Determining the Veracity of Social Media Articles using Large Language Models

Raghav Sharma

Received July 15, 2023
Accepted November 05, 2023
Electronic access December 15, 2023

In today's interconnected society, misinformation is one of the most prevalent dangers of social media. Social media networks like Twitter and Facebook have so much information on their sites that it is almost impossible to differentiate real from fake news. Having a system that can automatically detect the truth from lies can be very helpful for everyone on the internet and can help prevent false information from spreading. This paper focuses on detecting whether a certain post is referring to true or fake news. This is done by using a pre-existing data set on examples of real and fake news posts. I create a model based on these examples and use aspects such as text, title, and linked sources to predict the truth of the post. The specific models I tested were logistic, decision tree, and random forest classifiers, along with neural networks and Transformers. These models can predict the veracity of any article with very high accuracy, and in some cases, achieving 100% accuracy. With solutions like the ones explored in this paper, one can browse social media and be sure whether the article they are reading is true or fake, leading to a safer and more secure online experience.

Introduction

Social media has connected our modern world in a way that would be unfathomable even 50 years ago. While there are numerous positives to this societal change, there are some inevitable pitfalls as well. One of these disadvantages is the widespread disinformation that is being circulated through the internet. This disinformation can cause people to develop skewed ways of looking at the world, associate with groups they would not otherwise, or act in truly horrific ways. Misinformation caused many groups to develop anxiety and psychological distress (Rocha 2021)[1]. Another effect of the spread of misinformation is that this fake news dominates social media platforms. According to Jim Rutenberg, a New York Times writer, "Fake news dies hard". This means that fake news is more prominent in social media and is much harder to get rid of (Rimer 2017)[2].

While social media like Twitter have systems set up to detect fake news, these systems are often managed manually and are very inefficient. Twitter only has systems to stop the spread of misinformation after it has been posted. This leads to fake news being able to spread far before it is even caught by Twitter. With solutions discussed in this paper, social media posts can be processed at the time of posting to flag whether the post is citing fake or true news articles. Without this, fake news will continue to go viral and influence the human population in disastrous ways (Dizikes 2018)[3]. Figure 1 shows the number of fake news social media posts compared to the number of posts about fact-checking those posts. This large disparity between the numbers of fake news and fact-checking posts demonstrates the urgency of fact checking systems in today's social media.

To stop this from happening, there needs to be a way for people to be able to discern between what is true and what is fake news on the internet. My research presents a solution to this problem using machine learning and deep-learning technologies to detect and label fake and true news. Using machine learning to approach this problem is preferred because there is a vast array of articles with both true and fake news, so there is no scarcity of data. A machine learning solution to this problem also offers an instant detection of fake news, which is preferable to the slow, manual techniques utilized today.

There is also a large audience for this solution, and using machine learning can help the solution adapt and develop to the current circumstances of social media. This paper uses a dataset of thousands of tweets that come from true and fake sources. The methods of data organization and categorization are then discussed. Next, the various model archetypes are tested and the model with the highest is calculated. Finally, the implications of these results are discussed along with ways the research can be built upon in the future.

Data

The dataset used for this research is a publicly available dataset that compiles thousands of Twitter articles the source of true and fake news articles (Bisaillon 2020)[4]. The dataset then takes the text of these articles and labels them as true or

