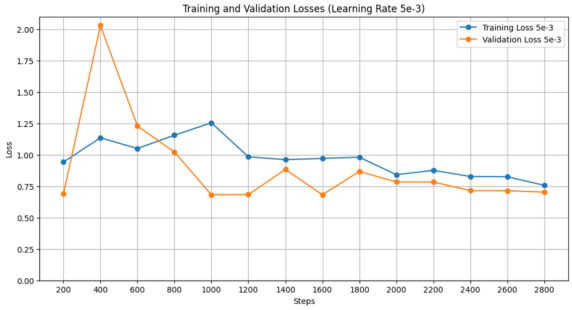
Learning Rate	**Test Accuracy**	**Val. Accuracy**	**F1 Score**	**AUROC**
$5.e-5$	0.99862	0.99890	0.99875	0.99999
$5.e-4$	0.49285	0.56385	0.00000	0.15830
$5.e-3$	0.49285	0.56931	0.00000	0.50000



(a) Learning Rate $= 5.e-5$.



(b) Learning Rate $= 5.e-4$.



(c) Learning Rate $= 5.e-3$.

Figure 4: Training and Validation Losses during training of each model with learning rates $5.e-5, 5.e-4$ and $5.e-3$.

Table 3: Model Training and Evaluation Loss with Different Learning Rates from Hugging Face's '.evaluate()' Function.

Learning Rate	**Training Loss**	**Val. Loss**
$5.e-5$	0.04284	0.00812
$5.e-4$	0.70908	0.68513
$5.e-3$	0.96867	0.56931

4.2 Model Trained with Varying Sample Sizes

At all sample sizes $\geq$1000, the model achieves excellent performance with accuracy, F1, and AUROC topping out within 99-100% with very minimal loss. The only time it performed worse was when we decreased the sample size all the way down to 100, where accuracy was 77%, F1 was 72% and AUROC was 80%. In short, increasing the sample size yielded improvements in performance, however improvements became marginal with at most a 0.01% increase in accuracy between the 5000 and 10,000 datasets.

Without running the other tests, this would indicate the model likely doesn't overfit since it doesn't negatively respond to small decreases in sample size. However, its ability to still achieve extremely high training performance metrics, despite only with 5% of the original training set, can still imply it's adapting too closely to the training data.

