Comparison of RNN and Transformer Architecture on Stance Detection

SIHAN WU
University of Waterloo
Electrical And Computer Engineering
s357wu@uwaterloo.ca

YUZHE ZHANG
University of Waterloo
Electrical And Computer Engineering
y2877zha@uwaterloo.ca

Abstract

The Fake News Challenge(FNC) is a natural language processing(NLP) task dealing with identifying fake information. As the first stage, stance detection is of vital importance. By abstracting this problem as a text classification problem, we present a comparison between tradition recurrent models and newly introduced transformer models. To further show the capability of capturing contextualized text representation of the transformer model, we also experiment with two models using its produced representation feature. In the end, our best BERT model builds a state-of-the-art FNC score with 10247.5/11651.25(87.95%) point, which surpasses all the recurrent models. And models using BERT generated features are all easily outperforms the recurrent models and given baseline model as well.

1 Introduction

The Fake News Challenge(FNC) is a natural language processing(NLP) task, which tries to explore the performance of artificial intelligence in assessing the veracity of news story(Pomerleau and Rao, 2017). Serving as the first stage of FNC, stance detection is realized by classifying the relationship between the given headline and news body into different four categories, which includes Agree, Disagree, Discuss and Unrelated.

Abstracting this problem as a text classification problem, there are multiple models available for addressing it. Recurrent neural network(RNN) architecture with its natural ability to process sequence of text has always been a good choice. As LSTM(Schmidhuber and Hochreiter, 1997) further improved with attention mechanism(Rocktäschel et al., 2015) enabled, the potential of recurrent models are considerable.

Even though, due to its recurrent nature, the ability of its parallel computing is limited. On the other hand, the newly introduced model with transformer(Vaswani et al., 2017) architecture has shown great capability in parallel computation. And the text representation it generated is also shown to be better than the embedding produced by recurrent models.

In this work, we present a comparison between the recurrent architecture and transformer architecture on the text classification task, the FNC stance detection. To show the ability of the transformer model, we also use its generated representation as a feature. And finally, we show that BERT models and GradientBoost models using BERT features all outperform the recurrent models with around 30% on the FNC score. And the BERT model with the sequence length of 512 presents the best result on both FNC score and F1 score. As a conclusion, we argue that the transformer architecture model, BERT, is better than the simple traditional recurrent models.

2 Related Work

Text classification is a task of assigning predefined labels to given texts. And there are lots of approaches for solving this problem, such as Naive Bayes, convolution neural network, recurrent neural network, newly introduced transformer structure(Vaswani et al., 2017) and so on.

The Naive Bayes approach can be implemented in two ways with different assumption(McCallum et al., 1998). The basic idea behind the Multi-Variate Bernoulli model makes a simple assumption that each word in the document is generated independently of other words from a Bernoulli distribution. It then calculates the probability of each word in the document against each label to predict the given texts. On the other hand, Multinominal model utilizes the frequency information. However, both methods make simple assumption that

the length is independent of labels, which might be inappropriate. As for the long sequence of documents, a word is more likely to appear than in short ones.

Approaches of deep learning have shown great performance, like the convolution architecture proposed by (Kim, 2014), and gained more and more attention recently. Such architecture normally has some embedding layers at the beginning of the model to map the word text into continuous word vectors. The mapping is usually implemented with the Word2Vec models. Such a model is normally fed with a large amount of text and produce some fixed length word representation. These representations then possess the property that words sharing common context would locate close to each other in the vector space. Popular models are like which (Mikolov et al., 2013) proposed, the Continuous Bag of Word(CBOW) and Skip-gram. They produce the representation matrix that trained on 1.6 billion words. Instead of learning through predicting the surrounding words, another approach Glove proposed by (Pennington et al., 2014) produce the representation based on the weighted global co-occurrence matrix.

Mapping the input word in the text to continuous vector representation through those pre-trained word embedding then offers a very simple and brutal text representation for the next stage. To be clear, take this representation as features, it might be applied to lots of approaches.

