FAKE NEWS CHALLENGE: STANCE PREDICTION USING DEEP NEURAL NETWORKS AND TRANSFORMER MODELS

Rishabh Karwayun, Atharva Gokhale

University of Waterloo

ABSTRACT

The primary goal of the Fake News Challenge (FNC) is to deploy Natural Language Processing (NLP) techniques and Machine Learning (ML) algorithms to overcome the issues arising due to fake news articles. Fake news are generally made up stories to deceive the the readers, possibly for secondary gains. Such a situation creates confusion about the facts of the current events. The rapid growth in generation and spread of fake news has possibly led to a misinformed public opinion resulting in chaos in few situations. Classifying the news articles as a hoax or a real story is a difficult task even for the trained experts on account of the large number of facts that needs to be considered in the process.

However, with the recent developments in the field of Artificial Intelligence we can rely on the different techniques which can automate the perspective of the experts used to classify a particular news article as a fake one. We can think of this issue as a text classification problem and deploy various ML techniques such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) [1] and state of the art transformer models [2]. Our primary objective is to surpass the performance of the baseline model which is implemented using the classical machine learning technique, Gradient Tree Boosting. This base line implementation could achieve 79% test accuracy. It was possible to break the baseline barrier and achieve 10542.50 FNC score using the RoBERTa model developed by Facebook.

Index Terms— FNC, Natural Language Processing, Neural Network, Transformers

1. INTRODUCTION

The increase in spread of fake news is a serious issue as it is responsible for influencing the readers in a wrong direction. According to the New York Times the fake news is defined as ”made up story with the intention to deceive, often with monetary gain as a motive”. Detecting the fake news is a very complex problem considering its different explication considering its statistical analysis [3]. However, this complex task can be broken down into stages, the first one being, determining what other news organizations are saying about the topic. This stage is also called as Stance Detection.

The training and testing data set provided in the FNC consists of two CSV files, the first one just has all the unique article bodies along with its respective body ID. The second file consists of stances which maps relation between a particular heading and a particular article body based on the body ID. The stances under consideration are unrelated, agree, disagree and discuss. The body-heading pair having an unrelated relation signifies that the heading has absolutely nothing to do with the article body and can be considered as the fake news. As the training data is heavily skewed on account of maximum data samples with unrelated as their stance, we have modified the data for better results.

Now, considering the FNC as a text classification problem there are multiple approaches to solve the problem efficiently. We started with the basic neural network, the Convolutional Neural Network (CNN) which achieved decent results, but less than the milestone implementation, however achieved best results of all other deep neural network implementations. Next obvious option was the Recurrent Neural Network which is specifically used for sequence of text proposed by [4]. We could observe that the performance of simple LSTM was way below the expectation considering the baseline accuracy and the FNC score. The Bi-directional LSTM proposed by [5], an efficient extension of LSTM . As a result, the bidirectional LSTM has best performance results as compared to the unidirectional LSTM model.

We have provided a detailed comparative analysis of the performance of CNN, unidirectional LSTM and bidirectional LSTM for the FNC train and test data set. We have explored the performance of state of the art transformer models such as Bidirectional Encoder Representations from Transformers (BERT) developed by Google [6] and its optimized and more efficient version RoBERTa developed by Facebook. We could beat the baseline accuracy by a significant margin using the pretrained RoBERTa model which achieved 93.91% test accuracy and FNC score of 10542.50.

2. RELATED WORKS

2.1. Baseline Implementation

The implementation [7] serves as baseline test accuracy for the further implementations. The approach involves three major steps to classify the test data correctly. The first step is related to data preprocessing. In this step, all the characters other than alphanumeric characters are removed. Later, all the words are converted to lower case. Then these lower cased words are tokenized, converted into list of string in order to perform further preprocessing. Next step is to lemmatizing the data. Finally stopwords are removed.