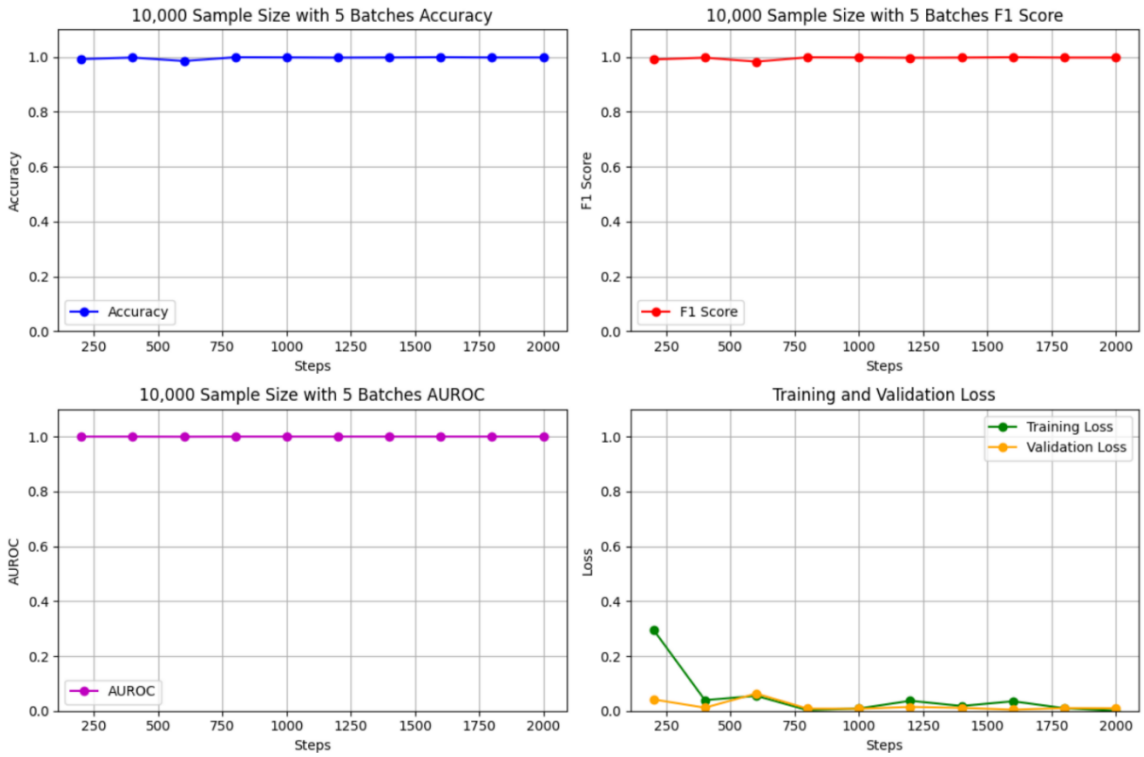


Figure 5: Accuracy, F1 Score, AUROC, and Training and Validation losses of the model trained with learning rate $5.e-5$ and 10,000 samples.

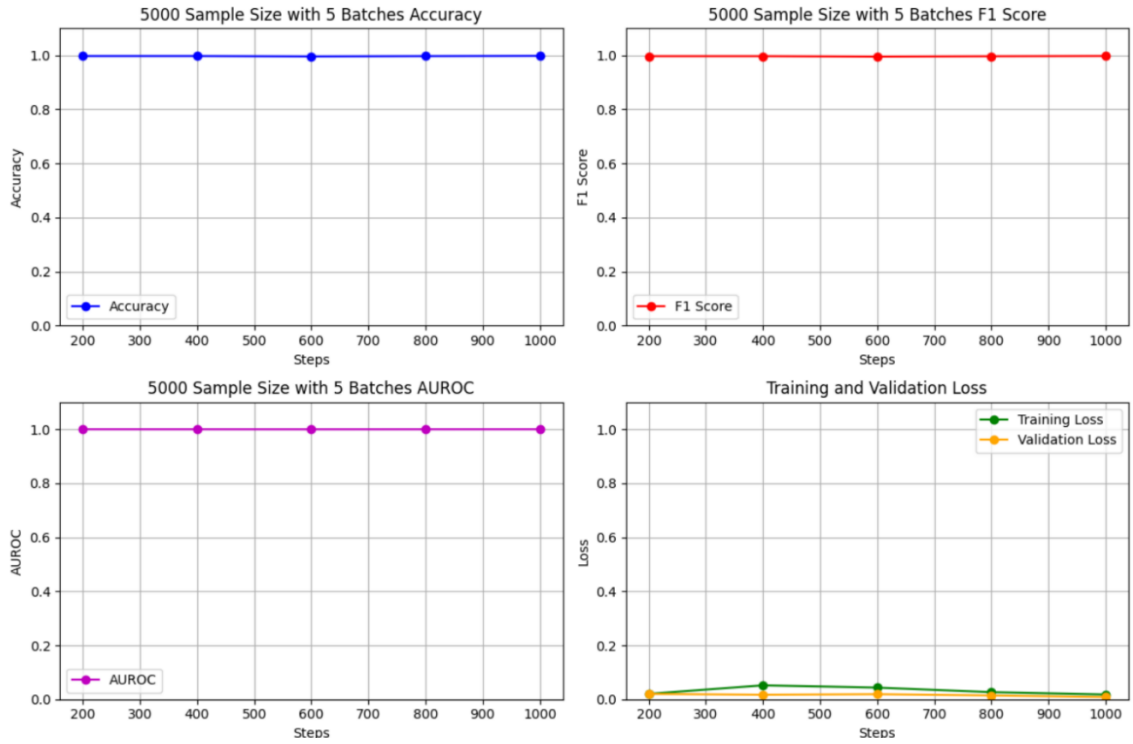


Figure 6: Accuracy, F1 Score, AUROC, and Training and Validation losses of the model trained with learning rate $5.e-5$ and 5,000 samples.