Recurrent neural network(RNN) has been broadly applied to natural language processing(NLP) problem. This kind of neural network is designed for modeling sequential data and has been testified to be quite efficient in sequential tagging tasks. In the text classification field, it usually maps the input tokens into pre-trained embedding.

LSTM neural network, which stands for long short-term memory, introduced by (Schmidhuber and Hochreiter, 1997), is a particular type of recurrent neural network. Such network manages to keep contextual information of inputs by integrating a loop that allows information to flow from one step to the next. In this architecture, there is an update gate and a forget gate instead of one update gate in Gated Recurrent Unit(GRU)(Cho et al., 2014). This architecture gives the memory cell an option of keeping the old value at the previous time step and adding to it the value at the current time step.

However, as standard LSTM network process sequence in time series, it leads to neglecting contextual information in the future. To tackle the problem, bi-directional LSTM network was presented(Schuster and Paliwal, 1997). The basic idea of bi-directional model is to split traditional RNN neuron into two parts, one for positive time direction(forward states), the other for negative time direction(backward states). This structure provides complete contextual information in the past and future to each node of the input sequence in the output layer.

Besides, attention mechanism(Rocktäschel et al., 2015) is widely used in the fields of machine learning, speech recognition and image caption, which aims at improving the model of encoder and decoder architecture based on RNN(LSTM & GNU). The Attention mechanism is implemented by preserving the LSTM encoder's intermediate output of the input sequence, then training a model to selectively learn these inputs and correlating the output sequences with the model output.

For text classification, the convolution model uses several different filters to extract features from the embedding and applies the max-over-time pooling to further produce more abstract features(Kim, 2014). The convolution procedure doesn't depend on front word processing as in the recurrent model. Thus it can be implemented in a parallel manner and able to take care of longer sequence compared to the recurrent model. (2017) proposes a hybrid model composed of gradient-boost and deep convolution model. Two convolution models are fed separately with news headline and body to generate corresponding features and then later taken in by a multi-layer perceptron(MLP) with the four classes as output. The text convolution doesn't work so well that they combine an XGBoost model to make the prediction. The XGBoost, which stands for extreme gradient-boost, is proposed by Chen et al.. It is a specific implementation of GradientBoost method, which is first introduced by (Friedman, 2001). XGBoost introduces the second-order gradients and advanced regularization, and can be trained in parallel or distributed manner. The XGBoost part is fed with few features extracted from preprocessed n-grams features, including the count features, TF-IDF features, SVD features,

word2vec features, and sentiment features. The final model ranks first in the competition with 9556.60 point, which is 82.02%.

A simple baseline model implemented with gradient-boost is provided along with the training and test dataset by Pomerleau (2017). By utilizing some handcrafted features, such as co-occurrences of the word in headline and news body, refuting and polarity words, it reaches a final score of 75.20% on the test dataset and 79.53% on development dataset.

Vaswani et al. proposed the transformer architecture that outperforms the previous reported best model in two translation task (Vaswani et al., 2017). The structure follows the common machine translation way with an encoder and decoder, however, using only attention mechanism. The multi-head self-attention mechanism allows a word to attend in different position of the sequence. And since the feed-forward neural network part is independent of other positions, it then makes it possible for parallelization. Later introduced BERT(Devlin et al., 2018) is then built using the encoder architecture of transformer. Devlin et al. employ a bunch of ideas from transfer learning(Howard and Ruder, 2018), semi-supervised sequence learning(Dai and Le, 2015), contextualized representation(Peters et al., 2018) and the language model of transformer(Radford et al., 2018) as well. By utilizing a delicate idea of the masked language model, the model is trained to be able to predict the masked words and provides a well pre-trained model. Offered large and base BERT models are first trained on massive text dataset to capture the text representation, which can be easily used later in the fine-tuning process for the specific task. The base version makes use of an encoder with 12 layers, 12 attention heads and 768 dimensions of representation. Nevertheless, the mask idea further allows it to be run in a more parallel manner.