In the next step, some important features are extracted from both, the headings and article body section of the training data. The features includes, overlapped feature (Overlap between headline and body), Refuting features (Refuting words in headlines), Polarity features, Hand features(Counting the N-grams in headline in the article body) and cosine similarity. Finally,Gradient Tree Boosting algorithm is used to train the model and make predictions on the test set. This baseline implementation achieves a score of 8748.75. Our primary goal is to surpass this benchmark using any other machine learning implementation.

2.2. Deep Neural Networks

We have implemented the basic two deep neural network models, Convolutional Neural Network and Recurrent Neural Network in our approach [8]. Our first approach to beat the baseline implementation was using a deep convolutional neural network which have achieved promising results for few fake news data sets proposed by [9] [10]. Also, according to [11] CNN performs better for extracting local and position-invariant features from text data which makes it a potential deep learning model for the FNC. Recurrent Neural Networks are explicitly used for sequential data (text, voice, time series prediction). According to [11] LSTM model achieves promising accuracy for fake news detection. However, for our dataset, a simple LSTM model achieved significantly low FNC score and test accuracy as compared to the baseline implementation. Using a bidirectional LSTM was the next appropriate step to increase the score considering the research approach proposed by [12]. We could significantly increase the FNC score using a stacked bidirectional LSTM model, which ultimately achieved best FNC score for the deep learning models. It is further possible to increase the score using the deep learning techniques itself by incorporating data specific features as used in the project milestone implementation. However, this approach still lacks in performance considering the FNC score achieves using the state of art transformer models. The transformer models are rigorously trained on a very large text dataset which enables it to construct highly efficient word embeddings.

2.3. Transformer Models

2.3.1. Transformers and BERT

Transformers are a relatively recent development in the field of Natural Language Processing (NLP). Introduced in 2017 [13], Transformers have proven to been quite influential in numerous NLP applications. The main advantage that they have over methods like RNN, LSTMs and GRUs is that in theory, and given sufficient computing resources, their attention mechanism can have an infinite reference window. Another major advantage of Transformers over above architectures is that the above architectures need the data to be fed in serially, and thus require substantial training time [14]. Due to this serial nature, they fail to utilize the power of the GPUs. On the other hand in Transformers, the encoding can be done in parallel, and hence they can use GPUs for faster training. Transformers contain Encoder-Decoder sub-parts which also have applications when used independently. While Encoders can learn about language, grammar, context; Decoders on the other hand in the example of Machine Language Translation, learn about the mapping between the first and the second language. OpenAI's GPT is an example of Decoders stacked together, where as Google's BERT is an example of stacked Encoders [15].

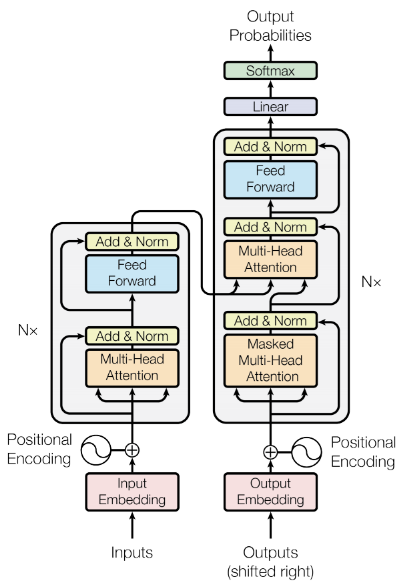

Fig. 1. Architecture of Transformer[13].

The interesting aspect of Transformer encoding is that they also add a positional information into the embedding. It is done because encoder in a Transformer has no recurrence like in the RNNs. For each word, Positional Encoding is calculated using the sine (for words at even positions) and cosine (for words at odd positions) functions. These Positional Encodings are then added to the corresponding embedding vector. By doing this, position of a word in a sentence is also taken into account. Sine and Cosine functions were chosen as they have linear properties that the model can easily learn.

Bidirectional Encoder Representation from Transformer (or BERT) [2], developed by Google in 2019, is an example of stacked Transformer Encoders. Training of BERT happens in 2 phases: In the first phase called **Pre-training**, BERT was trained over the English Wikipedia (2,500M Words) and BooksCorpus (800 Million Words). In this phase, the model learns about the language and context. In next phase, called **Fine tuning**, the model is fine tuned on data specific to the task at hand. Due to the amount of data pre-training is performed on, and the manner in which it is done, BERT is able to learn deep insights about the language [16] [17].