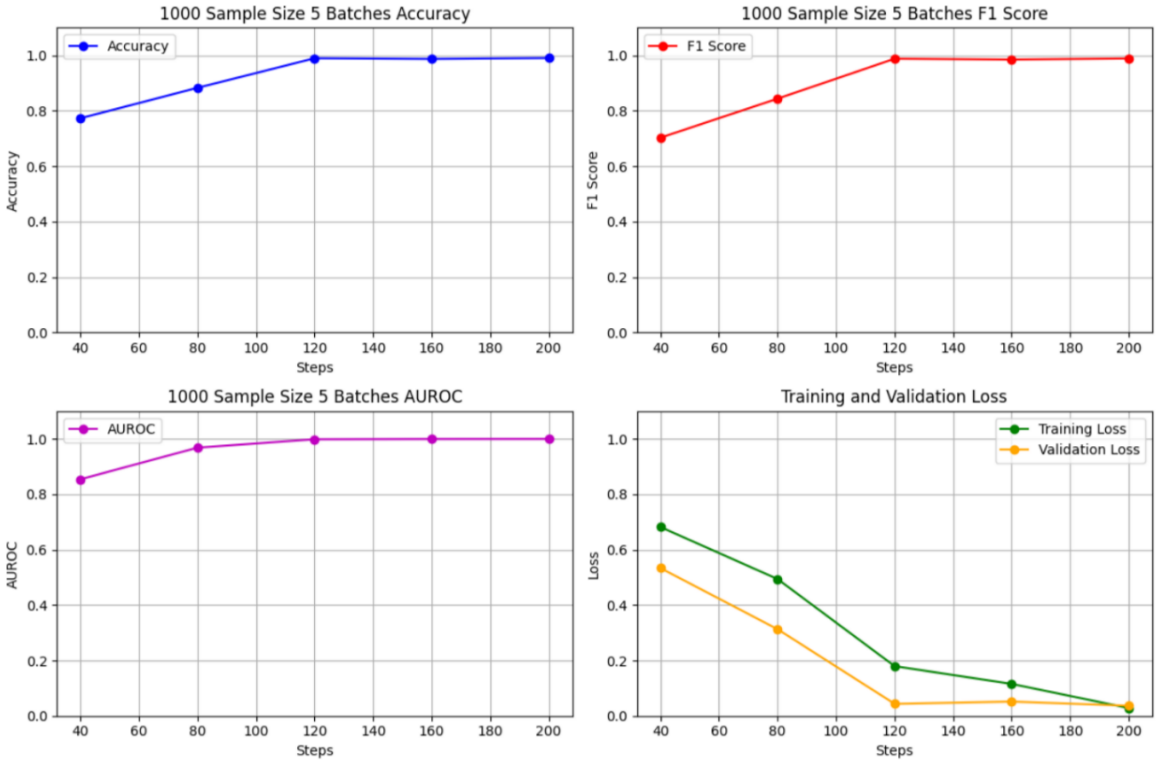


Figure 7: Accuracy, F1 Score, AUROC, and Training and Validation losses of the model trained with learning rate $5.e-5$ and 1,000 samples.

