Stance Detection with BiLSTM, CNN, and BERT
MSCI-641 Fake News Chanllenge

Tianyu Shi
University of Waterloo
t33shi@uwaterloo.ca

Yamin Zhou
University of Waterloo
y629zhou@uwaterloo.ca

Abstract

The Fake News Challenge was organized in early 2017 to encourage development of machine learning-based classification systems that perform "stance detection" – i.e. identifying whether a particular news headline "agrees" with, "disagrees" with, "discusses," or is unrelated to a particular news article. We developed several deep neural network-based models and used BERT, Google's neural network-based technique for natural language processing (NLP) pre-training, to tackle the stance detection problem. We found that BERT model delivered the best performance.

1 Introduction

"The rise of fake news highlights the erosion of long-standing institutional bulwarks against misinformation in the internet age. Concern over the problem is global."(Lazer et al., 2018) According to a Pew Research report (Barthel et al., 2016), 64% of US adults said that "made-up news" has caused a "great deal of confusion" about the facts of current events. Fake news in social media may influence many aspects of people's daily life, or even the outcome of a presidential election (Allcott and Gentzkow, 2017).

The Fake News Challenge (FNC) aims to explore the strategies to combat the fake news problem using machine learning and natural language processing (NLP) techniques. (Pomerleau, 2016a) The stage-1 task of FNC is **Stance Detection** that estimates whether the body text from a news article may agree, disagree, discuss or be unrelated to the headline.

The evaluation system of the FNC-1 task is illustrated in the Figure 1. In fake news detection, [Related, Unrelated] classification task is expected to be less important and easier than the task of classifying [Agrees, Disagrees, Discusses]. In consequence, 25% score weight is assigned to the [Related, Unrelated] task, and 75% score weight is distributed to the [Agrees, Disagrees, Discusses] task.

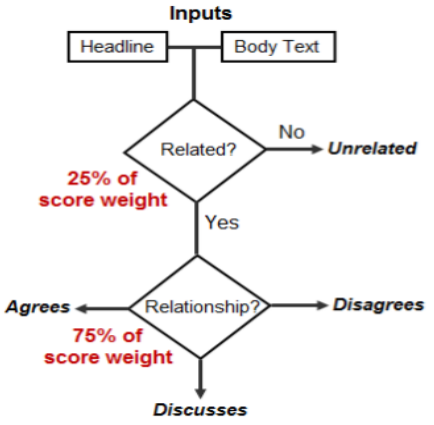

Figure 1: FNC-1 Scoring System (Pomerleau, 2016a)

A baseline model using hand-crafted features (word/ngram overlap features, indicator features for polarity and refutation) and GradientBoosting classifier is provided by the FNC-1 organizers.(Galbraith, 2016) The baseline model achieves a weighted accuracy score of 79.53% and a competition score of 8748.75.

Our approach of investigating the FNC-1 problem includes Bidirectional LSTMs and CNN Long Short-Term Memory Network. We also fine tune the pre-trained multilingual BERT model for our 4-categories-classification task.

2 Background

2.1 Convolutional Neural Network

A convolutional neural network (CNN) is a kind of feed-forward articicial neural network designed to learn local patterns in local spatially correlated gridded data. Thus, a one dimensional CNN can be designed to extract local information from a text stream by scanning across n-grams that have been encoded using a word-to-vector embedding mechanism. The resulting high-level features can be successfully used for sentiment analysis, machine translation, text classification, and other NLP tasks.

CNN always contains two basic operations, namely convolution and pooling. The convolution operation using multiple filters is able to extract features (feature map) from the data set, through which their corresponding spatial information can be preserved. The pooling operation, also called subsampling, is used to reduce the dimensionality of feature maps from the convolution operation.

2.2 Long Short-Term Memory Units

Long Short Term Memory architecture is a special kind of RNN introduced by (Hochreiter and Schmidhuber, 1997), which enables one to learn long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior. Moreover, LSTMs are the most powerful and well known subset, are a type of artificial neural network designed to recognize patterns in sequences of data. What differentiates RNNs and LSTMs from other neural networks is that they take time and sequence into account, they have a temporal dimension.