While pretraining BERT, two tasks are performed [18]. First is called **Masked Language Modelling (MLM)**. In MLM, 15% of the words in a sequence are randomly 'masked'. It follows a complex logic where 80% of the time masked words are replaced with a [MASK] token, 10% of the time they are replaced with random words and 10% of the time they are left as it is. This is done to prevent the model from over-fitting on a particular token or a position in a sequence. The model then tries to predict the masked words, based on the context provided by rest of the words in the sequence. Unlike other RNN techniques that find context in a single direction, MLM helps BERT in learning in a true bi-directional context. This learning is even better than Bi-directional LSTMs as they too are nothing but stacking of LSTM's that learn individually in a single direction. Context learnt by BERT in this manner is truly bi-directional. Second task preformed in pretraining is **Next Sentence Prediction (NSP)**. Model is given a pair of sentences, X and Y. It has to predict whether Y actually appears after X in the data or not. For this task, 50% of the samples given to the model have X and Y as consecutive sentences and in rest Y is a random sentence that does not appear after X.

For fine tuning BERT, it is then trained on data specific to the task. Adding a small layer on top of the core BERT model can result in using BERT for different tasks. In October 2019, Google announced application of BERT on English Language queries in the United States [19]. As of December 2019, BERT has been adopted by Google Search for over 70 languages [6].

2.3.2. RoBERT

Robustly Optimized BERT approach (RoBERTa)[20] is an improved version of BERT developed by Facebook. RoBERTa introduced dynamic-masking, which means that the mask token is changed during epochs. It also does away with the Next Sentence Prediction task otherwise performed during BERT pretraining. Pretraining of RoBERTa is performed on 160GB of textual data. This includes the data used for pretraining BERT, which is around 16GB. Apart from this, pretraining data includes Web text corpus, CommonCrawl News data set and Stories for Common Crawl. All in all, pretraining of RoBERTa would take approximately four to five times longer as compared to BERT. Larger batch sizes and learning rates were used for pretraining RoBERTa. The combination of larger batch sizes, larger pretraining data results in a much better model than BERT. It performs best on almost all of the GLUE tasks[20]. RoBERTa is the current state of the art model [16] and we wanted to gauge its performance on FNC challenge[21].

3. APPROACH

We have studied various classical machine learning, deep neural networks and transformer model approaches to achieve maximum score in the fake news challenge. We have fine-tuned the hyper-parameters of few already existing models to achieve better results. We could successfully surpass the baseline implementation score using the RoBERTa model proposed by [20]. Considering the deep learning techniques, maximum FNC score was achieved using the Convolutional Neural Network and the stacked bidirectional LSTM model jointly (minimal variation). We have considered the F-1 score as our primary evaluation metrics which clarifies which target class is classified erroneously by a particular classification model.

3.1. Word Embeddings (word2vec, Glove and Fasttext

Creating word embeddings is a primary task in any language related problem. The process helps us to convert the words in our dataset into numbers in a form of a vector representation. The vector representation of the words allows the words with similar meaning to be understood by machine learning algorithms. Initially, we implement the word2vec neural network which generated the word embeddings on our train dataset. But, as we can observe, the training dataset is very small and hence the embeddings created are not capable of efficient vectorization of the test data. Thus, we decided to use the pretrained embedding models to vectorize the data. These embeddings play a vital role in training the neural network as these vectors are later used as the weight matrix in the embedding layer of the model. We have deployed two state of art pretrained embeddings models such as Glove50d [22] and Fasttext300d [23] developed by Stanford and Facebook respectively. There exists higher dimensional embeddings of the same model, yet we has to restrict ourselves considering the available computing resources.