3 Approach

Based on these great ideas discussed in the previous section, we set up six models for the stance detection task, including the basic bidirectional LSTM, the attention mechanism enabled one, the conditional encoding LSTM with attention(CEA), base BERT, the baseline gradient-boost with BERT feature, the gradient-boost with only BERT feature.

3.1 Preprocessing

A simple preprocessing is conducted to remove the irrelevant information. The rest part follows the common cleaning approach using a regular expression to remove non-alphanumeric characters.

The tokenization process is done separately in different models, as a special model like BERT has its manner of dealing with it. While the other models just simply go through the normal tokenization process.

BERT takes advantage of the WordPiece(Schuster and Nakajima, 2012) idea that breaks the word into subword units while doing tokenization. These basic English characters and subword units take good care of out of vocabulary(OOV) words. It also adds segment id encoding and position encoding along with the embedding, which it learns from the pre-trained stage.

3.2 Models

The LSTM models all conducted with the sequence length of 128 due to the limitation of computing capability.

The following BERT related models all use the same pre-trained uncased base version, which contains 12 layers, 12 self-attention heads and hidden states with 768 dimensions. The maximum text sequence length, after tokenization, it could take is 512.

3.2.1 Bidirectional LSTM

Comparing to the standard LSTM model introduced in (Schmidhuber and Hochreiter, 1997), Bidirectional LSTM has two networks, one access information in the forward direction and another access in reverse direction. The network has access to the past as well as future information and hence the output is generated from both the past and future context. Therefore, bidirectional LSTM has better model performance in comprehending contextual information when comparing to unidirectional LSTM, and it is used as the baseline of our experiments. The model structure is displayed in Figure1.

In this model, the text of the headline and body are concatenated first and then fed into the model, as illustrated in Figure 1.

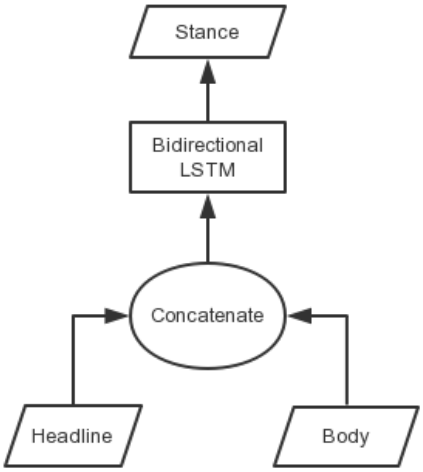

Figure 1: Bidirectional LSTM.

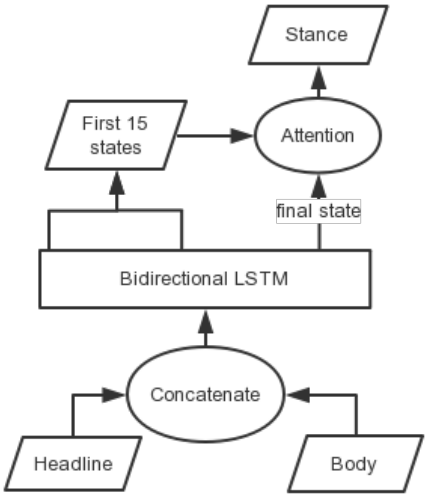

Figure 2: Bidirectional LSTM with attention.

3.2.2 Bidirectional LSTM with Attention

Attention

The attention mechanism presented by (Rocktäschel et al., 2015) was implemented in Tensorflow. In this paper, attention mechanism is realized with Keras framework by customizing an attention layer class. The brief explanation of attention mechanism is introduced below.

Let all vectors be column vectors. The attention window is defined as the first L output states produced by the LSTM. k is the dimension of the hidden state, and N is the total sequence length. Let $Y \in R^{k \times L} = [h_1, ...h_N]$ be a matrix of the output states of LSTM in the attention window. Let $e^L \in R^L$ be a vector of $1s$. Let $W^y, W^h, W^p, W^h \in R^{k \times k}$ and $w \in R^k$ be trainable matrices. A final state h^* is produced as follows:

$$M = \tanh(W^y Y + W^h h_N e_L^T)$$

$$\alpha = soft\max(w^T M)$$

$$r = Y\alpha^T$$

$$h^* = \tanh(W^p r + W^x h_N)$$

The model structure is displayed in Figure 2.