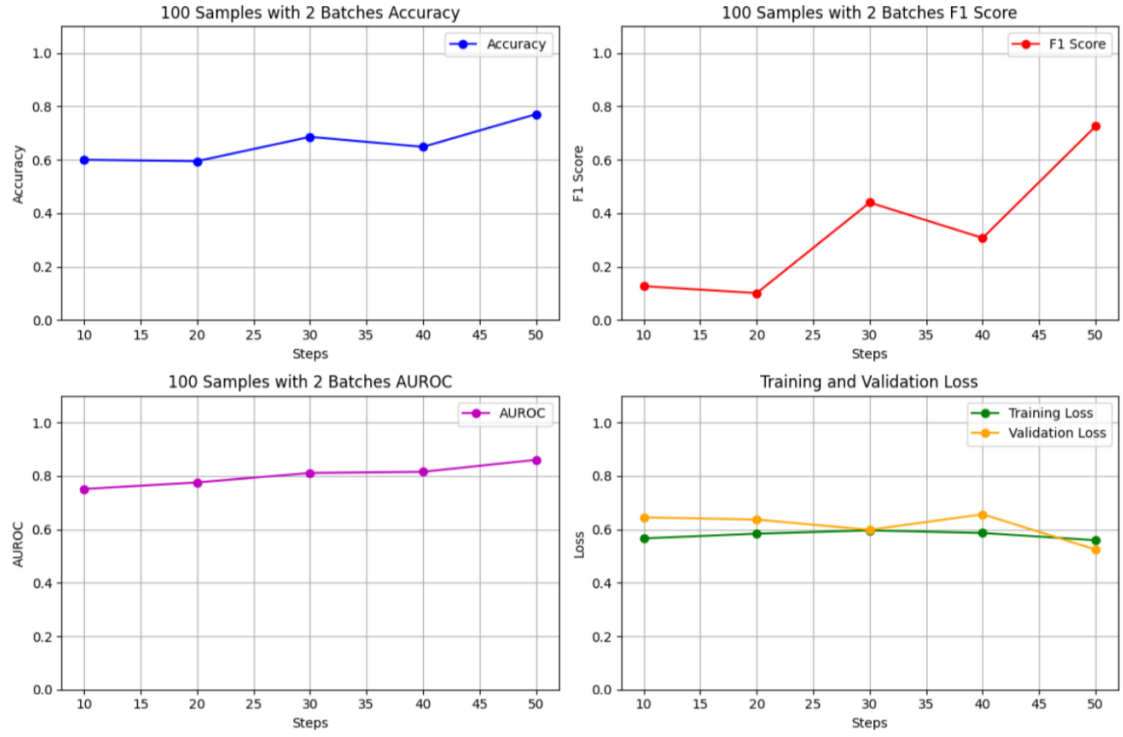


Figure 8: Accuracy, F1 Score, AUROC, and Training and Validation losses of the model trained with learning rate $5.e-5$ and 100 samples.

Table 4: Model Evaluation Metrics with Different Sample Sizes from Hugging Face's '.evaluate()' Function.

Samples	Test Accuracy	Val Accuracy	F1 Score	AUROC
100 samples	0.77005	0.77085	0.72596	0.86055
1000 samples	0.99560	0.99043	0.98885	0.99976
5000 samples	0.99835	0.99809	0.99774	0.99999
10,000 samples	0.99917	0.99918	0.99903	1.00000

Table 5: Model Training and Evaluation Loss with Different Sample Sizes from Hugging Face's '.evaluate()' Function.

Samples	Training Loss	Val. Loss
100 samples	0.5593	0.52471
1000 samples	0.0281	0.03661
5000 samples	0.0176	0.00814
10,000 samples	0.0087	0.00452

4.3 The Best Model from Changing Learning Rates and Sample Sizes

Using the data from the tables, we generated a heatmap to put the metrics of every model trained together. The heatmap better visualizes how each model, besides the 100 samples, 5e-4, and 5e-3 models, have excellent performance, with their performance being above 99% and training losses below 0.05.

The model trained on 10,000 samples has the best performance overall, with only a sliver of a difference between models like the model trained with the learning rate 5e-5. However, the model trained on a 5e-5 learning rate may be better since it was trained on more data; it utilized the full 20,800 samples of the training set compared to 10,000.