3.2. Preprocessing for Neural Networks

Certain preprocessing steps are very necessary and are commonly used in all the executed neural networks in this paper. We get the vectorized data after the basic data preprocessing which needs to be reshaped in order to make it compatible as a input the neural network. This first step we need to pad each sequence of text data so as to maintain a uniform size of each input sample. We have truncated the extensive words in text sequences with longer than 128 words. In the concatenated model where we have a different neural network for the headings and article bodies, we could define a different maximum length of the sequences depending upon the length of the longest sample in headings and article body respectively. The target labels are also transformed to the categorical state in order to get the probabilistic output of the predicted labels for the test data. For the weight matrix of the embedding layer, we use the word embeddings generated by the pretrained model (Glove / Fasttext).

3.3. Convolutional Neural Networks

Convolutional neural network (CNN) is a variant of artificial neural network which uses perceptrons for cognitive tasks like language processing. They are regularized version of Multi-layer perceptron (MLP) where each neuron in a particular layer is connected to all neurons in the next layer [24]. In this model the primary task is to create the word embeddings which creates a word vector for every unique word in the vocabulary. However, just using the words from the training data the vocabulary formed is very limited and the model under-performs for the test dataset as there will be many words that are not in the train vocabulary. We implemented the 1-D Convolutional Neural network as our preliminary model along with the Glove 50d pretrained embeddings [22].

The convolutional layers extract the features from the training data. The filters in the convolutional layers can establish the relation between two adjacent words, the frequency of a particular pair of words repeating, relation between two words with a word in between. At each step, the filter value is multiplied with the word embedding values where the filter values of 1 times the word embedding values result in the word embedding values, while filter values of 0 result in 0. In such a way the filters deduce the relation between the words in the text under consideration. We have stacked the convolutional layers where the subsequent layers can extract more complex information from the training dataset. Later, we use a max pooling layer which iterates over the tensors and calculated the highest value for a given filter dimension. Thus the feature space is compressed. Finally, we can ensure that important features are retained, while empty spaces are discarded.

Finally, we have the full connected layers with a particular activation function depending upon its position in the model. These layers are basically responsible for classifying the data into the available target classes. We have used the categorical cross-entropy loss which is the best option for the cases with multiple target classes. The activation function used for the dense layer is 'ReLu' which performs better for sparse data like the one in FNC. We have used the Softmax activation function in the output layer which is preferred in case of multiple target classes. The optimizer used for the CNN is ADAM which is the state of art optimizer provided by the Keras API and also overcomes the drawbacks of all its predecessors.

3.4. Recurrent Neural Networks (LSTM, Bidirectional LSTM)

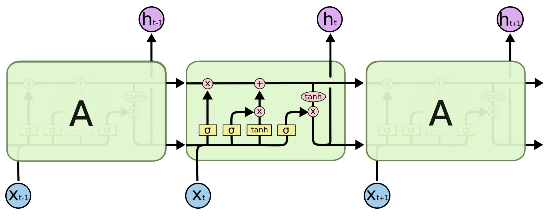

Fig. 2. Repeating Module in LSTM[25].

We have executed two variants of LSTM model, the standard LSTM model proposed by [4] and its successor, Bi-directional LSTM model proposed by [5]. The bidirectional LSTM model has two networks, the first one access the text data in forward direction while the other network access the data in reverse direction. Thus, we can say that the network has access to the past and future data while training the model. Our LSTM is basically a many to many classifier where we feed in a list of multiple words at a time and predict to which target class amongst the 4 a particular input belongs. LSTM-RNN model is preferred for a long-range semantic dependency based classification problems [12]. LSTM model is a of RNN competent in learning long-term dependencies. We have executed all the variations of LSTM models for limited number of epochs as they take significantly large amount of time to train the model as it uses sequential processing. Additionally, the LSTM models consume large amount of memory again on account of its sequential processing. Thus, the test accuracy and the FNC score for RNN models is on the lower side and can be improved by increasing the number of epochs.