3.2.3 Conditional encoding LSTM with Attention

To further improve the model performance, we use a more complicated model name conditional encoding LSTM with attention(CEA), which is introduced by Pfohl(Pfohl et al., 2017). The model architecture is implemented in Keras framework. Within the model structure, the headline and body are processed separately in two basic LSTM models. The final hidden state of the LSTM model used to process the headline is harnessed to initialize the first hidden state of the model used to process the body, and that is the core of conditional encoding mechanism. Then the attention mechanism operates over the first 15 output states of the headline LSTM in conjunction with the final output state of the LSTM used to process the article bodies.

The model structure is displayed in Figure 3.

3.2.4 Base BERT

The base BERT model takes the input of a combined sequence of headline and body texts. It first combines the two texts with a [SEP] token and adds a [CLS] token at the beginning.

[CLS] Body [SEP] Headline

Then it pops out tokens from the longer texts, which is usually the body text, to make sure the sequence length meet the requirement. After tokenization, the model transforms the tokens into embedding.

The model takes care of the tokenization process. BERT utilizes WordPiece and divides words

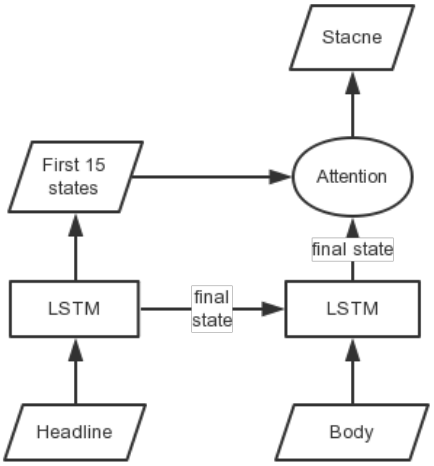

Figure 3: Conditional encoding LSTM with attention.

into subwords. The produced tokens are then first converted to token id using the pre-trained vocabulary. To further provide the input with relative position information, some learned position encoding and segment encoding are added to the mapped embedding vector as final input to the model.

After running the pre-trained model, the first token representation of the last layer, the representation of [CLS] token, is then taken as input to a feed-forward neural network, as the [CLS] token is designed to capture the information of input text sequence.

The model structure is displayed in Figure 4.

3.2.5 Baseline GradientBoost with BERT Feature

To explore the performance of the contextualized representation captured by BERT, the representation of [CLS] token of the last layer is extracted as a new feature. Along with the original features offered in the baseline GradientBoost model, they form up as new input to the model.

However, due to the limitation of computing capability, the maximum sequence length for extracting the new feature is set to only 128.

The model structure is displayed in Figure 5.

3.2.6 GradientBoost with BERT Feature

To provide another reference object, a simple GradientBoost with the extracted feature of BERT is setup. This model would follow the same settings of the previous model to see how powerful the

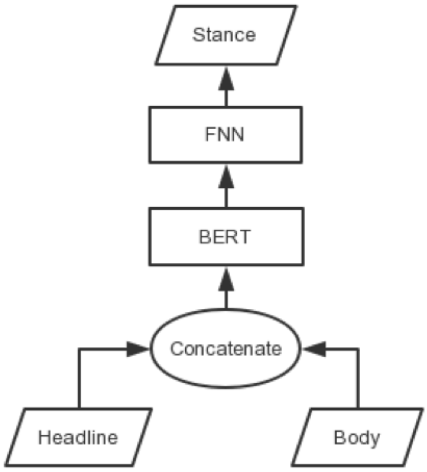

Figure 4: Base BERT model.