2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art language representation model published by researchers at Google AI language in 2018. BERT learns and tries to make improvements on top of many recent clever ideas in the Natural Language Processing (NLP) community, such as Semi-supervised Sequence Learning (Dai and Le, 2015), ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), the OpenAI transformer (Radford et al., 2018), and the Transformer (Vaswani et al., 2017). It is considered that BERT is one of the latest milestones in the field of NLP, which broke several records on NLP tasks such as GLUE (7.7% point absolute improvement) and SQuAD v2.0 Test F1 (5.1% point absolute improvement). (Devlin et al., 2018)

BERT is a fine-tuning based model that contains two steps in its framework. The first step is pre-training, where the model experiences semi-supervised training (Dai and Le, 2015) on large amounts of unlabeled data. After pre-training, the model could be initialized with the same parameters and will be fine-tuned using labeled data for specific tasks. BERT is directional, meaning that it reads the entire sequence of words at once rather than sequentially reading from left-to-right or right-to-left. Generally speaking, bidirectional models are considered more powerful than uni-directional language models. In order to achieve the bidirectional feature, BERT uses a masked language model (MLM) (Taylor, 1953) in pre-training that avoids the problem of information leaking in the lower layers and enables a token to see itself in deeper layers. It should also be noted that BERT is pre-trained with a binarized *next sentence prediction* (NSP) task so that BERT obtains a better understanding of the relationship between two sentences.

The core architecture of BERT is the stack of bidirectional Transformer encoders (Vaswani et al., 2017). The structure of the Transformer is shown in Figure 3, which contains two parts called encoder and decoder.

Two BERT models of different sizes were posted in the paper (Devlin et al., 2018) where one is called $\mathbf{BERT_{BASE}}$ and the other is called $\mathbf{BERT_{LARGE}}$. There are 12 encoder layers with 768 hidden units in $\text{BERT}_{\text{BASE}}$ and 24 encoder layers with 1024 hidden units in $\text{BERT}_{\text{LARGE}}$. The large version BERT also has more self-attention heads (16) and parameters (340M) than the base version BERT (12 heads, 110M parameters).

2.3.1 Masked Language Model (MLM)

As mentioned before, BERT takes a procedure called "masked LM" (MLM) to train a deep bidirectional representation. BERT chooses 15% of the input tokens at random, within which 80% are replaced with [MASK] token, 10% are replaced

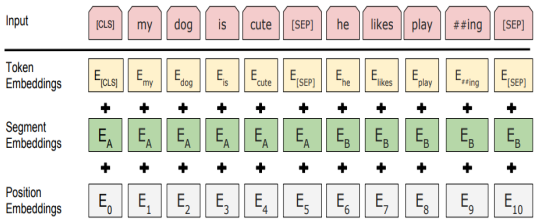

Figure 2: BERT input representation(Devlin et al., 2018)

with a random token, and 10% are kept unchanged. The model attempts to predict the original value of the masked words using the non-masked tokens in the sequence. (Devlin et al., 2018)

2.3.2 Next Sentence Prediction (NSP)

In the Next Sentence Prediction (NSP) task, pairs of sentences are the inputs and BERT learns to determine whether the second sentence is the subsequent sentence of the first sentence. The model distinguish the two sentences based on the [CLS] token inserted at the beginning of the first sentence and the [SEP] tokens inserted at the end of each sentence. The input embeddings are composed of the token embeddings, the segmentation embeddings, and the position embeddings as shown in Figure 2.

The above input sequence will be processed by the Transformer, and produces the output of the [CLS] token that could be sent into a classification layer such as softmax to calculate the probability of sentence B being the subsequent sentence of sentence A.

It is not hard to fine-tune BERT for classification tasks which follows the same logic as the NSP task. For the FNC-1 task, we could treat the body sequence as sentence A and the head sequence as sentence B, then classifies the output into four categories [Unrelated, Agrees, Disagrees, Discusses].

2.3.3 Transformer

The Transformer is a powerful sequence transduction model based solely on attention mechanisms.

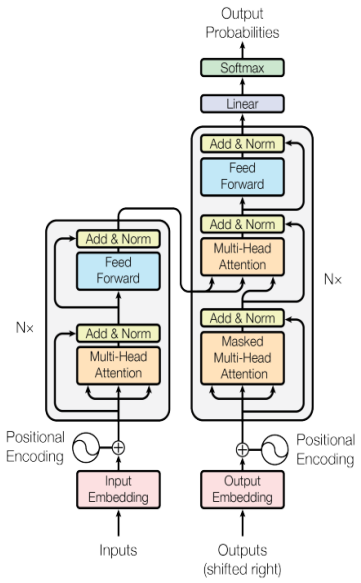

Figure 3: The Transformer - model architecture (Vaswani et al., 2017)