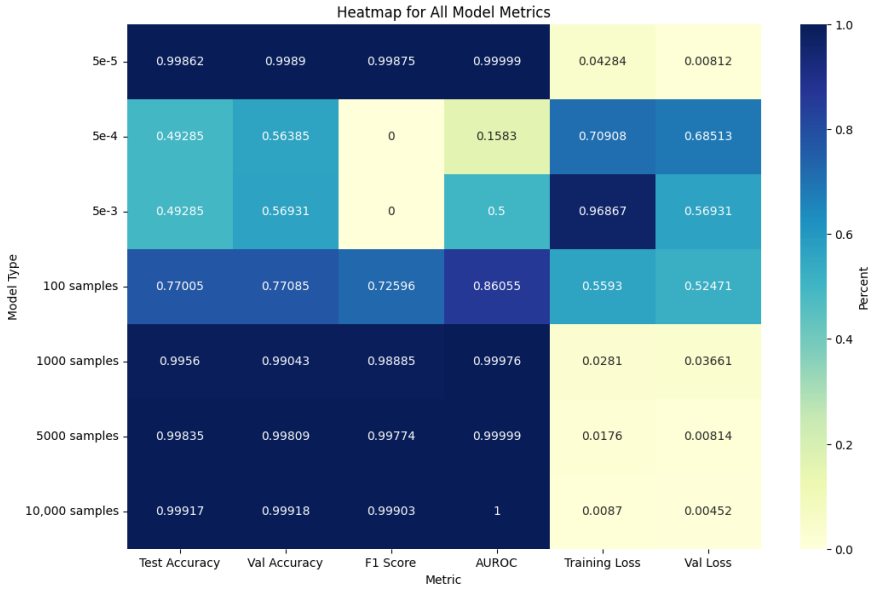
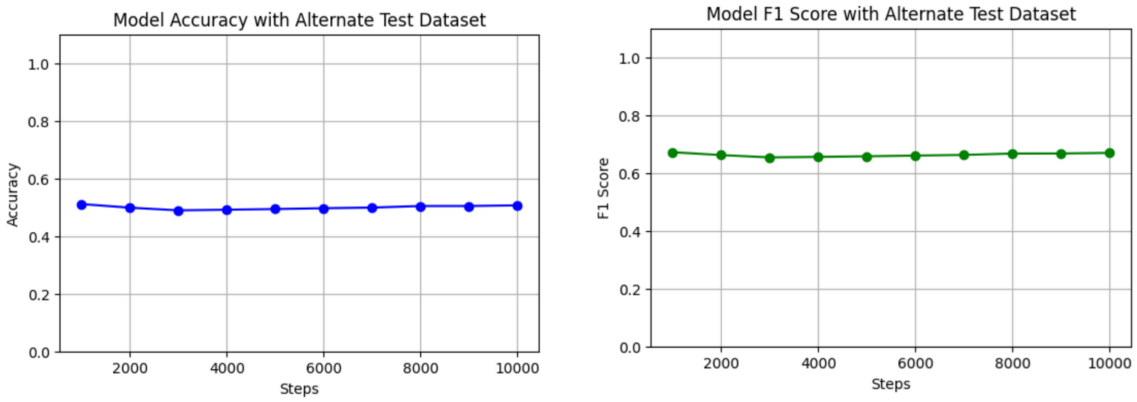


Figure 9: A Heatmap displaying all models trained with varying sample sizes and learning rates with their metrics.