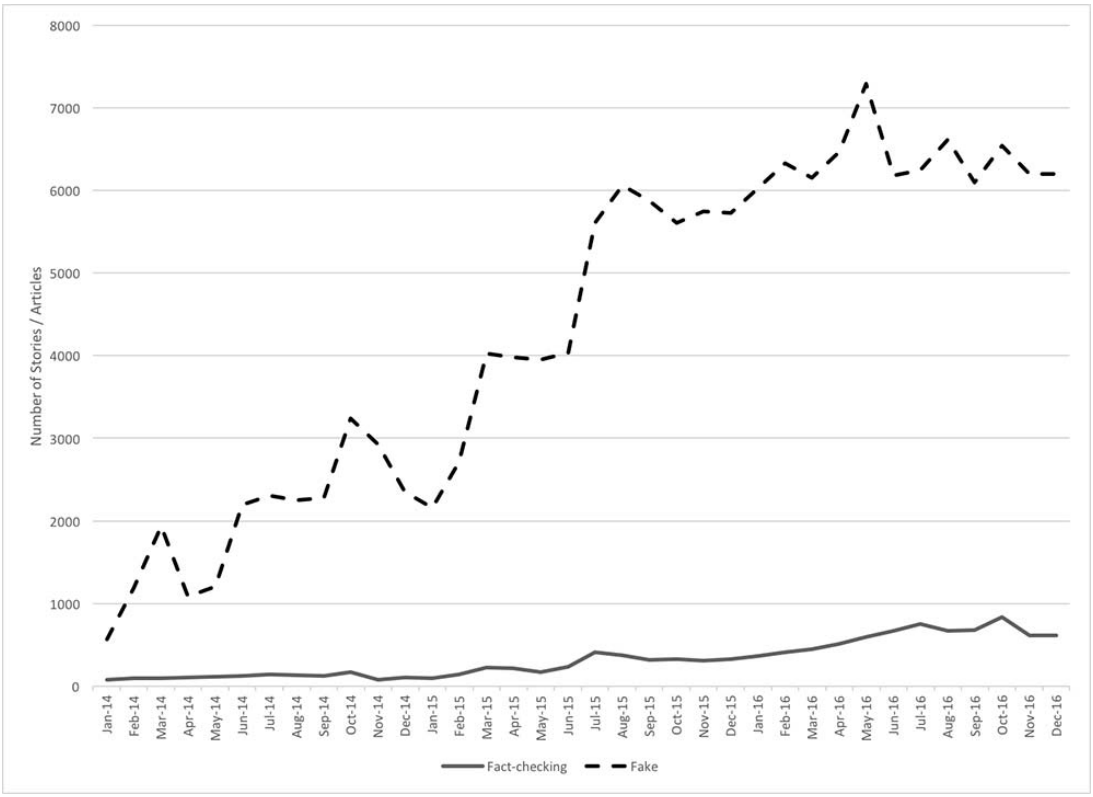

Fig. 1 Number of Fake news posts vs. Fact-checking posts (Figure taken from Rimer 2017)[2]

false. The dataset consists of 21,418 true news articles and 23,538 fake news articles. The balanced numbers of fake and true news articles show that this is a well-balanced dataset. To get a better grasp of the data I use, I conducted some experiments on the dataset, namely examining the frequency of certain words and n-gram analysis. An n-gram is a group of words that are n-words long. Firstly, I analyzed what words are the most common in both sets of data, true and fake. Both datasets have comparable results, the words "Trump" and "said" are the most frequently used words, with "Reuters" being used noticeably more in the true articles.

Next, I did an n-gram analysis of the data. For example, "Hello there" is a bi-gram, and "Real or Fake" is a tri-gram. The top tri-grams for the true data were "President Donald Trump," "President Barack Obama" and "Reuters President Donald." For the fake articles, the top tri-grams were "21st Century Wire," "New York Times", and "Donald Trump realdonaldtrump." Figure 2 shows the most common trigrams for both datasets, true and fake articles.

Baseline Models

The baseline models I used are logistic regression, decision tree classifier, random forest, and XGBoost classifier models. I also developed a neural network and fine-tuned transformer model. I chose these specific models to have a wide array of options when choosing my main model and to discern how different model architectures can affect the accuracy of their classification.

For each baseline model, I began by using GridSearchCV for the hyperparameter tuning. A hyperparameter is a setting you can use for your model to slightly tweak the way the model handles the data given to it. GridSearchCV stands for grid search cross-validation and is a way to test multiple hyperparameters to see which one performs the best for your task. After setting up the GridSearchCV, I began tuning hyperparameters.