The architecture of the Transformer could be described as stacks of encoders and decoders connecting with self-attention and point-wise, fully connected layers. There are encoder self-attention, decoder self-attention, and encoder-decoder attention in the Transformer model. With self-attention, each position in the encoder/decoder could attend to all positions in the previous layer of encoder/decoder. The output of the encoder will be sent into the encoder-decoder attention layers as input sequence where all positions are allowed to be attended by every position in the connected decoder. (Vaswani et al., 2017)

The structure of the attention is shown in Figure 4. The first attention is called "Scaled Dot-Product Attention" that takes queries, keys (dimension d_k), and values (dimension d_v) as input. The calculation is performed on the matrix scaled by the following equation (Vaswani et al., 2017):

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

The format of attention used in the Transformer is called "Multi-Head Attention" that is the concatenation of the results from h scaled by the attention dot-product. The queries, keys, and values are linearly projected h times, which enables the model to jointly attend to information from different repre-

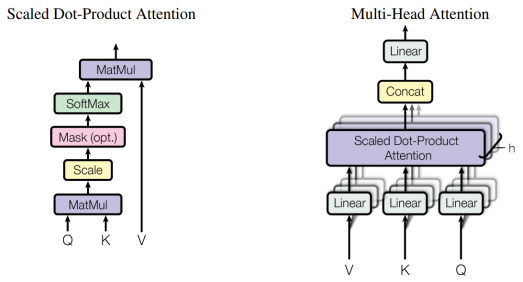

Figure 4: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. (Vaswani et al., 2017)

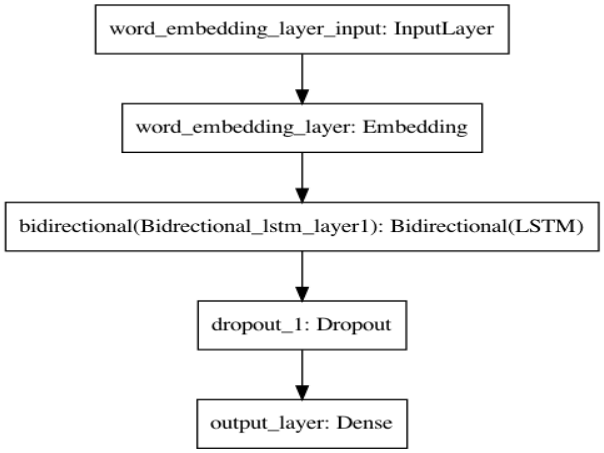

Figure 5: The Structure of Bi-directional LSTM model

sentation subspaces at different positions. (Vaswani et al., 2017)

3 Approach

The fake news detection problem is a classification problem. The proposed solution detects fake news by determining whether the information present in the article is correct on the basis of quantifying the bias of a written news article and by analyzing the relationship between the news article headline and article body. This section describes the structure of the models and the data processing method used for the experimentation.

3.1 Bi-directional Long Short-Term Memory

For Bidirectional LSTMs model, we tokenize the sentences into words. Then words are converted into a word embedding matrix (input embedding layer) of 50 dimension by using GloVe. Then we add a Bi-directional LSTM layer with the number of layer units 100. A dropout layer added to avoid overfitting.The structure of the Bidirectional LSTMs model as in Figure 5

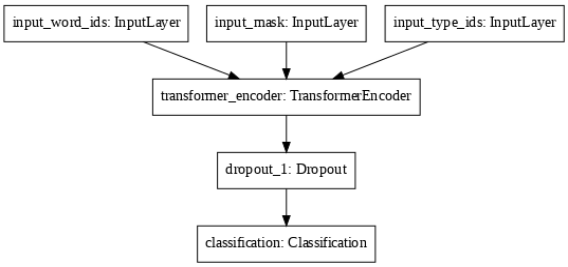

Figure 6: The Structure of FNC-1 BERT model

3.2 CNN Long Short-Term Memory model

In our CNN Long Short-Term Memory model, the sentences are first tokenized into words. Then words are converted into a word embedding matrix (input embedding layer) of 50 dimension. We use GloVe for converting sentences into sentence matrices. Convolutional filters of different window sizes are applied to this input embedding layer to generate a new feature representation. Pooling method is applied on new features and pooled features from different filters are concatenated with each other to form hidden representation. They combined with LSTMs to support sequence prediction. These representations are then followed by one or multiple fully connected layers to make the final prediction

3.3 BERT model

In our BERT model, token embeddings, segment embeddings, and position embeddings are sent into the Transformer structure as the input. The text body and text head will be tokenized and will be concatenated into length of 512, and will be processed by the Transformer, as in Figure 6. The output layer performs the stance classification job that determines the relationship between the input text body and text head.