4.4 Model Tested on an Alternate Dataset

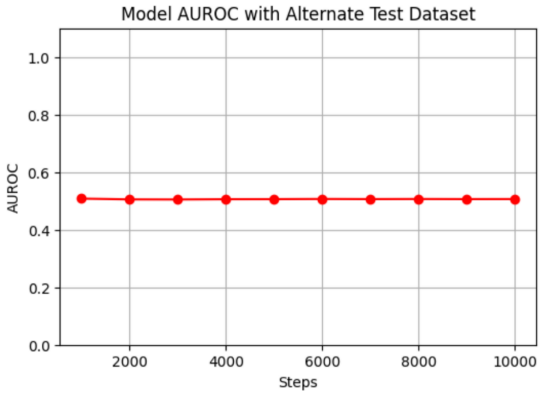
Because it had the best overall performance trained on the **entire** dataset, we used the model whose learning rate was 0.00005. The accuracy, F1 score, and AUROC for the alternative dataset were 50.65%, 66.92%, and 50.65%, respectively, starkly contrasting with the higher values seen in the original dataset. An accuracy and AUROC of 50.65% implies that the model classified only about half of the articles as real or fake which is essentially no better than random chance. The F1 score being 66.92% suggests that in datasets other than the original the model's precision and recall are extremely suboptimal. This suggests that there's a decrease in the model's universal ability to differentiate between real and fake articles among a variety of different articles. The accuracy of the dataset consisting of all reliable articles was only 1.32% and the accuracy of the dataset consisting of all fake news was 99.81%. This implies that the model is inadequate at determining if an article is real, but can reliably tell if an article is fake.



(a) Accuracy Scores

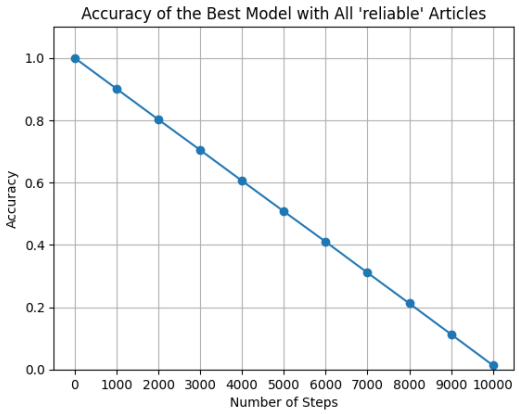
(b) F1 Scores

Figure 10: Metrics of the best model with the alternative dataset (Accuracy and F1).

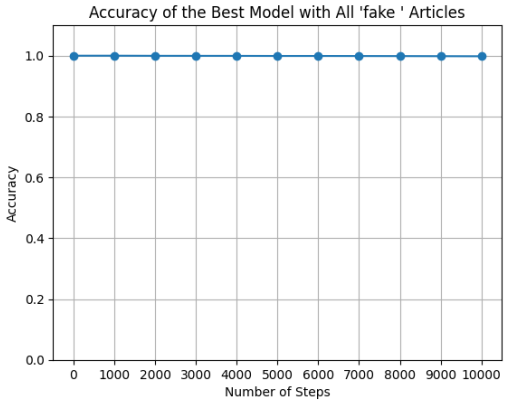


(a) AUROC Scores

Figure 11: Metrics of the best model tested with the alternative dataset (AUROC).



(a) Accuracy score of real articles from alternate dataset (Final: 1.32%).



(b) Accuracy score of fake articles from alternate dataset (Final: 99.81%).

Figure 12: Metrics of the best model with the alternative dataset (Accuracy and F1).

4.5 Model Accuracy on AI-Generated articles

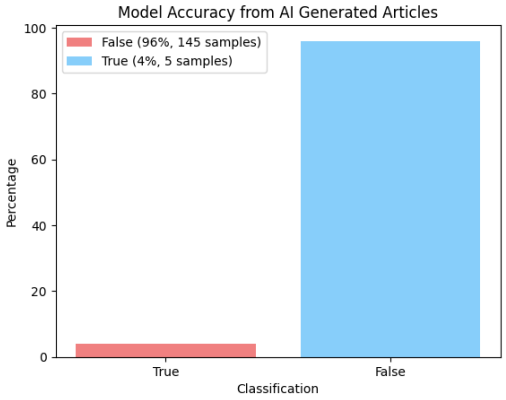


Figure 13: Distribution of AI Articles Classified as Real or Fake.

