Natural Language Processing - Fake News Detection

mbtj48

1 Introduction

Within the developing world of modern AI, understanding large corpora of text is paramount. Therefore, modern deep learning techniques can help not only improve the scale and speed of processing such data but also the quality of the extracted semantic meaning of the data. As a consequence, text-based generative models such as GPT-3 [2] and DALLE-2 [8] are becoming prevalent within social media. The generative models in question can be fine tuned to produce nearly human quality custom outputs after a few short hours of training and as such can be used for malicious purposes such as spreading fake news or self-generating spam mail.

2 Problem Definition

We seek to classify the stances of articles and their headlines using the Fake News dataset [7]. This dataset contains 4 classes: unrelated, disagree, discuss and agree, relating to the link between the article and it's headline. Within popular media, this is called "clickbait" and describes how badly the headline is related to the content to get someone to read it's contents.

3 Proposed Solutions

We utilise a hierarchical approach to classify the documents. This means we utilise an initial model to determine whether the headline is related or unrelated to the article body, and then another more dense model to classify the related predictions into agree, disagree and discuss. As a baseline, we first develop 2 shallow models; one with TF-IDF embeddings and the second with transformer embeddings, and then develop 2 deep models using an LSTM to improve the performance. For initial testing, we use a validation set which is 20% of the training dataset, however we later realised this gave a poor representation of model quality as there was some cross over between training and validation. Further data analysis would need to be performed to ensure there are no article or headline overlaps for the validation set.

3.1 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF [9] is a feature extraction method which calculates a vector that represents the document's features based on the frequency of words. The naïve technique, is performed with the following formula:

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\Sigma_{t'\in d} f_{t',d}}, \tag{1}$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}, \tag{2}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D). \tag{3}$$

Here, $\mathrm{tf}(t,d)$ denotes the term frequency of a term t in the document d, $\mathrm{idf}(t,D)$ denotes the quality of a term across all documents by taking into account the number of documents which contain that term across every document. For a high vector in tf-idf, the frequency in the document needs to be

Table 1: Validation (left) and test (right) set metrics for TF-IDF features with XGBoost

	Precision	Recall	f1-score	Support	Precision	Recall	f1-score	Support
Unrelated	0.7265	0.9832	0.8356	7269	0.7244	0.9811	0.8334	18349
Related	0.2229	0.0128	0.0243	2726	0.3837	0.0306	0.0566	7064
Accuracy			0.7186	9995			0.7169	25413
Macro Average	0.4747	0.4980	0.4299	9995	0.5540	0.5058	0.4450	25413
Weighted Average	0.5891	0.7186	0.6143	9995	0.6297	0.7169	0.6175	25413

Table 2: Validation (left) and test (right) set metrics for BERT features with XGBoost

	Precision	Recall	f1-score	Support	Precision	Recall	f1-score	Support
Unrelated	0.9925	0.9948	0.9936	7269	0.9864	0.9867	0.9866	18349
Related	0.9860	0.9798	0.9829	2726	0.9654	0.9648	0.9651	7064
Accuracy			0.9907	9995			0.9806	25413
Macro Average	0.9892	0.9873	0.9882	9995	0.9759	0.9757	0.9758	25413
Weighted Average	0.9907	0.9907	0.9907	9995	0.9806	0.9806	0.9806	25413

high, while the overall frequency of the word in the whole corpus remains low (thus omitting key frequent words such as “the”). Therefore, tf-idf works well for tasks where positional embeddings are not important such as topic modelling and keyword extraction; however, it is therefore, by itself, unable to encode the relationship between the headline of an article and the body. For our implementation, we use the sklearn TfidfVectorizer to fit the training documents to 2048 features. In order to improve TF-IDF’s features, we preprocess the text to lowercase, remove non-ascii characters, select up to $n = 2$ n-grams and remove terms with an overall frequency of 5 or less from the vocabulary.

3.2 Transformer Embedding

Transformers have become the state-of-the-art implementation for text-based language modelling. They seek to learn the importance (attention [10]) of the words surrounding a hidden word in a sequence. We utilised BERT transformer embeddings [4] to encode the headline and article body, which embeds document information in the initial classification token. BERT takes the encoder of the typical sequence-to-sequence transformer generative model and is trained on predicting a masked word, while also predicting whether a sentence follows another sentence. Therefore the model inherently learns the relationship between sections of text. One of the primary disadvantages of transformers is their long training times, however utilising a pre-trained model partially mitigates this.