The hyperparameters I chose to tune for the decision tree were max-depth and max-features because those were the ones that I thought could have the biggest effect on the results of the model. For the Random Forest Classifier, I chose the same hyperparameters, max-depth, and max-features, but also added a n-estimators parameter too. The max-depth determines how

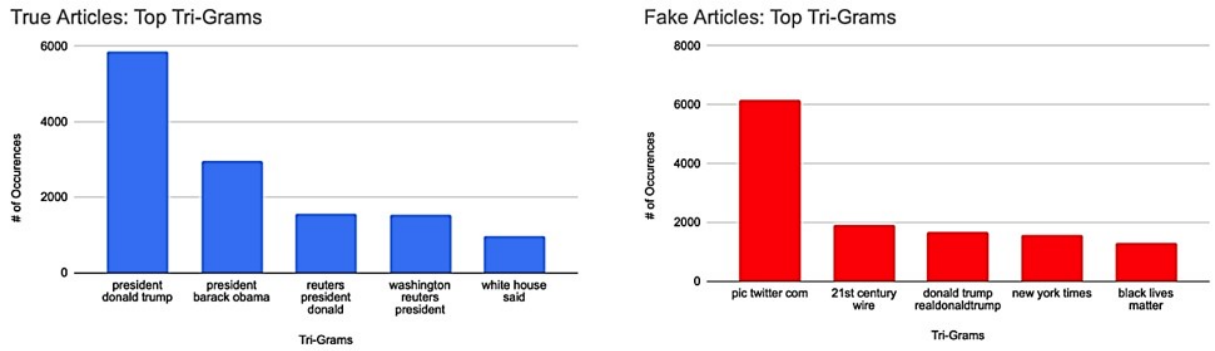

Fig. 2 Most Common Tri-Grams for Datasets.

deep the model will go. The depth depends on where you cut off the splitting of the leaves, and if you leave this alone, the model will keep running until all the leaves are "pure," which means there is almost no uncertainty. The max-features determines how many features the model will consider when splitting the leaves. The n-estimators parameter tells the model how many decision trees to put in the random forest.

I also used the XGBoost model, which is very similar to the random forest model, but there is one key difference. The XGBoost model uses boosting, instead of the bagging that the random forest uses. This means that instead of taking the average of all the individual decision trees to find the accuracy, each model will use the weights changed in the previous model to improve every model down the line.

Neural Network

A neural network takes inputs from a specified number of neurons, runs them through many layers of neurons, all with specified weights, and has an output layer where the value of each neuron determines the output of the model. For the neural network, I had to tokenize the data, meaning giving each word its number which corresponds to it. I also had to pad the articles, which made them all uniform in length, making them readable by the neural network.

I chose to create a neural network with 2 hidden layers. After testing neural networks from 1 to 5 hidden layers, the model with 2 hidden layers proved to be the best. There was a fast run time for the model, and there was no detriment to the accuracy of the model when compared to the models with more hidden layers. This made 2 hidden layers the perfect middle ground for this model. The first hidden layer consists of 128 neurons, and the output layer is just one neuron and uses the sigmoid activation function. A sigmoid function is a way to classify an output as one of two results. There is a gradual slope between the two asymptotes, and based on where the output falls on the graph, it is classified as one of the two results.

I used this model to predict the labels of the data, whether the article is true or false, and then compared it to the actual labels from the original dataset. I chose not to tune the hyperparameters on this neural network model. This is because the performance was already so high for the prior models that tuning the hyperparameters would be a lot of work for very minimal, or even no gains.

Fine-Tuned Transformer

Finally, I chose to use a Transformer to see exactly how accurate a model could become. A Transformer is a large language model that was trained on a lot of data on another computer. Once the model has been trained for the specific task, the part of the model that learned how to "speak" language it was trained in is then exported and can be utilized for other purposes. Transformers offer an open-source, easy option for the development of very accurate machine learning models (Wolf et al. 2020)[6]. I used the "distilbert-base-uncased" version for the optimal balance between precision and efficiency. A visualization of a transformer can be seen in Figure 5. The output of a transformer can be whatever one wants, the only specialty is that the model will use the knowledge of language gained from the dataset, in this case "distilbert-base-uncased".