4 Experiments and Results

4.1 Data Description and Pre-processing

Datasets for the FNC-1 task could be downloaded from the GIT repository (Pomerleau, 2016b). To be specific, 'train_bodies.csv' and 'train_stances.csv' are combined into one based on BodyID and then splitted into training set and validation set with 90%-10% ratio for the BERT model and with 80%-20% ratio for LSTM models. The test set is prepared by combining 'competition_test_bodies.csv' and 'competition_test_stances.csv'.

In the training set, the text head has string length

Stance	Percentage
Unrelated	73.13%
Agrees	7.36%
Disagrees	1.68%
Discusses	17.82%

Table 1: Percentage of each stance category in training set

in the range from 9 to 225, and the text body has minimum string length of 38 and maximum string length of 27579. The training set has 49972 rows in total, and the distribution of each stance is shown in Table 1.

For the pre-processing of LSTM models, an integer label is set to each news article as 0 for unrelated, 1 for agree, 2 for disagree and 3 for discuss. The news articles' head and body are put together as 'text', stop words are removed. Then we split the sentences into lists of tokens, At the same time, the contents are transformed into lower case,punctuation are removed. Then the sentences are turned into space-separated padded sequences of words. Pre-trained GloVe word embeddings are used to deal with the high dimensional news articles. The dimension of GloVe that I used is 50, and it is trained on a Twitter Corpus. For the embedding layer, a masking technique was used to ensure that the zero-padded input would not influence loss calculations.

For the pre-processing of the BERT model, we tokenize the text body and text head using the BERT tokenizer that transforms the contents into lower case, handles the punctuation, and tokenizes the text piecewisely. [CLS] token will be inserted at the beginning of text body sequence and text head sequence, and [SEP] token will be inserted at the end of each sequence. We observe that the sequence length of the tokenized text body could be much larger than the embedding sequence limit of 512, and the sequence length of the tokenized text head is about 80. Therefore, we keep all the tokens from the text head, and truncate a portion of tokens from the text body in order to produce a concatenated sequence of length less than 512. Zero-padding is implemented later so that we obtain input that fits the BERT embedding layer. As mentioned in the NSP task, the output of the BERT tokenizer contains token embeddings, segment embeddings, and position embeddings. Similar to the pre-processing of LSTM models, the four categories are relabeled by integer from 0 to 3.

4.2 Fine-tuning

For the LSTMs models, there are many parameters such as dropout rate, number of epochs, batch size, token length, etc. that can be tuned. Since LSTMs are really expensive to train, we only consider some parameter sets. For both models, the number of layer units is 100. We apply different dropout probabilities for the models and finally we choose 0.8 and 0.2 for the two models.

The maximum number of words of the 10th percentile of the combined head and body after removing stopwords, sequenced by number of words, is 596 words. Similarly, the 50th percentile is 1437, and the 90th is 3179. It is tricky to choose the appropriate cut-off because it is much too expensive to train when the full headlines and bodies are used. We try token lengths of 150 and 1000 and we find that the longer the token length is, the longer the training time will take, but it improves the final accuracy.

Hidden Units for the LSTMs influences the performance. Large number of hidden units make the model easily to overfit the training dataset. We have tried 300, 200, 128, 100, 64 units. We use 128, 64 for two connected hidden layers in the CNN LSTM model. However, this does not mean that they are the best choice for our task since we have only tried limited number of choice due to the intensive demand for computing power for even training one LSTM.

We do not specify the learning rate since we used Adam (AdamW).(Loshchilov and Hutter, 2017) as the way of optimization and it uses an adaptive learning rate to train the model.We also apply different number of epochs and batch size. Both models seems to converge after 5 epochs with batch size 128.