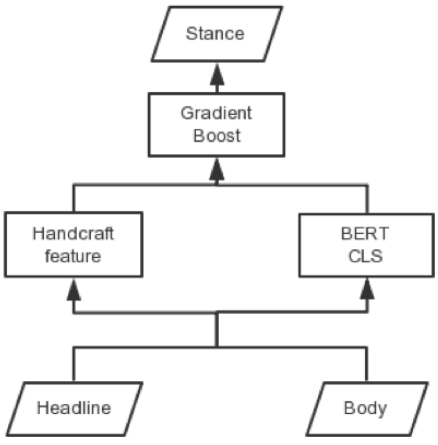

Figure 5: Baseline GradientBoost with BERT Feature.

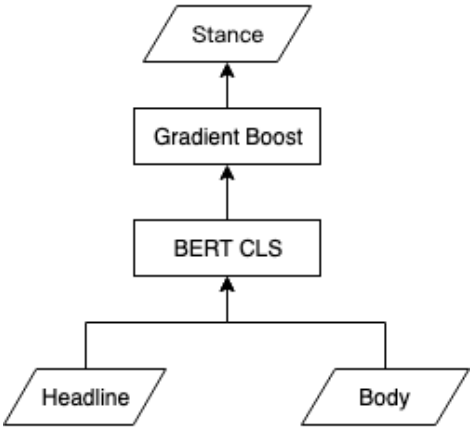

Figure 6: GradientBoost with BERT Feature only.

Stance	Count
Agree	3678
Disagree	840
Discuss	8909
Unrelated	36545

Table 1: Stance distribution in training dataset.

contextualized representation is.

The model structure is displayed in Figure 6.

4 Experiments

4.1 Dataset

Pomerleau and Rao provides the FNC dataset, including the training and testing dataset. The training dataset is formed up of 49972 records with only 1683 different body texts. Each record contains a news headline, a body text and a stance as a label.

As illustrated, the dataset is highly imbalanced with around 73% of the records labeled as Unrelated. What's worse is that the total records are 30 times the size of body texts. As a result, the model would tend to be overfitting easily. However, as presented later, model like BERT can greatly outperform the result of GradientBoost that are designed to overcome the problem of overfitting.

4.2 Metric

Since the task is classification, the F1 metric is employed to evaluate the performance of the model. The FNC also provides a scoring function based on some defined rules as a metric of competition. The detailed information about it is illustrated in Figure 7. We use the two metrics together to compare the performance of the different model.

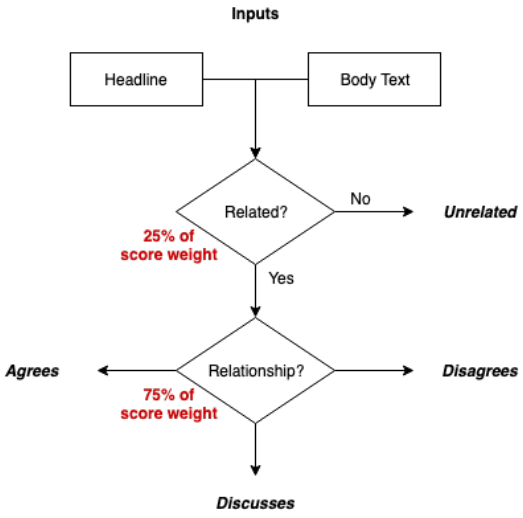

Figure 7: FNC scoring system.

4.3 Training

LSTM models are implemented in Keras framework and trained using Jupyter notebook. The unique tokens are fed to 50-dimensional GloVe embedding vectors that are pre-trained on the Twitter corpus. Since the sequences are of different length, we introduced zero paddings to make all sequences fit the maximum sequence length conserved in the corpus. For these three LSTM models, we set the default value for the number of hidden layer units at 100, and the dropout probability at 0.2. Besides, Adam optimizer was used for all models, and the parameter of embedding vectors are adjusted to non-trainable to avoid overfitting. The training process is run for 40 epochs, and the batch size was set at 128. For better comparison with BERT and saving training time, we set the sequence length at 128. As for complicated models with attention mechanism, the attention length was at a fixed value of 15.