As with the neural network, the data needed to be tokenized and padded, but in this case, the tokens for the words came from the "distilbert-base-uncased" transformer. Once tokenized and padded, I converted them back into datasets and set the input and outputs for my transformer. The inputs would be these new datasets with the tokenized data, and the output

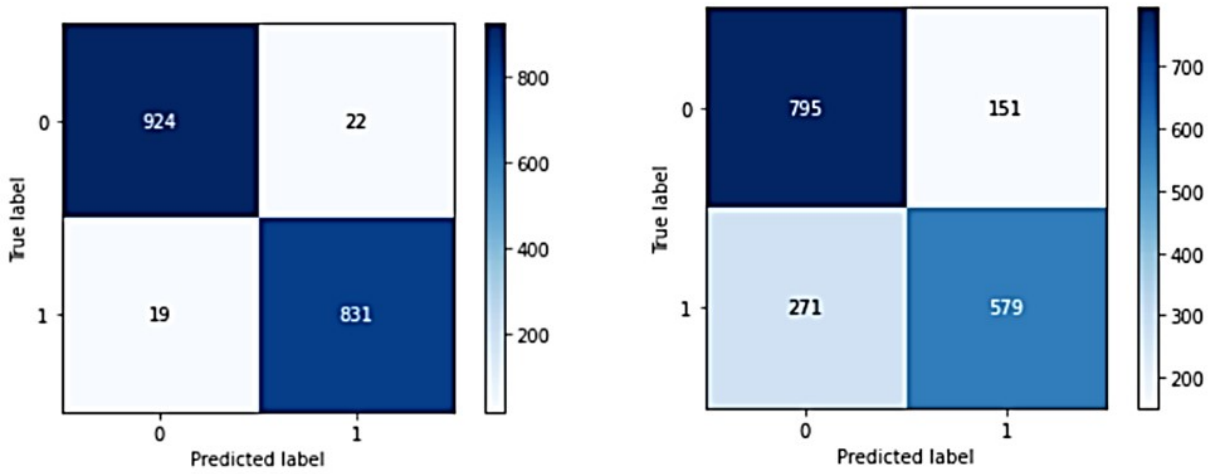

Fig. 3 a) Confusion Matrix for Decision Tree Classifier b) Confusion Matrix for Logistic Classifier