For the FNC-1 BERT model, we investigate parameters such as epoch, batch size, truncation length, learning rate, etc. For the optimizer, we choose a variant version of Adam with decoupled weight decay (AdamW).(Loshchilov and Hutter, 2017) Table 3 shows the parameters, metric, and loss function

Parameters	Values
Epoch	5,6,10,40
Batch Size	128
Token length	150, 1000, 2000
Truncation Length	150
Learning Rate	0.001
Optimizer	AdamW
Vocabulary Size	40000
Dropout	0.2,0.8
Metric	accuracy
Loss Function	CategoricalCrossentropy

Table 2: LSTMs Hyperparameter Setting

Parameters	Values
Epoch	2,3,4
Batch Size	16,32,64
Truncation Length	150,200,250,300,350,400
Learning Rate	$2 \cdot 10^{-5}$,$3 \cdot 10^{-5}$,$4 \cdot 10^{-5}$,$5 \cdot 10^{-5}$
Optimizer	AdamW
Vocabulary Size	30522
Dropout	0.1
Metric	SparseCategoricalAccuracy
Loss Function	SparseCategoricalCrossentropy

Table 3: BERT Hyperparameter Setting

that we have tested. It takes about 150 to 300 seconds to train one epoch with our parameter setting using the free TPU supported by Google Colab.

4.3 Result

To check the prediction accuracy to see whether the news article is correctly classified, the "accuracy" evaluation metric is used. Accuracy is calculated as the ratio of the number of correctly predicted samples to the total number of samples for a given data set.

From Table 4 we can see that Bert model gets the highest accuracy among the three models. The sparse categorical accuracy of the BERT model on the test set ranges from 91.80% to 93.47%. The parameter combination that gives the best performance uses batch size of 16, truncation length of 400, and learning rate of $4 \cdot 10^{-5}$, which achieves a competition score of 10439.25. The confusion

Model	TrainAccy	ValAcc	TestAcc
BiLSTM	72.33%	74.34%	70.04%
CNN LSTM	80.56%	73.53%	72.20%
Bert	99.53%	99.46%	93.47%

Table 4: Result of models

Actual \Pred	agree	disagree	discuss	unrelated
agree	1286	171	280	19
disagree	89	270	78	12
discuss	502	204	4004	125
unrelated	26	52	102	18193

Table 5: Confuse matrix of Bert model

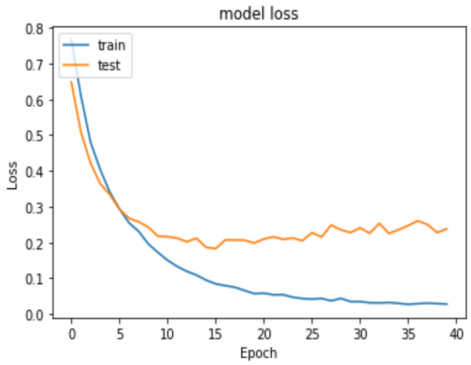

Figure 7: CNN LSTM model loss with 40 epochs

matrix (Table 5) is used for evaluation.

We set up 40 epochs for training the LSTM models at the beginning. After checking the pictures of loss (Figure 7) we can see that it is over-fitted. The better number of epochs for the LSTM models is 5 (Figure 9) and for the BERT model is 3 (Figure 8). Figure 10 shows that at 1 epoch the Bert has much higher prediction accuracy than the other two models.

4.4 Discussion

Fake news detection has been examined using several methods in accordance with the scope and format of available fake news data and technical approaches. Our focus is on verifying whether the headline matches the body text. In this study, we tried several deep neural network-based models.

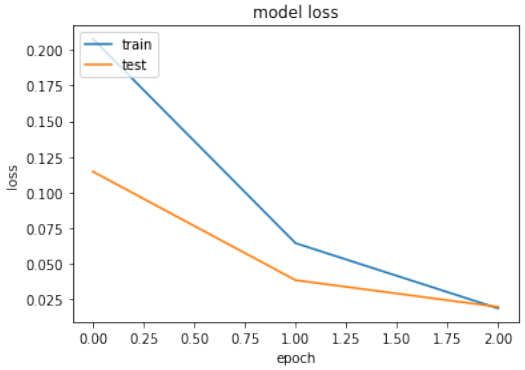

Figure 8: BERT model loss with 3 epochs

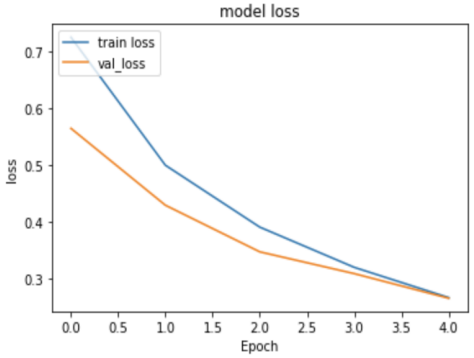

Figure 9: BiLSTM model loss with 5 epochs

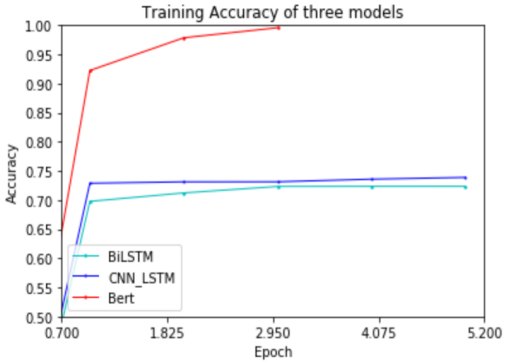

Figure 10: fig: Training Accuracy of three models with 5 epochs

However, the results of these models do not exhibit good performance and can be improved.