3.3 Gradient Boosting

We incorporate the popular XGBoost [3], gradient boosting classifier which optimises a random forest with a gradient descent algorithm on a predefined loss function. This predicts the class based on the tfidf vector and therefore, it performs proportional to the imbalance of the class distribution when classifying whether the body of text is related or unrelated to the headline as tfidf doesn’t encode positional information.

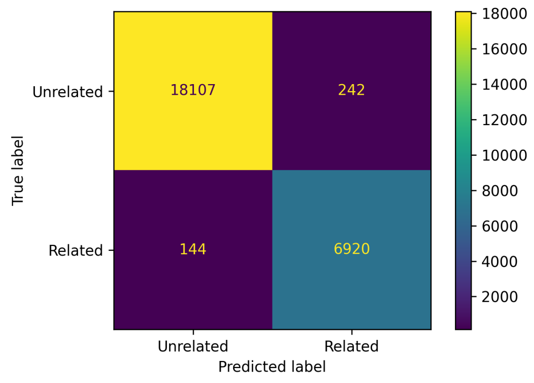
For the transformer, the BERT classification token’s embedding is a 768-sized vector that is passed into the gradient boosting model and successfully encoding a strong relationship between the headline. This experiment is seen in Tabs. 1 and 2.

Table 3: Test set metrics for TF-IDF LSTM model

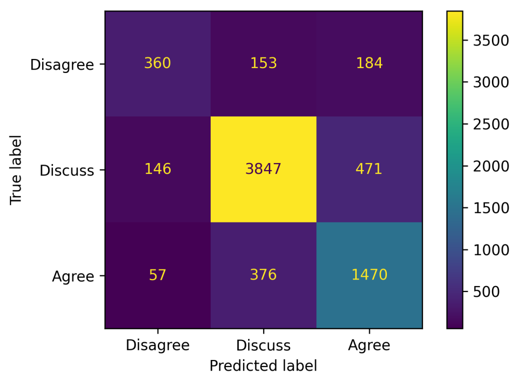
	Precision	Recall	f1-score	Support
Unrelated	0.7612	0.5859	0.6622	18349
Related	0.3270	0.5227	0.4023	7064
Accuracy			0.5683	25413
Macro Average	0.5441	0.5543	0.5322	25413
Weighted Average	0.6405	0.5683	0.5899	25413

3.4 Long-Short Term Memory

For consistency, we utilise the same architecture for both TF-IDF and BERT embeddings. This architecture encodes the tokenised input using the relevant embedding, then a 2-layer bi-directional LSTM with post dropout probability of 0.5. Lastly, the model has a classifier linear layer with a sigmoid activation function to output a probability of unrelated & related. Furthermore, we use the cross-entropy loss, commonly used for text classification [5], and the AdamW optimiser throughout. At this stage, tuning the optimiser hyperparameters was not required to yield strong results.



(a) Confusion Matrix for LSTM model with BERT embeddings



(b) Confusion Matrix for final RoBERTa model

Figure 1: Confusion matrices for each section of the hierarchical model

Due to the large data imbalance in the dataset, we immediately incorporated a weighted random sampler for training. This ensures each class is equally represented when training to reduce overfitting on over-represented classes.

Interestingly, the features for TF-IDF are mutually exclusive, so we don't expect a time series model to perform well; the hope was that 2 LSTM layers could learn the combinations of features that correspond to a given class. However the model overfitted and yielded a high training accuracy, but a low testing accuracy; indicating that the difference in vocabulary between train and test hindered the model. A more accurate TF-IDF model would tokenise the headline and article body separately and then combine them using several linear layers, thus encoding the relationship between the headline and article vectors.

3.5 RoBERTa

Lastly, we finetune RoBERTa base sequence-2-sequence classifier [6] in two stages to predict agree, disagree and discuss. The first stage, freezes all roberta layers, thus optimising the classification output of the original RoBERTa model. This allows the model to get a "warm" start for this problem set.