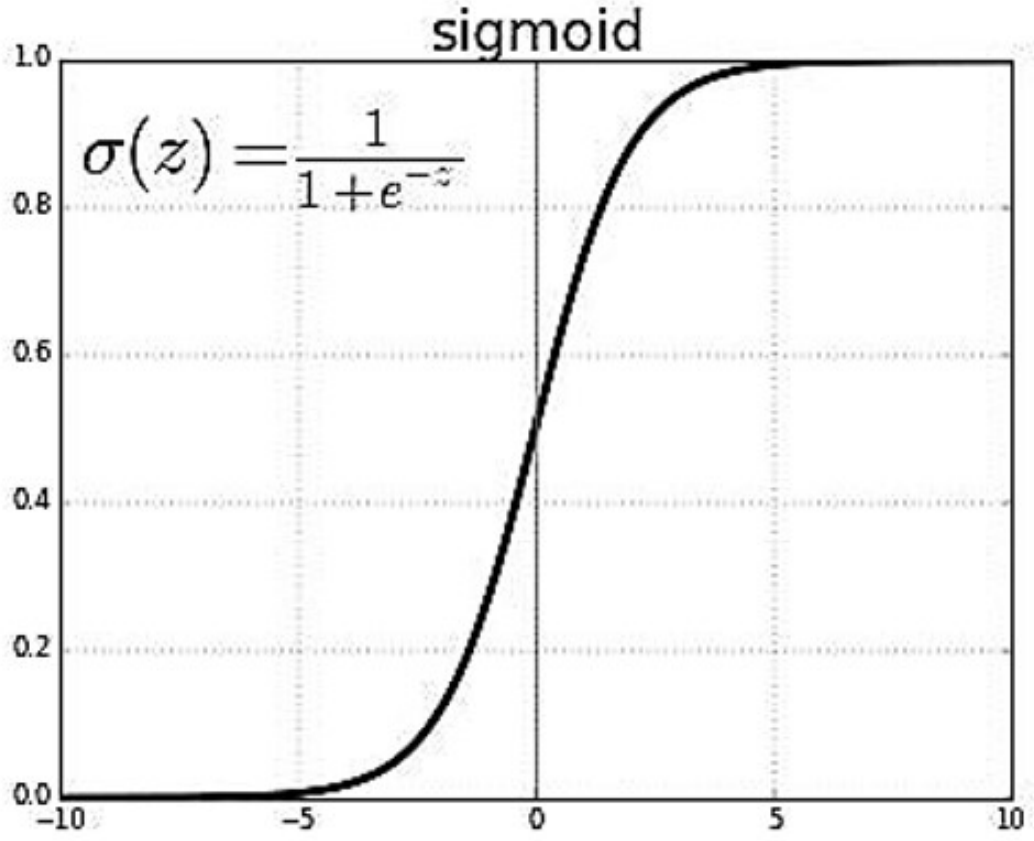

Fig. 4 An image of a sigmoid function (Taken from Suman 2022)[5]

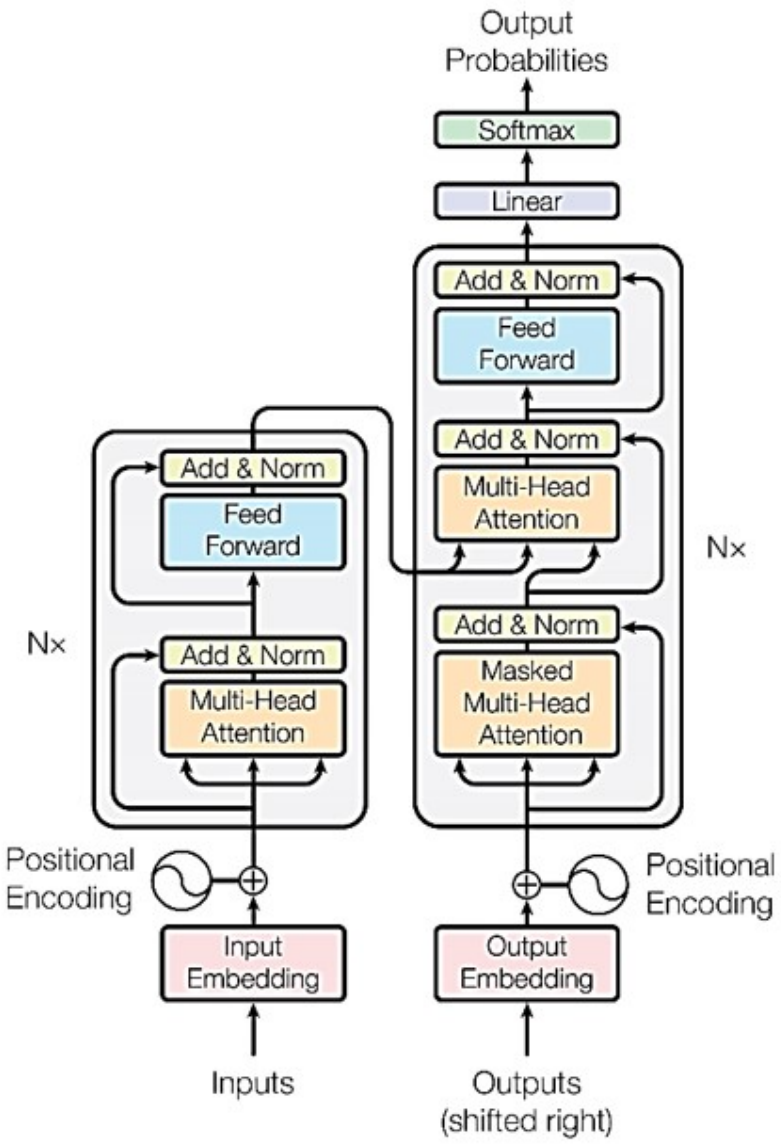

Fig. 5 Visualization of a transformer (Taken from Mavuduru 2020)[7]

would be the labels of veracity for each article.

I used the Huggingface transformer library to do this. The Huggingface library has a collection of pre-trained transformer models, and their library allows you to be able to download and use their pre-trained models. A pre-trained transformer is a transformer that has been trained on a large data set and refined for a specific purpose. I then used these pre-trained transformers to tune them for my purpose.