The model successfully categorized a majority of the AI generated articles as fake with 96.7% of articles categorized as unreliable (5 of 150 articles were deemed real). Of those 5 articles, all of them were articles writen using trigrams. It appears that when given texts that the model trained to associate with 'reliable' means our model can be prone to false positives.

5 Discussion

Our results mostly support the idea that Fadheli and Payong's original model isn't capable of understanding linguistic patterns and merely picks up on common strings, meaning it isn't generalizable. The tests for increasing step size caused significant instability and oscillation in the model's performance, suggesting that the model's parameters are highly sensitive to changes. This implies the model is memorizing common texts rather than linguistic patterns.

Alternatively, the model's continuously high performance on restricted sample size gave us reason to believe that there is a generalizable pattern that is salient in the texts. From only 100 samples, the accuracy rises from 0.6-0.8. Coupled with the minimal deviation between training and validation metrics, this implied there might have been some kind of discernible pattern.

Despite those results, the model continues failed to show generalizability with alternative data. At first, we thought the model was at least as good as guessing with an alternative dataset of true and false articles, yielding 50% across all performance metrics. However, once we introduced sets that had

one type, the model proved to incapable of discerning reliable articles. This may imply that the testing data provided with the training set are likely articles that share the same writing style as the training set, causing the higher performance when it might instead be overfitting.

Lastly, the model's ability to detect the majority of AI articles to be false aligns with the performance using the alternative dataset. The model likely found a pattern in reliable texts that GPT 3.5's transformer based architecture does not seem to pick up, however it is unclear if this is more indicative of the models discrimination abilities or GPT 3.5's generation abilities. Because the only articles falsely marked as 'true' were all trigram articles, it further supports the idea that the model doesn't do anything beyond memorizing common keywords.

While these results are somewhat conflicting, the black box that is language modeling is beginning to show light. Most notably, the form of data this model encounters appears to be extremely flawed.

Presupposing that there is a general pattern that makes fake news distinct from real news, it is thus necessary to scrutinize the methods of data collection. Binary classification is overly simplistic, given it negates the fact that an article can be partially true and partially false. This is even more of an issue when there is a variety of authors and writing styles which add noise to the data. Linguistic literature suggests that communicative styles are unique and individual, thus linguistic patterns of deception are going to vary significantly between authors (Grieve & Woodfield 2023).

To further problematize this issue, we can suspend the presupposition (that there is a general linguistic pattern in fake news). This opens up discussion about confounds that can explain the varied performance across different data sets. One likely source of this confound is article content. A majority of articles appear to be from US politics, as indicated by generated word clouds, whose most frequent words are 'Trump' and 'United States', which is a pattern consistent among available fake news data sets. Paired with the fact that these data sets pull from a narrower data set for reliable articles and a wide data set for unreliable articles (Bozkuş, 2022), it seems that the model may only be good at recognizing surface level consistencies between reliable articles. This is once again supported by the alternative hypo

In investigating this further, we would want to continue rigorous testing of the model, using metrics such as cross validation and regularization. Due to limited computational power, the batch sizes and size of data were limited, meaning that more robust and generalizable models may have been achievable if there was simply more computational power. Furthermore, the lack of significant data sets had limited our ability to investigate different types of discourse, as a majority of fake news data was centered around US politics, severely limiting the extent to which the model can perform.

6 Conclusion

Through rigorous evaluation of a fake news detector, we were able to determine that a high performing fake news classifier isn't generalizable. We deteremined that the model has picked up some general differences between real and fake news that cannot be reliably replicated by a large language model imitating real news. Additionally, the data the model is trained on is highly specific, contributing to its lack of generalizability.

While the model does have 99% accuracy on its given training data, it is clear to see that there are significant factors in the data collection methods and model parameters that contribute to its high performance outside of a concrete discernible and generalizable pattern that lies in the data.