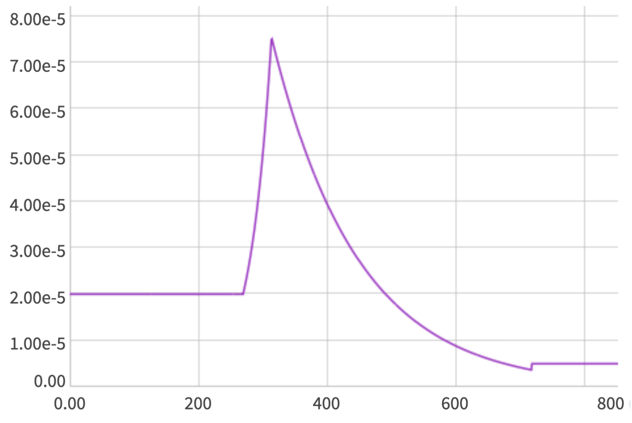
Table 4: Test set metrics for BERT Embedding LSTM model

	Precision	Recall	f1-score	Support
Unrelated	0.9921	0.9868	0.9895	18349
Related	0.9662	0.9796	0.9729	7064
Accuracy			0.9848	25413
Macro Average	0.9792	0.9832	0.9812	25413
Weighted Average	0.9849	0.9848	0.9848	25413

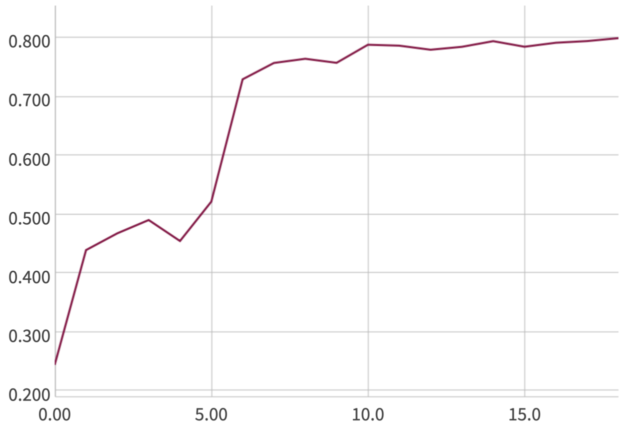
Table 5: Test set metrics for RoBERTa classifier at fine tune stage 1 (left) and fine tune stage 2 (right)

	Precision	Recall	f1-score	Support	Precision	Recall	f1-score	Support
Disagree	0.2215	0.3659	0.2760	697	0.6394	0.5165	0.5714	697
Discuss	0.8246	0.4823	0.6086	4464	0.8791	0.8618	0.8704	4464
Agree	0.3886	0.6742	0.4930	1903	0.6918	0.7725	0.7299	1903
Accuracy			0.5225	7064			0.8037	7064
Macro Average	0.4782	0.5075	0.4592	7064	0.7368	0.7169	0.7239	7064
Weighted Average	0.6476	0.5225	0.5446	7064	0.8050	0.8037	0.8030	7064

Secondly, the core RoBERTa model is unfrozen, leaving the embeddings frozen, consequently allowing the model to learn where to attend for this dataset rather than updating the learned representation of the words which the embedding provides. The default architecture and optimiser parameters would not converge, so therefore we utilise a similar training from [6] and provide a warm-up learning rate and then a slow exponential decrease as seen in Fig. 2a. However this resulted in high overfitting, so we therefore applied a network weight decay of 0.08 and increased the classifier dropout from 0.1 to 0.3. We also experimented with attention and hidden layer dropouts, but did not find these to improve the results. Eventually, the architecture produced the test set accuracy through training seen in Fig. 2b.



(a) Learning rate function



(b) RoBERTa validation accuracy solely on test set

Figure 2: Changes of learning rate and validation accuracy over training

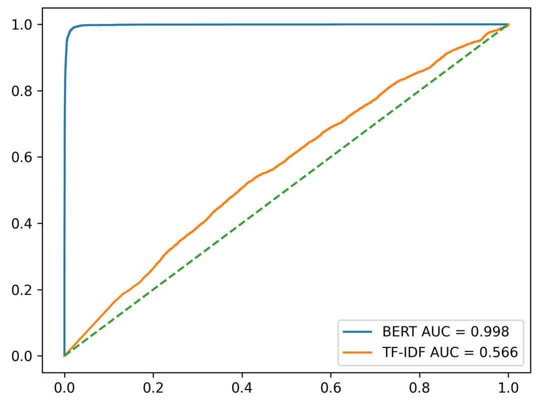
4 Evaluation

When evaluating the full end-to-end model on the test dataset, the over represented classes: unrelated & discuss perform well. However, the other classes generally have a lower performance; indicating that the model overfitted to the training dataset. Furthermore, for the task of fake news detection the

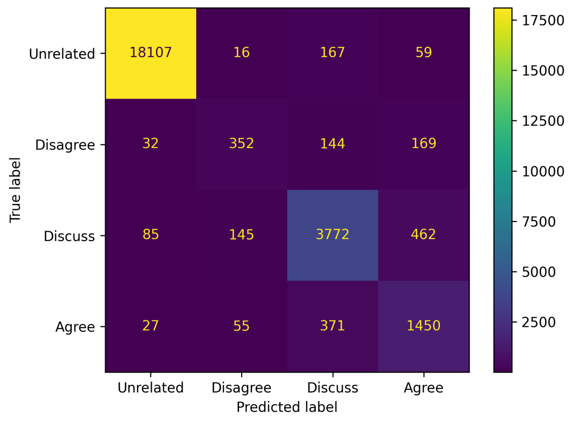
Table 6: Test set metrics for overall hierarchical classifier

	Precision	Recall	f1-score	Support
Unrelated	0.9921	0.9868	0.9895	18349
Disagree	0.6197	0.4351	0.5113	809
Discuss	0.8469	0.8614	0.8541	4379
Agree	0.6776	0.7729	0.7221	1876
Accuracy			0.9318	25413
Macro Average	0.7841	0.7641	0.7692	25413
Weighted Average	0.9320	0.9318	0.9312	25413

most important classes are unrelated and disagree. Therefore, it is best to optimise the accuracy of disagree and unrelated articles: with our model partially succeeding. Unfortunately, due to the lack of data entries, the model struggles to classify articles as disagree, as shown inFig. 3b and highlights the inability to determine the precise relationship with the body and headline. Interestingly, the rolling accuracy in the validation set always dips towards the end of the epoch, further indicating that this region is likely the disagree class that it cannot understand. Future work could explore focusing on helping the model understand the difference between agree and disagree with a custom loss function. For example, the probability of agree should be $1 - \mathbb{P}[\text{disagree}]$.



(a) Receiver operating characteristics curve for related and unrelated deep models



(b) Confusion Matrix indicating distribution of actual vs real predictions.

Figure 3: Test set evaluation

5 Ethical Considerations

Through our end-to-end model, we utilise pre-trained embedding architectures, therefore resulting in a high likelihood of corrupted and biased models. Due to the typical nature of training, these models see many instances of gender bias, potential racist embeddings and general discrimination. Sadly, this is due to the heavy skew in society resulting in a data imbalance of specific instances, such as the vector representation of male jobs being vastly different to female [1]. The corresponding biases in text-based generative models could lead to an influx in generated social media articles with embedded biases from the learned text models. As a result, maintaining the bias in our society and culture.

6 Conclusion

To conclude the hierarchical approach with transformer enhancements works well for the fake news detection task; resulting in strong accuracy for the unrelated task while making the rest of the dataset more balanced for further training. In the future, it would be beneficial to explore alternative sequence-to-sequence classification model such as "RoBERTa-large-mnli". The MNLI task is incredibly similar to the fake news detection task and seeks to predict 3 classes for text data pairs: contradiction, neutral and entailment; therefore linking well to this problem. While this model should perform better for this problem, it is not ideal for practical use due to the vast quantity of parameters (355M Vs. RoBERTa base's 125M) that are used to make a prediction.

References

[1] Tolga Bolukbasi et al. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings". In: *Advances in neural information processing systems* 29 (2016).

[2] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[3] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* 2016, pp. 785–794.

[4] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[5] Robert Dzisevič and Dmitrij Šešok. "Text Classification using Different Feature Extraction Approaches". In: *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream).* 2019, pp. 1–4. DOI: 10.1109/eStream.2019.8732167.

[6] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[7] D Pomerleau and D Rao. "Fake News Challenge (2017)". In: *URL http://www. fakenewschallenge. org* ().

[8] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents". In: *arXiv preprint arXiv:2204.06125* (2022).

[9] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning.* Vol. 242. 1. Citeseer. 2003, pp. 29–48.

[10] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).