Conceptually, the way this is done is by cutting off the input and output layers of the transformer and replacing them with your inputs and outputs. A transformer has three parts: an input, an encoder, and an output. The encoder stores all the values and weights for the transformer. These values determine how the model "learned" the language, and using this, with small tweaks, you can practically perform any function. These small tweaks to the model are called fine-tuning. Fine-tuning takes the encoding part of the transformer and changes it slightly until it gives the proper outputs.

Finally, I tuned hyperparameters using the Optuna architecture. Optuna is a tuning software made for transformers and goes about adjusting the model in a similar way to the baseline models. To do this, specify the parameters you want to test out, and a Trainer is made, which iterates through all of them, finally choosing the ones that give the highest accuracy.

For this specific model, the parameters that produced the best results were a learning rate of 3e-05 and four epochs. A learning rate determines how much the model "moves" after every iteration to achieve higher accuracy. An epoch is the number of times you run each trial of your model, with each epoch improving upon the last.

Results

For the logistic classifier, the accuracy of the model was 97%, the recall was 97%, and the precision was also 97%. Accuracy is how many times the model was correct in total. Precision is how good the model is at predicting a specific aspect of the data. Recall tells you how many times the model was able to correctly find these aspects (Dang 2022)[8]. For the decision tree classifier, a max-depth value of 30 and a max-features value of 'auto' turned out to be the best for this model, yielding an accuracy of 80%. With this model, a recall of 68%, and a precision of 79%. The random forest classifier resulted in a much better accuracy score, of 97%. The hyperparameters that performed the best were a max-depth of 30, a max-features of auto, and a n-estimator of 1000. The disparity between the accuracy values shows how much better the Random Forest Classifier is than the Decision Tree Classifier, with an accuracy increase of 17

The neural network had a very big jump in results when compared to the previous models. When running this model, the accuracy was already at 100% after just one epoch, and every epoch after kept that same accuracy level. When looking at the confusion matrix, you can see that the model perfectly predicted the veracity of each piece of news. As with the neural network, the Transformer also got 100% accuracy after just one cycle, but there was a lot less fine-tuning needed for this model. These results show that this Transformer architecture is by far the most effective way to do natural language processing. In this case, it could be used to determine the veracity of a multitude of different articles with near-perfect accuracy. Table 1 offers a tabular view of the best hyperparameters for the models. The models shown in the table reflect all the models with which hyperparameter tuning was done. Table 2 shows all the results of each model type with all the evaluation metrics.

To validate the best model, an area under the receiver operating characteristic test (AUROC) is necessary. This test shows the relation between true positives and false positives from a classifier model. Because the accuracy of this model is so high, the AUROC curve looks slightly strange. In Figure 6, the AUROC curve for the Fine-Tuned Transformer is shown.

The score from an AUROC tells how well the classifier model works. The scores scale from 0 to 1, with 1 being a perfect model. A score of 0.5 means the model is random, or 50% of the classifications are false positives. As seen in Figure 6, the model performs extremely well. The score for the validation data is 99.2% accuracy, and for the training set, it is 100%. This AUROC curve validates the model's efficiency, and also proves that the model is not overfit. If there is a large disparity between the scores for the training and the validation data, then the model will be over fit. This is not seen in this AUROC curve, so it is safe to say the model is not overfit