BERT models are trained using Google CoLab, which supports TPU. Most hyperparameter settings follow the same set of the original fine-tuned MRPC example. The only parameter to be tuned is the sequence length. Since the length of 75% of the body text is over 400 after tokenization. We tune the model with sequence length 128, 256, and 512.

To extract or encode the text into features using BERT AS SERVICE(Xiao, 2018) would require a very long time and very large space if features of all layer are considered. Instead, to generate features for GradientBoost model, we only use

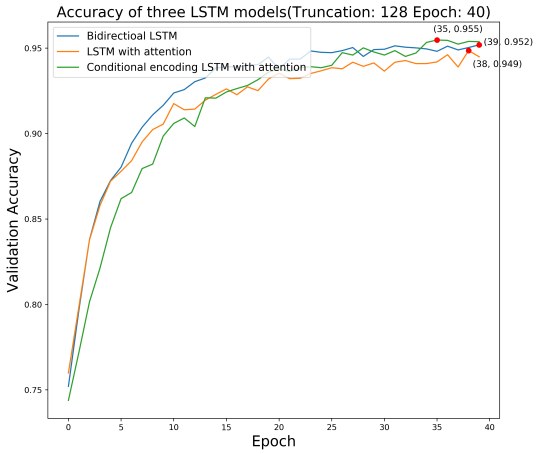

Figure 8: Training accuracy of three LSTM models with 128 input sequence length.

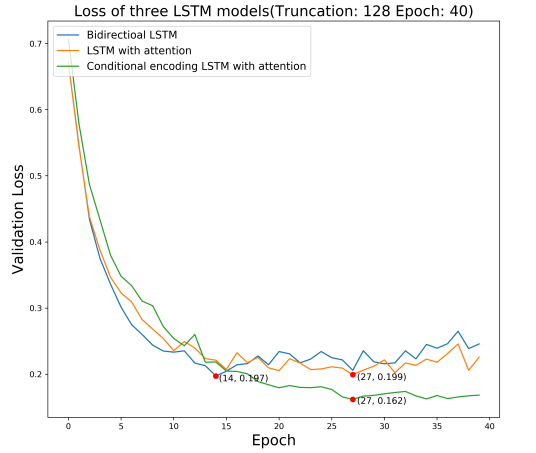

Figure 9: Training loss of three LSTM models with 128 input sequence length.

the representation of [CLS] token in the last layer with sequence length 128. Both GradientBoost models are then trained on the local machine with the same hyperparameter settings as the baseline model for comparison.

4.4 Result and Analysis

As shown in Figure 8, the BiLSTM model and the attention mechanism employed BiLSTM model tend to be overfitting after 14 epochs. By comparison, the one with conditional encoding works better and are not that easy to overfit. The training accuracy of these three models all gives a quite good result as shown in Figure 8, which are all easily over 90%. Nevertheless, the accuracy results on the training set are only around 70%.

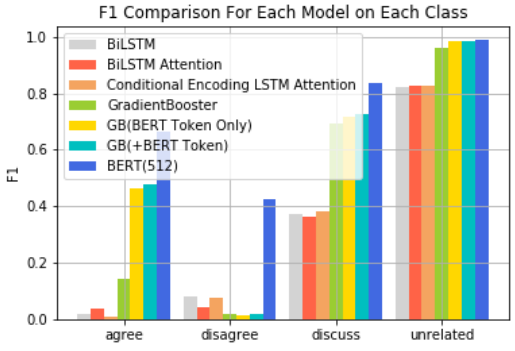

Figure 10: Comparison of F1 result of different model.

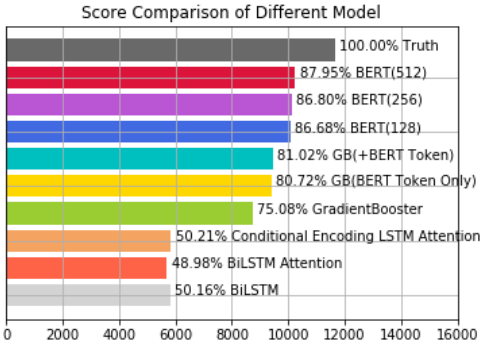

Figure 11: Comparison of FNC score result of different model.