By using the confusion matrix to check the results we find that the CNN_LSTM model failed because the two convolutional layers, which came first in the network, ended up overpowering the LSTM layer. The local correlations learned by the convolutional layers, intended to be the meanings of short phrases or sentences, do not convey the necessary information to determine the classification, whereas the LSTM layer is designed to take into account the long-term correlations that encode this information.

In addition, word embedding is important for improving the performance of the model. word2vec and GloVe were previously used for word embedding. These do not exhibit good performance because they employ fixed vector values rather than fluid vector values for words. To complement this, we use BERT. The results of using BERT in this study has much higher performance than other models.

Pre-trained word embedding model: For this project, we downloaded the GloVe model online with 50 dimensions. This model is trained on a large corpus, which produces a robust representation of words, however, this also leads a lot of out-of-vocabulary words and this may lose a lot of information of the original texts. The other problem of word embedding is polysemy, i.e. when a word has two different meanings. Polysemous words are a common phenomenon in natural language, and also a manifestation of language flexibility and efficiency. Words are embedded into vectors based on the learned context of the word, without any mechanism to distinguish multiple possible meanings, as in the case of a polysemy.

Overfitting problem: We found that it is easy to overfit. We get over 90% accuracy on the validation data set when the models are trained. However, the accuracy on test data is much lower. We tried the following different ways to reduce this problem:

- reduce the network's size by removing layers or reducing the number of hidden elements in the layers
- add regularization, which comes down to adding a cost to the loss function for large weights
- add dropout layers, but it will randomly remove certain features by setting them to zero

LSTM tends to forget long-distance relationships: The forget cell in the LSTM can easily lose information for relationship between distant words when the sequence is too long.

5 Conclusion and Future Ideas

5.1 Conclusion

The aim of this work is to predict fake news articles by exploring different models. The BiLSTM is well suited for a long-range semantic dependency based classification, while The CNN_LSTM performs better for extracting local and position-invariant features. However, these two models were outperformed by the BERT model. While the unrounded predictions of the CNN_LSTM model in one-hot form varied slightly, they almost all leaned towards the "unrelated" classification. Thus, in the end, this model was nearly trivial in the sense that it

classified nearly all articles as "unrelated". Interestingly, this ended up giving a fair accuracy of 71%, due to the skewness of the data. The BERT model gives a considerably high accuracy on predicting fake news; however, there is still space for improvement.

5.2 Future Ideas

Our BERT model is limited when predicting the 'disagree' class since only 270 out of 499 (54.11%) cases are correctly predicted. Our fine-tuning process should cover more hyperparameters like dropout rate and vocabulary size. Despite using the default tokens in BERT vocabulary, we could add and build tokens representing our desired features. Furthermore, we could modify the original architecture of BERT to improve the performance of it. There are many state-of-the-art models trying to improve and overcome the limitations of BERT for us to explore in the future, for example, DistilBERT (Sanh et al., 2019), RoBERT (Liu et al., 2019), ALBERT (Lan et al., 2019), and XLNet (Yang et al., 2019).

6 Appendix

Here is the link to our GitHub repository for this project: `https://github.com/stone1994105/FNC-1`

References

Hunt Allcott and Matthew Gentzkow. 2017. Social media and fake news in the 2016 election. *Journal of economic perspectives*, 31(2):211–36.

Michael Barthel, Amy Mitchell, and Jesse Holcomb. 2016. Many americans believe fake news is sowing confusion. *Pew Research Center*, 15:12.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

et al. Galbraith. 2016. fnc-1-baseline.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

David MJ Lazer, Matthew A Baum, Yochai Benkler, Adam J Berinsky, Kelly M Greenhill, Filippo Menczer, Miriam J Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, et al. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Rao Pomerleau. 2016a. Fake news challenge stage 1 (fnc-i): Stance detection.

Rao Pomerleau. 2016b. fnc-1-data.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.