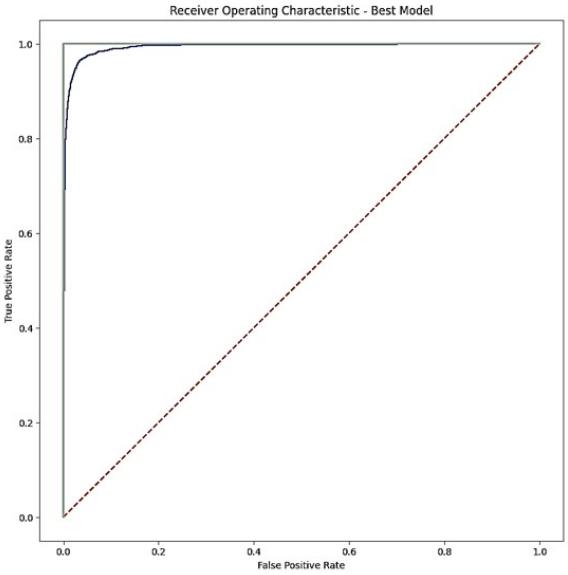

Fig. 6 AUROC curve for Training and Validation Data on Fine-Tuned Transformer

Discussion and Conclusion

Predicting the veracity of news articles with near-perfect accuracy was achieved by using deep learning machine learning models. These results could pave the way for future research into the trends of fake and true news, how websites can detect fake news on their websites, and even teach writers how to write in a more convincing, truthful way. Social media user share posts to feel like members of society who contribute, and with fake news being everywhere, this leads to the exponential spread of the fake news (Li 2022)[9]. With companies like FactCheck and RumorGuard setting up services to help fight the spread of fake news, the solutions to the fake news epidemic can start to be seen. The accuracy and efficiency of these models needs to be improved, but this is a necessary first step in the fight against fake news.

Classifying news articles as true and false using Transformers has shown that a very high, almost perfect accuracy can be achieved. These outcomes illustrate the effectiveness of these deep-learning models and how, when implemented correctly, they can yield fantastic results.

	Learning Rate	Number of Epochs	Max_depth	Max_Features	N-estimators
Decision Tree Classifier	NA	NA	30	'auto'	NA
Random Forest Classifier	NA	NA	30	'auto'	1000
Transformer	3e-05	4	NA	NA	NA

Table 1 Best performing Hyperparameters

	Accuracy	Precision	Recall
Logistic Classifier	97%	97%	97%
Decision Tree Classifier	80%	79%	68%
Random Forest Classifier	97%	98%	98%
XG Boost	99%	98%	99%
Neural Network	98%	98%	99%
Baseline Transformer	100%	100%	100%
Fine-Tuned Transformer	100%	100%	100%

Table 2 Results of All Models: Accuracy, Precision, and Recall

While these results were promising, a limitation in my research could be the type of dataset I used. The data may have been too "easy" for these advanced models, and there might have been blatant patterns in the true and fake news that made them easy to decipher. With more diverse data, the transformer may have had a harder time determining the labels of the data and would have had to change the weights of its neurons in a more obscure, interesting way. While the Transformer produced great results, there is still a lot of room for better, more precise models to be built that can be even more prolific with harder, more obscure data.

After completing the model, I also built an online version of the model that can be used to predict the veracity of articles. You can type in or paste the text into a text box, and at the click of a button, the model will tell you the percentage of certainty that it thinks that the given article is true or false. This online implementation was possible because of the Huggingface online git repository, which allows you to upload your model and other required files and it can run the model on any specified input. This makes the model much more accessible to the public, and more people now can use it. While there are still some problems to fix in the model, such as bias due to the data and simplified models, this is an important step for the future implementation of this model. The current link to the online module is here: "https://huggingface.co/raghavsharma06/testtrainer". The link to the GitHub with all the code is here: https://github.com/RaghavS06/VeracityDeterminer. If these problems can be addressed, then this architecture can be used to weed out fake news from large social media sites like Twitter and Facebook with great accuracy. Building online tools for public use has also shown that this is a possibility, and it is only a matter of time before this solution can be implemented online, broadening the scope of the research, and making this have a large impact all over the globe.

References

1 Y. M. Rocha and G. A. Moura, *Gabriel Alves Desidério,Carlos Henrique de Oliveira,Francisco Dantas Lourenço,Larissa Deadame de Figueiredo Nicolete The impact of fake news on social media and its influence on health during the COVID-19 pandemic: a systematic review*, J Public Health (Berl.

2 S. Rimer, *Fake news influences real news*, `https://www.bu.edu/bostonia/2017/fake-news-influences-real-news/`.

3 P. Dizikes, *MIT News*.

4 C. Bisaillon, *Fake and real news dataset*, `https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset`.

5 A. Suman, *Medium.com*.

6 T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest and A. Rush, Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45,.

7 A. Mavuduru, *Medium.com. December*, **1**, year.

8 T. Dang, *Guide to accuracy, precision, and recall*, `https://www.mage.ai/blog/definitive-guide-to-accuracy-precision-recall-for-product-deve`

9 J. Li and X. Chang, *Inf Syst Front*, **11**, 1–15.