This study reinforces the need to be skeptical and inquisitive when evaluating the efficacy of language modeling tools. The advent of a fake news detector does not necessarily mean that the burden of manual fact checking has been lifted, rather it raises responsibility for readers to look upon news with a more critical eye.

7 Contributions

Josh: Helped create alternative and AI datasets and helped analyze their results. Also did the write-up portion on testing the alternative and AI datasets.

Matt: Wrote out the scaffolded model, came up with and explained all the testing methods and metrics used, and trained models using varying learning rates. Worked on LaTeXtypsetting; creating

graphs and other visuals such as the WordCloud, heatmap, and line graphs; slides, and helped revise sections.

Kasey: Researched background, methods, and created discussion. Provided linguistic backing and literature to form argument. Edited deliverable and slides to create cohesive flow.

Kaylee: Researched background, developed introduction and motives behind project, helped research and explain graphed outputs. Helped write rough drafts of slides and deliverable sections.

Hanlin: Trained model on varying sample sizes and created graphs to summarize model performance with additional metrics, and wrote about the results in final report and the slides.

Ananya: Worked on the methods section and parts of the discussion. Created slides for the methods portion and created the appendix in the final deliverable.

Chloe: Researched the BERT model and methodology. Wrote the groundwork and outline for the methods section. Helped format the final deliverable's visuals and citations into LaTeX.

8 Appendix

HuggingFace: A machine learning platform where it helps users build, deploy, and train machine learning models. Includes Transformer library used to create the class for the model and use evaluation functions.

Matplotlib: Used to create data visualizers such as line graphs and pie charts.

Seaborn: Used to create heatmap to visualize metrics across all models and find the best performing one. Used to visualize data distribution to check if it's balanced.

NLtk: Allows python programs to work with human language data and process natural language. Made it easy for us to to NLP processing to visualize article contents using n-grams.

Numpy: Scientific computing library used to manipulate large amounts of data so that it can be used in ML models.

Pandas: Library for open source data analysis and manipulation. Used to convert .csv files to Python readable dataframes.

PyTorch: This has a wide variety of tools that support computer vision, natural language processing, and other ML algorithms. Converted our pandas dataframe into PyTorch datasets to split training and validation sets during trainig.

Sklearn/Scikit: Provided easy-to-use functions for calculating accuracy, F1 scores, and AUROC.

Transformers: Neural networks based on learning context and tracking sequential data.

9 References

1. Ferreira Caceres, Maria Mercedes, et al. "The Impact of Misinformation on the COVID-19 Pandemic." AIMS Public Health, vol. 9, no. 2, 12 Jan. 2022, pp. 262–277, www.aimspress.com/aimspress-data/aimsph/2022/2/PDF/publichealth-09-02-018.pdf, https://doi.org/10.3934/publichealth.2022018.

2. Bozkuş, E. (2022). *Fake News Detection Datasets.* Kaggle. https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets.

3. Grieve, J., & Woodfield, H. (2023). *The Language of Fake News.* Cambridge: Cambridge University Press.

4. Jurafsky, D., & Martin, J. (2023). *Speech and Language Processing.* Stanford University Press.

5. Lifferth, W. (2018). *Fake News.* Kaggle. https://www.kaggle.com/c/fake-news/overview.

6. Payong, A., & Fadheli, A. (2022). *Fake news detection in python - the python code.* Python Code. https://thepythoncode.com/article/fake-news-classification-in-python.

7. Patel, S. (2021). *Fake News Detection.* Kaggle. https://www.kaggle.com/c/fake-news/overview.

8. FitzGerald, James. "Texas Storms: Power Cuts Expected for Days." BBC News, 29 May 2024, www.bbc.com/news/articles/czdd5y9d25jo.amp.. Accessed 12 June 2024.

9. HuggingFace. "Trainer." Huggingface.co, 2024, huggingface.co/docs/transformers/en/main_classes/trainer.

10. Bozkus, E. (2022). *Fake News Detection Datasets* Kaggle. https://www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets.