Busting Fake News in a Digital World - Project Final Report

Authors: Chua Cheng Ling | Colin Ng Chenyu | Tan Wei Zhong Kevin | Nguyen Thanh Duc | Tran Quang Thanh | Le Hong Long
Email: e0323077@u.nus.edu | e0325455@u.nus.edu | e0309723@u.nus.edu
e0313523@u.nus.edu | e0313566@u.nus.edu | e0313558@u.nus.edu

Abstract

The spread of misinformation has severe detrimental consequences on society, resulting in an erosion of faith in traditional institutions in our society. As such, there is a great need to be able to detect fake news accurately so that consumers would be able to discern potential misinformation. In this paper, we implement and review various machine learning models such as Naive Bayes Classifier, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Bidirectional Encoder Representation from Transformer (BERT) models to predict fake news. We will then discuss and evaluate these models. Through our findings, we propose that BERT is the best model for the detection of fake news, achieving a 79% accuracy .

1 Introduction

With the advent of technology over the last decade, new media can be easily accessed and spread wide. While this advancement has made it convenient for users to follow their preferred news sites at the click of a button, it has also blurred the line between reliable media and maliciously fabricated news, widely known nowadays as 'Fake News' [1]. Social media platforms are a great source of the circulation and exchange of information today. We are now able to receive information instantly across large distances about events happening around the world. However, there are bad actors that use these platforms to spread misinformation by taking advantage of the virality of fake news. A possible reason why the spread of misinformation is so quick is due to the high level of trust that we have in the people we choose to follow on social media [2]. As we usually follow those that we respect on social media, we are more inclined to believe in the information they share and may even unknowingly spread misinformation as well.

A Straits Times news article dated in 2018 reports how even though 4 in 5 Singaporeans are confident in spotting fake news, over 90% mistakenly identified at least one out of five fake headlines as real [3]. This highlights how challenging it is to spot fake news. Furthermore, the current pace of life in the 21st Century has created 'infostorms' that cloud our senses, causing people to easily believe something without checking if the information is accurate [4]. This spread of misinformation can be swift and cause major damage to public trust in our institutions [5].

The intention of spreading fake news is often to mislead readers, and as such, causing damage to the reputation of a person or an entity [6]. However, the impact of such misinformation can also detrimentally cause a sense of fear, change in political mindset and may even induce racism in a person [7]. Having a robust model that can identify fake news can go a long way in aiding consumers discerning the trustability of an article. This will hopefully reduce the spread of misinformation. Hence, to prevent such a situation from happening, our group aims to develop and evaluate the best model that can successfully classify a news article based on text data, and hopefully bring about some clarity in the cloud of information of this age.

2 Related Work

2.1 Linguistic Approaches

Many linguistic approaches to fake news detection have been used to identify deception in recent years. Bing Liu et al. 2012 analysed fake reviews on amazon based on sentiment analysis, lexical, content similarity, style similarity and semantic inconsistency to identify fake reviews [8]. Bondielli & Marcelloni, 2019 highlighted the importance of having both real and fake news in our dataset. In addition, they detailed how Natural Language Processing tools such as NLTK can be used to extract important features of the text such as tokens and parts-of-speech [9].

However, methods based just on word analysis is not enough to identify deception. Feng et al. 2012 proposed syntactic stylometry for deception detection and a focus on deeper language structures such as the syntax tree and attempt to classify based on it [10]. Given the great variety of writing styles in various news articles, it is prudent that we are able to correctly interpret the text to correctly classify news articles.

2.2 Neural Networks Approach

The use of convolution neural networks (CNN) was primarily meant for image recognition, however Kim, 2014 was able to demonstrate that CNNs are able to still perform well for the purposes of text classification [11]. Furthermore, Yang et al, 2018 were able to successfully utilise a modified version of CNN involving combinations of text and image information for fake news recognition with an impressive precision and recall score of over 0.9 [12]. In the case of our project, we did a scaled down version by only considering text information. In addition, Devlin et al, 2019 acknowledged how pre-training is essential for language processing and explored how to use BERT in doing so [13]. This proved useful for our project which utilised BERT as one of the models for text analysis. Lakew et al. 2018 compared Transformers to RNNs and concluded that Transformers outperform RNNs for language processing [14] and Vaswani et al.2017 showed that Transformers can be trained significantly faster than CNN or RNN [15]. Even though their work is related to translation tasks, we believe that this will benefit us in understanding the effectiveness of Transformers over CNN and RNN in classification tasks.

3 Methodology

The relevant code for exploratory data analysis and model implementation can be found at https://github.com/chenggzzxcc/CS3244_Group10

3.1 Dataset

The dataset used for this project had around 4731 articles of data which was obtained from Kaggle[1]. The dataset can be considered noisy and require cleaning before they are ready to be used. Our final product of the dataset includes a total of 2 columns which are text and label (where 0 refers to real news and 1 refers to fake news). We have a total of 1857 entries of fake news and 2874 entries of real news. The imbalance of data may pose some challenges in our approach later.

[1] https://www.kaggle.com/c/fakenewskdd2020/

We will split our dataset into 2 parts: training and validation, then we will evaluate the accuracy on the validation set, and compare with our model submission on Kaggle. This accuracy is the percentage of correctly classified fake news.

3.2 Data Processing

In the case of text processing, the text input is first tokenized, where individual sentences are split into individual words. Each word is then converted into a vector of numerical inputs to be processed by a dictionary. We use GloVe embeddings to create word tokens that can be contextualised for our CNN and RNN models [16].

3.3 Text Analysis

Although many perjurers have control to an extent of what they write, some of these implicit and explicit differences may become apparent due to their state of mind and may be observed from their style of writing. We aim to explore the explicit differences between language styles between fake and real news in this section.

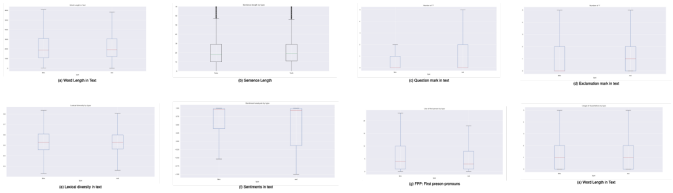

Figure 1: Analysis on news text

3.3.1 Insights Generated from Text Analysis

Figure 1 shows the boxplots of 8 hand-picked features that we had compared between real and fake news. From the individual plots, there are some minor differences that we can derive. For instance, for the number of question marks in each text, though there seem to be no difference in the median, the boxplot shows that the number of question marks can go as high as 4 in real news - a scenario that has never happened for fake news. One possible explanation may be due to the nature of how fake news is written, where they aim to provide false information/statistics instead of asking rhetorical questions in their text.

Notably, we observed that the number of exclamation marks in each text differs significantly between real and fake news, as evident in Figure 1(d) where the median number of exclamation marks used in real text is 2 while it is 0 for fake news. From this, we can make a hypothesis that real news are more inclined to use exclamation marks. Overall, though the other features do not show distinct differences through our plots, we believe they are still possible avenues to explore in our model.

3.4 Models

3.4.1 Naive Bayes Mode

In the training dataset, our naive bayes model counts the number of times a particular word appears in the text, given that the news is fake. These numbers are converted to probabilities and stored in a dictionary. The same approach is done with data where the news is real. This leads to two dictionaries, one each for real and fake that consists of words (that are stored as keys) and their corresponding value (which is the probability of the word appearing).

To test an unlabelled news article, the model calculates the probability of it being either real or fake using the two dictionaries, and classify it based on the probability that is larger.

3.4.2 CNN Model, CNN Variants and RNN Model.

3.4.2.1 CNN Model

CNN models were primarily developed for image recognition [11] but can be used for text processing as well in areas of text classification. We trained a vanilla CNN model with as well as other variants using our textual explicit features obtained from our analysis of our text in (3.3).

We utilise two sets of textual features in our CNN model, textual explicit features and textual latent features. Textual explicit features are derived from the data analysis in (3.3) for a total of 8 textual explicit features. The statistics of the text can then be organised as a vector and then transformed by a fully connected layer to form these text explicit features. With the convolutional approach, neural networks produce local features around each word of the adjacent word then combine them using the max pooling layer to create a fixed-sized word-level embedding. Therefore, we employ CNN to model textual latent features.

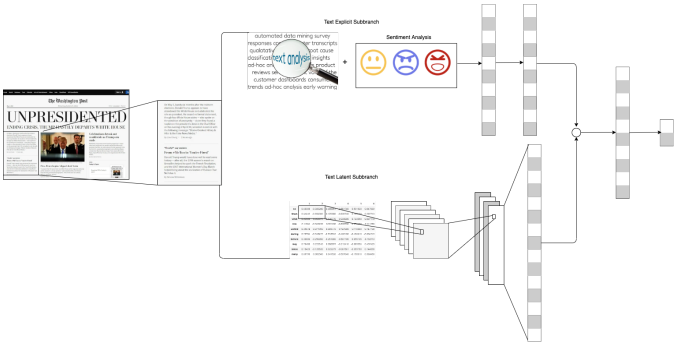

Figure 2. The architecture of the CNN model. The image here is illustrative and certain areas are not drawn for brevity. The details of the structure are found in figure 10 in Appendix A .

We use 1D convolutional for text data and filters are replicated across the entire field and share the same parameterisation forming a feature map [11]. A max-pooling layer is connected to the convolutional layer. We then apply maximum activation over the filters to the output of the max-pooling layer. Max-pooling enables position invariance over larger local regions and down-samples the input.

To avoid the issue of vanishing gradients, we chose to use the ReLU activation to avoid this problem. The gradient computation is simple, and the computation steps fast which shortens training time [17]. We employ dropout to prevent overfitting, which greatly improves the generalisation ability of our model. [18] We also utilise early stopping to avoid overfitting. [19]

We used Binary Cross Entropy Function as our loss function of choice under the inference framework of maximum likelihood [20]. We also used Adam optimizer to optimize our loss function [21]. The back-propagation algorithm was utilised to compute the gradients of the network structure. With fine tuned parameters, the loss converges to a good local minimum.

3.4.2.3 RNN Model

RNNs are a class of deep learning models meant to work well with natural language processing [22]. In this project, we have implemented two RNN models: Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) and we settled with GRU in the end.

One key architecture that was used in the implementation of our RNN model was the use of GRU. GRU is an improved version of RNN, where vanishing gradient is a problem. Instead of using 3 gates like LSTM, GRU only uses 2 gates: reset and update. Update gate behaves similarly to forget and input gates of LSTM, where it determines how much of past information can be passed on to the future. Reset gate also decides on the amount of old information to drop out of the memory [23][24]. In the end, GRU has better performance due to fewer parameters being used

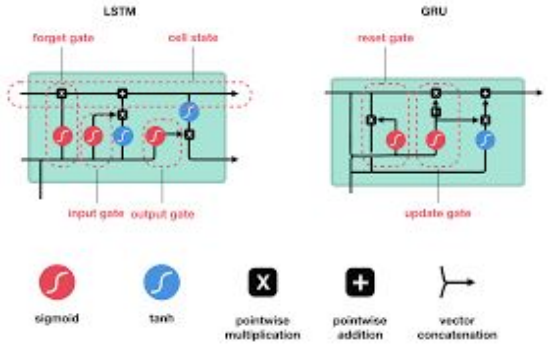

Figure 3: LSTM and GRU graph. Diagram credits: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

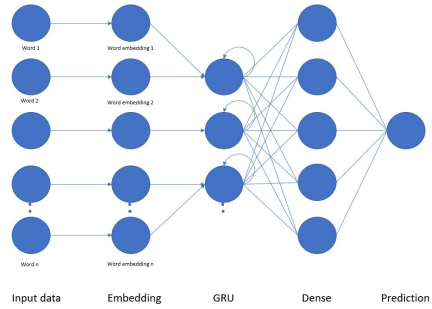

Figure 4: An illustration of the workings of the RNN Model

An embedding layer is being used to convert words into vectors followed by a stacked Bidirectional GRU, which involves duplicating the first recurrent layer in the network. The input sequence is then provided as input to the first layer and it provides a reversed copy of the input sequence to the second, with 16 output features. After which, there will be 2 Dense layers with output dimensionality of 256 and 1. The activation functions used are swish and sigmoid respectively before coming up with the final labels' prediction.

The Sigmoid function is a good choice for evaluating the final labels as it ranges from 0 to 1, which is similar to probabilities [25]. However, it could make the training 'stuck' in the case when strong negativity inputs could make weights in backpropagation slowly update [26]. On the other hand, Swish is a smooth function (there are no sudden changes of motion or vertex) and it is non-monotonic [27]. In our dataset, Swish shows a better performance for activation function after the stacked GRU or LSTM layers compared to other alternatives such as Softplus or Sigmoid where they have positive derivatives for all the points.

3.4.4 BERT Model

BERT is an ensemble of smaller architectures referred to as Transformers [28]. However, unlike Transformers which have an encoder and a decoder, BERT is an Encoder-only model. The workings of the BERT model is illustrated in Figure 5 below.

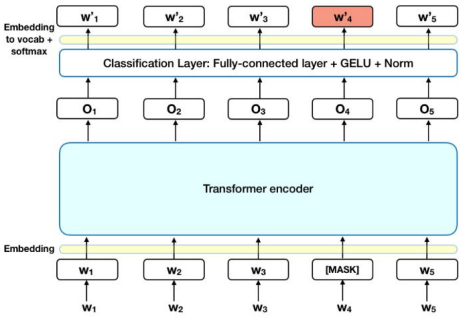

Figure 5: BERT overall architecture. Diagram credits: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

3.4.4.1 Embeddings

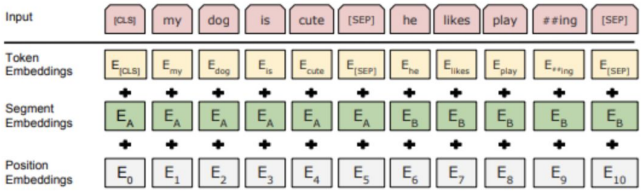

Figure 6: The embeddings of BERT. Diagram credits: https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

BERT uses a combination of 3 embeddings: word embeddings, position embeddings and token type embeddings. The final embeddings are then the sum of all 3 passed through a Layer Normalization layer to reduce the effects of bad training inputs [29].

The formula for embeddings is thus:

$$E_{final} = LayerNormalization(E_{word} + E_{position} + E_{type})$$

3.4.4.1.1 Token Embeddings

To encode the input, each word in the input is matched with either one or multiple tokens in a prebuilt vocabulary. Each matched token will then be converted to a corresponding embedding vector of size 768 which is trainable. For the single token case, the procedure is just an exact match. For the multiple case, each out-of-vocabulary word in the input is broken down into smaller tokens, with a "##" prefix if the token is the latter part of a word. To illustrate, the made up word "bewitchedman" can be broken down into ("bewitched", "##man"). This technique allows BERT to generate word embeddings for out-of-vocabulary words. We utilized the existing vocabulary as provided by Hugging Face's transformers library.

3.4.4.1.2 Position Embeddings

In our BERT, the absolute token position values are then mapped to a corresponding embedding vector of size 768 to preserve the embedding size. However, this choice of embeddings also introduces a new problem which is the length of the input as the position embeddings input values cannot be arbitrarily large. This issue, as seen later on, is resolved by limiting the size of input and output for BERT to 512 tokens.

3.4.4.1.3 Segment Embeddings

In the original BERT, the model was also used for Question Answering tasks. To differentiate between the questions and the input contexts, this embeddings layer was added to BERT. However, our dataset is for Fake News Detection and the input is only the article, we only use the same segment embeddings as used for the context part of the Question Answering BERT.

3.4.4.2 BERT Encoder

The BERT Encoder block implements the Transformer Encoder architecture. It consists of 12 Transformer layers with 12 attention heads each. Each BERT Encoder layer consists of a Self-Attention layer and a Feed Forward layer [13].

3.4.4.2.1 Self-Attention

The Self-Attention layer attempts to find the correlations score between the word features by constructing 3 projection vectors Q (query), K (key) and V (value) from the product of the input with 3 corresponding 2D matrices W_Q, W_K, and W_V. The output for each Self-Attention head is calculated as:

$$A_j = \sum_j softmax(\frac{Q_i \times K_j^T}{\sqrt{d}}) \times V_j$$

where d is chosen according to the size of the Self-Attention layer to normalize the output value [30].

3.4.4.2.2 Multi-headed Self-Attention

BERT employs a multi-headed self-attention scheme to allow different self-attention heads to find out different relation features from the input and also to exploit the parallel computing capabilities of GPUs[15] This means there are multiple self-attention heads, each with its own set of projection matrices. The chosen number of heads is 6. The output from each head follows the equation above. All the outputs are then concatenated into a single attention tensor. However, to keep the model output and dimensions manageable, the concatenated features are then passed through a linear projection layer to transform back to the dimension of a single self-attention unit output.

3.4.4.2.3 Residual Connection

Before feeding the output of self-attention to the feed forward layer, a residual connection mechanism is employed [31]. The idea behind this is to allow gradients to flow better inside the network, both in the forward and backward pass. This is a good addition to counter the nonlinear Gaussian Error Linear Unit (GELU) activation function used in BERT. The residual connection then just means to add the output with the input of the layer. In BERT, this sum is then passed through a Layer Normalization layer to counter internal covariate shift which means the shift in the distribution of input to layers [32]. When this shift happens, the training process slows down as the affected layers have to learn to adapt to the new distribution. The formula for this technique in BERT is then:

$$Z_L^{'} = LayerNormalization(X + Z)$$

where X and Z are the input and output to layer L, respectively. This technique is applied on the inputs and outputs of self-attention and feed forward layers.

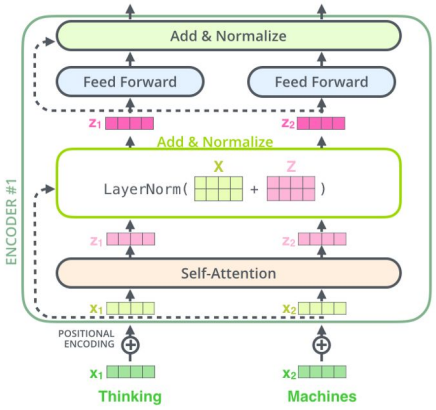

Figure 7: BERT residual connections. Diagram credits: https://towardsdatascience.com/breaking-bert-down-430461f60efb

3.4.4.2.4 Feed Forward

The output of self-attention, after residual connection, is passed through a feed forward layer with GELU activation which has been found to perform better than more traditional nonlinear activations such as ReLU while keeping the computation complexity reasonable [33]. As mentioned before, residual connection from the previous layer is also used to improve the robustness [33]. The output from a single BERT Encoder is then sent through 11 other stacked BERT Encoder layers.

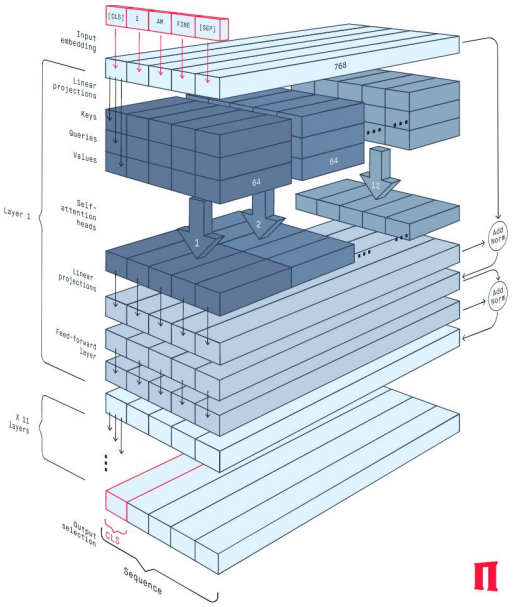

Figure 8: BERT Encoder. Diagram credits: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/bert-encoder

3.4.4.3 Fully Connected

After the Encoder block, the output features are passed through a fully connected layer to extract the relevant features and convert back to the size of the vocabulary. This allows BERT to predict the tokens in the vocabulary and deal with language modelling tasks.

3.4.4.3.1 Sequence Output

For each token in the input, the original BERT outputs a corresponding token classification at that position. This output has been used in tasks such as masked word prediction.

3.4.4.3.2 Pooled Output

Different from previous models, BERT also introduces a summarization output feature. This is achieved by adding a special token "[CLS]" to the beginning of the input. The sequence output corresponding to this token is then passed through a Linear layer with Tanh activation to get the pooled representation of the input. This representation is useful for tasks which require constant output shape such as classification.

The formula for sequence and pooled output is then:

$$Z_{sequence,i} = W_{FC}(H_{Encoder,i})$$

$$Z_{pooled} = \tanh(W_{pool}(W_{FC}(H_{Encoder,[CLS]})))$$

where H is the output representation of the BERT Encoder block

3.4.4.4 GELU and Layer Normalization

3.4.4.4.1 GELU

The formula for GELU is:

$$GELU(x) = xP(X \le x) = x\Phi(x) = 0.5x(1 + erf(\frac{x}{\sqrt{2}}))$$

which approximates to:

$$0.5x(1 + \tanh(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)))$$

3.4.4.4.2 Layer Normalization

Each sample is normalized according to the following formula:

$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

where μ_i is the mean of the batch of samples and σ_i^2 is the variance of the batch of samples.

Then, the final normalized value is calculated as:

$$y_i = \gamma \hat{x}_i + \beta \equiv \mathrm{LN}_{\gamma,\beta}(x_i)$$

3.4.4.5 Our BERT adaptation

In our implementation, we added a Fully Connected layer on top of the pooled output from BERT with 2 output classes - Fake and Real. Also, due to the length limit 512 of BERT, we employed a hard-voting strategy on the classification results of overlapping news segments which means we decide the final classification based on the larger number of votes for a specific class.

4. Experiments and Evaluation

4.1 Dataset

4.1.1 Quantity and Quality of Dataset

As mentioned previously, our dataset obtained from Kaggle has a total of 1857 entries of fake news and 2874 entries of real news, giving us sufficient data points to train our model. Furthermore, the data (i.e. news article) had a broad spectrum of topics such as politics, sports, entertainment, etc. that were evenly spread as compared to many other datasets that we found which only focused heavily on one topic. As such, our models would be able to learn the characteristics of real and fake news that covers a greater variety of topics and hence be able to reliably classify a news article regardless of its topic.

To train BERT, we splitted the news into segments of lengths 510, 254, and 62 with overlapping ratios of 0.2 and 0.4 during training. However, during evaluation and testing, only 510 length segments with 0.2 overlapping ratio was used. Shorter segments were padded according to the maximum length of the batch.

4.1.2 Test Dataset

Our test dataset was slightly unique as it consisted of only the news article itself without any labels (of whether the news article is classified as real or fake) provided. As such, we could only obtain our accuracy by submitting to Kaggle and this makes it very difficult for data snooping to occur, thus making our result more reliable.

4.2 Baselines

We compare our BERT model with several other baselines methods as shown in Table 1.

4.2.1 Comparison with Traditional Machine Learning Methods

The usage of traditional machine learning methods such as Naive Bayes Classification could be said to be less effective to detect fake news. This is because it does not consider the structure of the sentences nor the relationship between neighbor words, but rather just the count of words alone. Also, it fails to identify stylometric features such as metaphors and idioms but treats them as independent words to be counted.

4.2.2 Comparison with other Deep Learning Methods

Comparing BERT to CNN and RNN, BERT is expected to produce better results than CNN and RNN. This is a result of BERT being able to understand the text inputs in context and correctly interpret the sentence more reliably than CNN and RNN. This would be in line with our results which showed that BERT was able to achieve a higher accuracy than CNN and RNN, as discussed in section 4.3.

4.3 Hyperparameters of BERT

For BERT, we achieved the best result with a maximum learning rate of either 1e-5 or 1e-6. The optimizer used was AdamW [34] with the Warm Up Linear learning rate scheduler and a warmup ratio of 0.1. The batch size was set to 32 for a balance between memory consumption and training time. The model converged after a single epoch.

4.4 Experimental Results

The results of our models are given in Table 1

	Validation Accuracy	Test Accuracy (Kaggle)	Validation Precision	Validation Recall	Validation F1-Score
Naive Bayes	45.00%	49.00%	0.11	0.65	0.58
CNN	77.72%	60.43%	0.79	0.82	0.81
CNN + handpicked features	65.46%	53.21%	0.74	0.66	0.69
CNN + GRU	73.07%	52.29%	0.82	0.71	0.76
CNN + GRU + handpicked features	73.28%	54.01%	0.75	0.83	0.79
RNN (LSTM)	77.31%	71.90%	0.71	0.70	0.70
RNN (GRU)	77.31%	73.28%	0.71	0.67	0.68
BERT	100.00%	79.13%	1.00	1.00	1.00

Table 1: Results of our various models

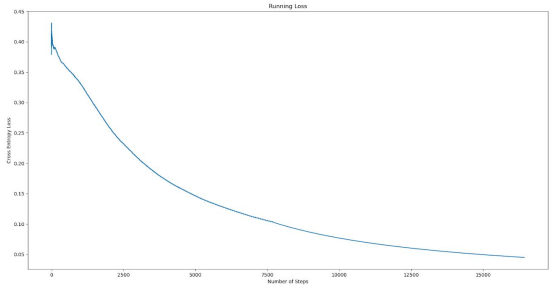

Figure 9: BERT Running Loss (Average Loss)

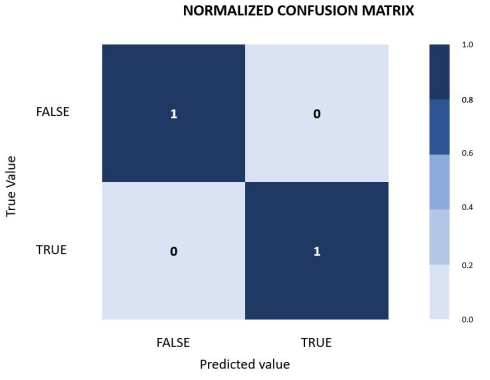

Figure 10: Normalised Confusion matrix of BERT

4.5 Result Interpretation

With the BERT model, we achieved a high accuracy score on the test dataset with 79%. This puts us in the top 8 of the public leaderboards for the Kaggle Challenge. (*For screenshots of the leaderboard and our score, see Appendix A for Figure 12 & 13).* Our team was able to obtain 100% accuracy on our training and validation set using our BERT model. We ultimately recognise that there is no such thing as a perfect model, and therefore we propose a few reasons why there could be such a significant difference in results for validation and the test set .

4.5.1 Choice of hyperparameter.

One reason for this can be the hyperparameter that we use. The chosen hyperparameter seems to work well on the training and validation data, but with the unseen data in testing, we have no control over the data. Therefore, this could have resulted in our test accuracy being lower than our training accuracy.

4.5.2 Difference in Training Dataset and Test Dataset.

We suspect that the gap between the validation and test dataset performance might imply some differences between the two datasets. After the train and validation split for our dataset, we suspect that our BERT model was not able to be generalised well enough to handle the test data.

4.5.3 Lack of Data to train Deep Learning Models

As our model was only trained on only 4731 entries of text data in the training set, we also suspect that it could be because since the BERT model is a deep learning model, there was not enough data to train our model to make it able to be generalizable to news datasets.

4.5.4 Possibility of Accuracy Paradox

We may be observing what is called an Accuracy Paradox. Accuracy Paradox for Predictive Analytics states that Predictive Models with a given level of Accuracy may have greater Predictive Power than Models with higher Accuracy [35]. Comparing the validation accuracy of the BERT (100%) and RNN model (77%), this may cause one to believe that such difference would also be seen in the test accuracy which was submitted to Kaggle. However, as seen in Table 1, the difference was not significant.

This phenomenon could be curbed by using other evaluation metrics such as entropy-modulated accuracy (EMA) and the normalized information transfer factor (NIT), which do not base the choice of classifiers using accuracy alone, but instead choose more meaningful and interpretable classifiers [36].

EMA is a pessimistic estimate of the expected accuracy with the influence of the input distribution factored out. More specifically, it works better than accuracy when the heuristic to maximise is the information transfer through the classifier [36]. NIT is a measure of how efficient is the transmission of information from the input to the output set of classes [36]. This factor measures the usefulness of the learning process in classifiers, makes it more difficult for them to use techniques such as specialization to 'cheat', and at the same time promote the interpretability of results [36].

The use of such metrics could give us a better evaluation of our model since we currently compared the performance of models using classification accuracy as the main metric.

5 Discussion

5.1 Possible Future Work

For this project, we have primarily focused on the classification of news articles as real or fake. However, given that the deep learning methods used are capable of more than just this purpose, we recognise that this project can be extended further to cover other areas of interest and fine tune existing methods to train models .

For instance, one possible extension to our project would be analysing the author's intention of a fake news article after classification. In particular, a focus area could be regarding satirical news which are often fake and are deliberately written for humorous purposes. The project can be extended to use deep learning to identify what the author's intention is in writing such an article. This could broaden our analysis beyond classifying news articles as genuine.

In addition, we could also further our analysis by finding the proportion of genuine news articles of various topics from a particular source. For instance, a source may provide real news for a topic like sports consistently but fake news about politics occasionally. In this way, we do not end up generalising the source as unreliable because of fake news in just one topic. Also, this could help to caution readers should the proportion of genuine news articles from that source be low.

5.2 Possible Improvements on our model approaches

Our implementation of CNN and RNN was all done starting from the news dataset itself and feeding the data directly. We could possibly improve upon this by making use of transfer learning. For example, the model developed by CNN could be used as a starting point for the model to be developed using RNN. This could be extremely beneficial in cutting down the time required to train models. This in turn could be helpful when larger datasets are involved. In addition, we can improve the training procedure by using Bootstrap and Bagging to randomize the training data and reduce the gap between validation and test accuracy. Furthermore, our current dataset is not big and general enough to scale up. We can improve the dataset by augmenting or generating more data using gaussian noise.

Additionally, we could also have used other data points apart from text data. Usage of image data has also shown to improve on certain neural network models [12].

6 Conclusion

This project has shown that being able to sift out the real news from fake news with high accuracy requires the use of deep learning methods to process language in cases where there is not much linguistic differences between fake and real news. Despite the capabilities of such deep learning models, it was necessary to perform data exploration to better understand the data and the results we should be expecting to get. We have attempted the implementation of several deep learning models and found that BERT and RNN perform better than CNN and are generally able to detect fake news with greater reliability. Ultimately, we hope that our findings in this paper will be able to support future research into the effectiveness of BERT models for fake news classification. We also hope that such technologies in the future can also mitigate or bring a stop to the spread of misinformation, threatening our society today.

References

[1] Shu, Kai & Sliva, Amy & Wang, Suhang & Tang, Jiliang & Liu, Huan. (2017). Fake News Detection on Social Media: A Data Mining Perspective. ACM SIGKDD Explorations Newsletter. 19. 10.1145/3137597.3137600. https://www.researchgate.net/figure/Fake-news-on-social-media-from-characterization-to-detection_fig1_318981549

[2] Bailey, J. (2020). 4 Reasons Why Fake News is So Compelling. Turnitin. https://www.turnitin.com/blog/4-reasons-why-fake-news-is-so-compelling

[3] Ng, H. (2018). 4 in 5 Singaporeans confident in spotting fake news but 90 per cent wrong when put to the test: Survey. The Straits Times. https://www.straitstimes.com/singapore/4-in-5-singaporeans-confident-in-spotting-fake-news-but-90-per-cent-wrong-when-put-to-the

[4] Chatfield, T. (2019). Why we believe fake news. BBC. https://www.bbc.com/future/article/20190905-how-our-brains-get-overloaded-by-the-21st-century

[5] Yuen-C, T. (2019, May 07). Parliament: Fake news law necessary to prevent crisis of trust that has hit other countries, says Shanmugam. Retrieved November 13, 2020, from https://www.straitstimes.com/politics/fake-news-law-necessary-to-prevent-crisis-of-trust-that-has-hit-other-countries-shanmugam https://www.straitstimes.com/politics/fake-news-law-necessary-to-prevent-crisis-of-trust-that-has-hit-other-countries-shanmugam

[6] Talwar, S., Dhir, A., Singh, D., Virk, G., & Salo, J. (2020, July 07). Sharing of fake news on social media: Application of the honeycomb framework and the third-person effect hypothesis. https://www.sciencedirect.com/science/article/pii/S0969698920306433

[7] Impacts of fake news. (2019, January 10). https://30secondes.org/en/module/impacts-of-fake-news/

[8] Hu, M., & Liu, B. (2004, August). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 168-177). https://dl.acm.org/doi/pdf/10.1145/1014052.1014073

[9] Bondielli, A. & Marcelloni, F. (2019). A survey on fake news and rumour detection techniques, Information Sciences, Volume 497, Pages 38-55, ISSN 0020-0255 https://www.sciencedirect.com/science/article/pii/S0020025519304372

[10] Feng, S., Banerjee, R., & Choi, Y. (2012, July). Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 171-175). https://www.aclweb.org/anthology/P12-2034.pdf

[11] Kim, Y. (2014). Convolution Neural Network for Sentence Classification https://arxiv.org/pdf/1408.5882.pdf

[12] Yang, Y., Zheng, L., Zhang, J., Cui, Q., Li, Z., & Yu, P. S. (2018). TI-CNN: Convolutional neural networks for fake news detection. https://arxiv.org/pdf/1806.00749.pdf

[13] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. https://arxiv.org/pdf/1810.04805.pdf

[14] Lakew, S. M., Cettolo, M., & Federico, M. (2018). A comparison of transformer and recurrent neural networks on multilingual neural machine translation. https://arxiv.org/pdf/1806.06957.pdf

[15] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008). https://arxiv.org/pdf/1706.03762.pdf

[16] Jeffrey Pennington, Richard Socher, & Christopher D. Manning. (2014). GloVe: Global Vectors for Word Representation. https://nlp.stanford.edu/pubs/glove.pdf

[17] Glorot, X., Bordes, A., & Bengio, Y. (2011, June). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315-323). http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf

[18] Hertz, J. A. (2018). Introduction to the theory of neural computation. CRC Press. https://books.google.com.sg/books?hl=en&lr=&id=NwpODwAAQBAJ&oi=fnd&pg=PT13&dq=Hertz,+J.+A.+(2018).+Introduction+to+the+theory+of+neural+computation.+CRC+Press.&ots=FNbaGi6bFt&sig=vdRVpVSHf8oLODCOybyXE6tgGH4&redir_esc=y#v=onepage&q&f=false

[19] Prechelt, L. (1998). Early stopping-but when?. In Neural Networks: Tricks of the trade (pp. 55-69). Springer, Berlin, Heidelberg. https://link.springer.com/chapter/10.1007/3-540-49430-8_3

[20] Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press. https://books.google.com.sg/books?hl=en&lr=&id=RC43AgAAQBAJ&oi=fnd&pg=PR7&dq=Murphy,+K.+P.+(2012).+Machine+learning:+a+probabilistic+perspective.+MIT+press.&ots=umfzbAQoZd&sig=ruDussGvnfLiFVleB3HZS2Bymtc&redir_esc=y#v=onepage&q=Murphy%2C%20K.%20P.%20(2012).%20Machine%20learning%3A%20a%20probabilistic%20perspective.%20MIT%20press.&f=false

[21] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. https://arxiv.org/abs/1412.6980

[22] Nigam, V. (2020). Natural Language Processing: From Basics to using RNN and LSTM. Medium. https://towardsdatascience.com/natural-language-processing-from-basics-to-using-rnn-and-lstm-ef6779e4ae66#:%7E:text=Recurrent%20Neural%20Networks%20or%20RNN,used%20in%20Natural%20Language%20Processing.&text=It%20also%20provides%20an%20additional,in%20a%20standard%20neural%20network.

[23] Phi, M. (2020). *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Medium. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[24] Kostadinov, S. (2019, November 10). *Understanding GRU Networks - Towards Data Science*. Medium. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[25] Sharma, S. (2019). *Activation Functions in Neural Networks - Towards Data Science*. Medium. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6#:~:text=The%20main%20reason%20why%20we,The%20function%20is%20differentiable.

[26] *Activation Functions : Why "tanh" outperforms "logistic sigmoid"*. (2019). Mc.Ai. https://mc.ai/activation-functions-why-tanh-outperforms-logistic-sigmoid/#:%7E:text=Getting%20stuck%20during%20training%20(train%20time)%3A&text=Logistic%20sigmoid%20can%20cause%20a,are%20very%20near%20to%20zero.

[27] Ye, A. (2020). *Swish: Booting ReLU from the Activation Function Throne.* Medium. https://towardsdatascience.com/swish-booting-relu-from-the-activation-function-throne-78f87e5ab6eb

[28] Devlin, J., & Chang, M. (2018, November 2). Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing. https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

[29] . (2019, February 19). Why BERT has 3 Embedding Layers and Their Implementation Details. https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

[30] Vig, J. (2020, October 29). Deconstructing BERT, Part 2: Visualizing the Inner Workings of Attention. https://towardsdatascience.com/deconstructing-bert-part-2-visualizing-the-inner-workings-of-attention-60a16d86b5c1

[31] Stickland, A. C., & Murray, I. (2019). Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. arXiv preprint arXiv:1902.02671. https://arxiv.org/abs/1902.02671

[32] Ba, J., Kiros, R., and Hinton, G. E. Layer normalization. CoRR, abs/1607.06450, 2016. https://arxiv.org/abs/1606.08415

[33] Hendrycks, D. and Gimpel, K. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. CoRR, abs/1606.08415, 2016. https://openreview.net/forum?id=Bk0MRI5lg

[34] Gugger, S., and Howard,. (2018, July 2). J. AdamW and Super-convergence is now the fastest way to train neural nets. https://www.fast.ai/2018/07/02/adam-weight-decay/

[35] Afonja, T. (2017, December 8). Accuracy Paradox. https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b

[36] Valverde-Albacete, F., & Peláez-Moreno, C. (2014, January 10). 100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox. https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0084217

Additional References

Wang, W. Y. (2017). " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*. https://arxiv.org/abs/1705.00648

Hai, Z., Zhao, P., Cheng, P., Yang, P., Li, X. L., & Li, G. (2016, November). Deceptive review spam detection via exploiting task relatedness and unlabeled data. In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 1817-1826). https://www.aclweb.org/anthology/D16-1187.pdf

Mihalcea, R., & Strapparava, C. (2009, August). The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers* (pp. 309-312). https://www.aclweb.org/anthology/P09-2078.pdf

BERT Encoder. (2020). Peltarion.Com. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/bert-encoder

Ghelani, S. (2019). *Breaking BERT Down - Towards Data Science*. Medium. https://towardsdatascience.com/breaking-bert-down-430461f60efb

(2019). *How the Embedding Layers in BERT Were Implemented*. Medium. https://medium.com/@_init_/why-bert-has-3-embedding-layers-and-their-implementation-details-9c261108e28a

Horev, R. (2018). *BERT Explained: State of the art language model for NLP*. Medium. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Appendix A

CNN	CNN + Handpicked Features		CNN + GRU	CNN + GRU + Handpicked Features	
Latent	Explicit	Latent	Latent	Explicit	Latent
Emb (500,100)	Input 6 × 1	Emb (500,100)	Emb (500,100)	Input 6 × 1	Emb (500,100
(3,3) Conv1D(80)		(3,3) Conv1D(32)	(3,3) Conv1D(32)		(3,3) Conv1D(32)
2 Maxpo1D	Dense(10) ReLU	4 Maxpo1D	2 Maxpo1D		4 Maxpo1D
Dropout(0.2)		(3,3) Conv1D(64)	Dropout(0.3)		Dropout(0.3)
Flatten		4 Maxpo1D	(3,3) Conv1D(64)		(3,3) Conv1D(64)
Dense 150 ReLU	Dense(10) ReLU	(3,3) Conv1D(128)	2 MaxPo1D	Dense(10) ReLU	4 MaxPo1D
Dropout(0.2)		4 Maxpo1D	Dropout(0.35)		Dropout(0.35)
Dense(10) ReLU		Flatten	(3,3) Conv1D(128)		(3,3) Conv1D(128)
Dense(1) Sigmoid	Merge		2 Maxpo1D		4 Maxpo1D
	Dense(128) ReLU		Dropout(0.4)	Dense(10) ReLU	Dropout(0.4)
	Dense(10) ReLU		GRU(50)		GRU(50)
	Dense(1) Sigmoid		Dropout(0.25)		Dropout(0.25)
			Flatten		Flatten
			Dense (128) ReLU	Merge	
			Dense(1) Sigmoid	Dense (128) ReLU	
				Dense (10) ReLU	
				Dense (1)) Sigmoid	

Figure 11: Architecture of CNN model and its variants.

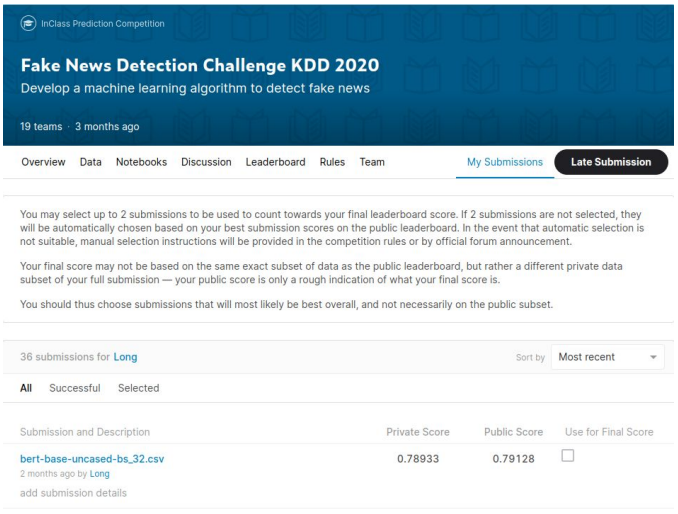

Figure 12: Kaggle's test accuracy for BERT

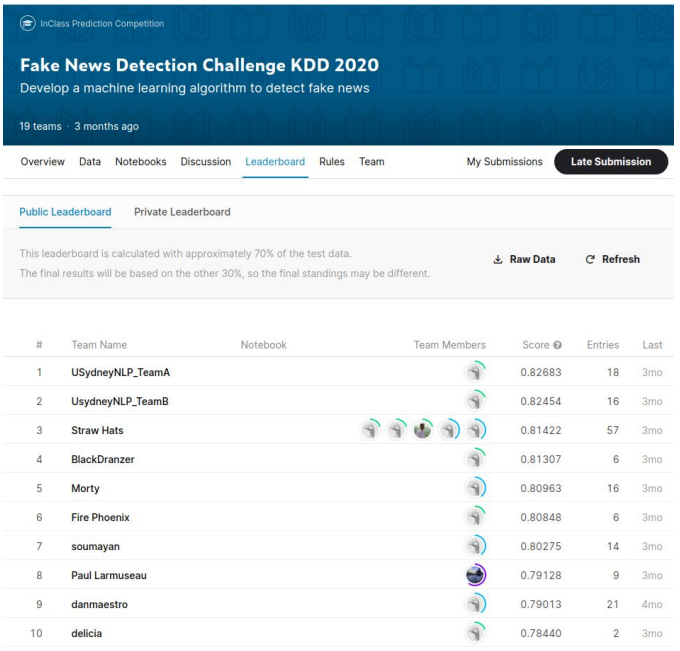

Figure 13: Kaggle's leaderboard for test accuracy in Fake News Detection Challenge

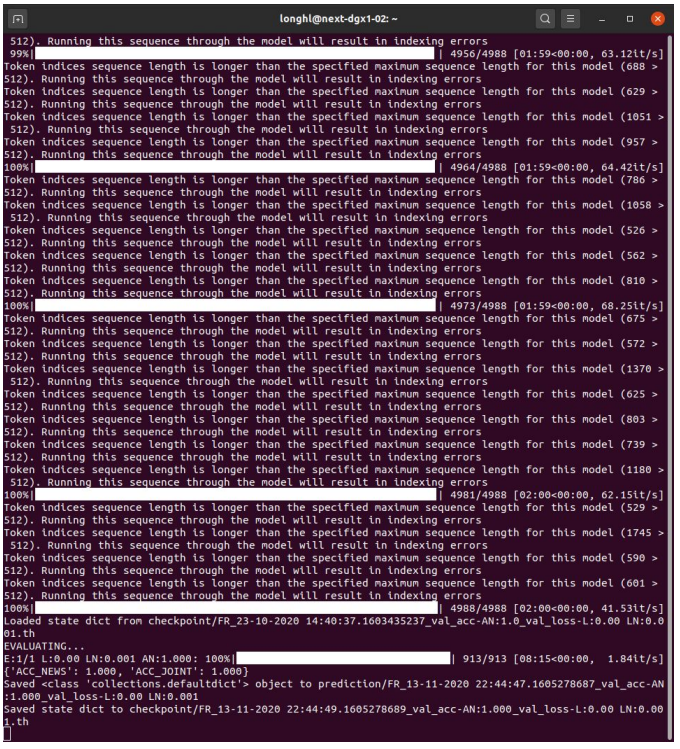

Figure 14: Screenshot of BERT's validation score