The baseline GradientBoost with BERT feature and the one using only BERT feature both get result around 88% accuracy on the test dataset. While on the other hand, the best BERT model gets an accuracy result of 92%. On the training set, BERT models with different sequence length can easily get better accuracy results that are over 96% with less training epochs, comparing to LSTM models.

The F1 result and FNC score result comparison are presented in Figure 10 and Figure 11. The result of LSTM models is not working as expected. The three of them all perform worse compared to the given baseline model. On the other hand, models using BERT feature and pure BERT works better. The BERT model that using 512 sequence length works best and builds a state-of-the-art result with an FNC score point 10247.5(87.95%). And as expected, due to the highly imbalanced problem of the dataset, the F1 score on the classes including Agree, Disagree, Discuss are not desirable. However, as shown in Figure 10, the F1 re-

sult of BERT on the three classes outperforms the other models quite much.

The reason why the LSTM model fails to meet expectation is probably due to the sequence length. Due to the limitation of our computing resources, we won't be able to further increase the sequence length, as the recurrent structure would take much longer time if we set the sequence length the same as best BERT model. In facts, 75% of the length of body text after preprocessing still have over 400 tokens. And since CEA takes advantage of extracting useful information from noisy input when the sequence length is long enough. As a result, set the length to 128 could certainly lose quite much information. Even though, BERT still produces an FNC score of 86.68%, which is quite close to the best one using 512 sequence length.

By intuition, the surprisingly good performance shown by BERT is due to its capability in producing the contextualized text representation. That's also confirmed by the GradientBoost models. With the BERT generated features introduced, the model outperforms the given baseline by 6% on FNC score. Even if using only BERT generated features, the model still outperforms by over 5%. The difference that baseline features providing is quite small.

Another thing that BERT surpasses the recurrent model is that its ability in parallel computing. As stated in the previous section, the modified transformer architecture enables BERT to run all position of the same layer in parallel. The training process of an epoch of BERT models takes around only 10 minutes on the CoLab environment for the case of 512 sequence length. Normally, the length doesn't affect the required time of training since the original structure takes 512 tokens. And BERT also takes fewer epochs to converge. The best model takes only 5 epochs and the other two take about 3 epochs. Comparing to the cost of the recurrent structure, BERT generated feature does save much effort and also performs better.

5 Conclusion

In this work, we present a comparison between models of recurrent architecture and transformer architecture on the FNC stance detection task. The BERT model using the sequence length of 512 produces a state-of-the-art result with 10247.5/11651.25(87.95%) FNC score points.

Due to the limitation of computing capability, we are not able to further increase the training sequence length, which leads to the poor performance of recurrent models. However, yet with 128 sequence length, BERT outperforms all the recurrent models with over 30%. Other models using BERT extracted features also improve quite much. Therefore, it's easy to conclude that the deep transformer encoder structure could generate better contextualized text representation.

And there is still a bit more future work to be done to deal with the problem of the imbalanced dataset and to explore the recurrent model with longer sequence length, which shall lead to a better result.

References

Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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*.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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*.

Oskar Triebe Pfohl, Oskar Triebe, and Ferdinand Legros. 2017. Stance detection for the fake news challenge with attention and conditional encoding. *CS224n: Natural Language Processing with Deep Learning*.

Dean Pomerleau and Delip Rao. 2017. The fake news challenge: Exploring how artificial intelligence technologies could be leveraged to combat fake news.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Jürgen Schmidhuber and Sepp Hochreiter. 1997. Long short-term memory. *Neural Comput*, 9(8):1735–1780.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

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.

Han Xiao. 2018. bert-as-service. https://github.com/hanxiao/bert-as-service.

Sean Baird Yuxi Pan, Doug Sibley. 2017. Talos targets disinformation with fake news challenge victory.