In these models we have concatenated the headings and the article body text data initially before feeding to the neural network. The embedding layer in the neural network contains the weight matrix with word embeddings obtained using the pre-trained 50-dimensional Glove embeddings. Next, We have used a single bidirectional hidden layer with 120 hidden neurons in one case and stacked activation layers after the Bidirectional LSTM layer in the second case. We have used the dropout layer in both the cases to avoid the model, overfitting on the train data. Finally, we have a dense layer with 4 neurons equal to the number of target classes with softmax activation function which is preferred for probabilistic output. We have used the ADAM optimizer for all the deep neural network models. The stacked bidirectional LSTM network achieves slightly better test accuracy but takes significantly greater amount of time to train the network as compared to its single layer variant.

3.5. Concatenated LSTM Model

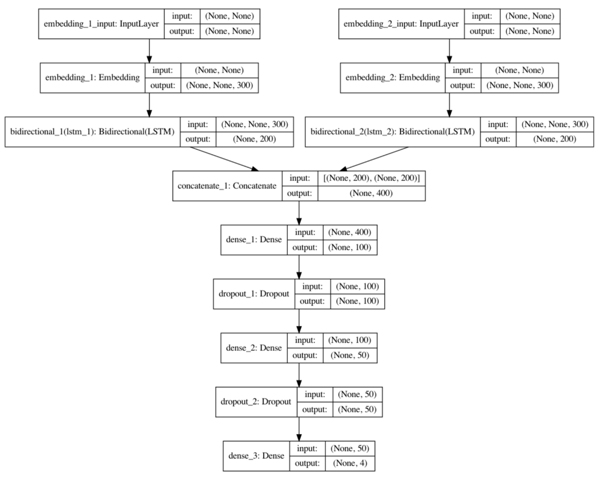

Fig. 3. Model Architecture of the Bi-directional LSTM Concatenated Model with Fasttext Embeddings [23]

The basic motivation for this model is to separately extract the features of headings and article body respectively using separate networks. This is again a bidirectional LSTM model where we have maintained separate bidirectional LSTM network for headings and article body each. This enables the model to extract the features separately from the two section of the dataset and later concatenate them to establish a relation between them in order to predict the stances. Each LSTM model includes 120 hidden neurons. Later, we concatenate the two networks as shown in the figure 8. Finally we add multiple dense layers followed by dropout layer to avoid overfitting. Again, the output layer is a dense layer with 4 neurons to predict the target class for the input data features. For this particular model we have used the 300-d fasttext word embeddings [23]. As the fasttext model is trained on significantly larger dataset the model is trained more efficiently and performs better on test data. The model uses similar set of hyper-parameters and optimizer to compile the model. We could achieve maximum test accuracy of all the deep learning model mentioned above. However, the training time for this model is maximum on account of the large pretrained word embeddings (300-d fasttext) and multiple bidirectional LSTM layers in the model.

3.6. Transformer based Models

We have used the Simple Transformer [26] implementation of Transformer based models. Simple Transformer library provides right out of the box implementations of models for Text Classification, Named Entity Recognition, Question Answering, amongst others. All these models contain pretrained Transformers that can then be fine tuned for the specific problem.

For BERT, the *base uncased* model of BERT was used. It has 12 layers, 12 attention heads and a total of 110M parameters. For RoBERTa, *base model* of RoBERTa was used. It is an optimized version of BERT's base model with a total of 125M parameters. It provides a final classification layer on top of embeddings, which can be used as a classifier right out of the box. For training, it takes in 2 features (headline and article body) and labels. BERT model was trained with max sequence length of 512 and 256, where as RoBERTa was trained with max sequence length of 512 and 256, with also varying the learning rates [27]. All the transformer models were trained for 5 epochs.

4. RESULTS AND ANALYSIS

4.1. Scoring

FNC challenge[28] provides an algorithm to calculate score based on predictions done by the model. The algorithm for calculating score has been described in the flowchart. A script is also provided that can calculate F1 scores based on this algorithm. The script has been used to calculate F1 scores for the considered models.

4.2. Data Preparation

The training data has 49,972 data points. There are 1,648 unique headlines and 1,683 unique article bodies.

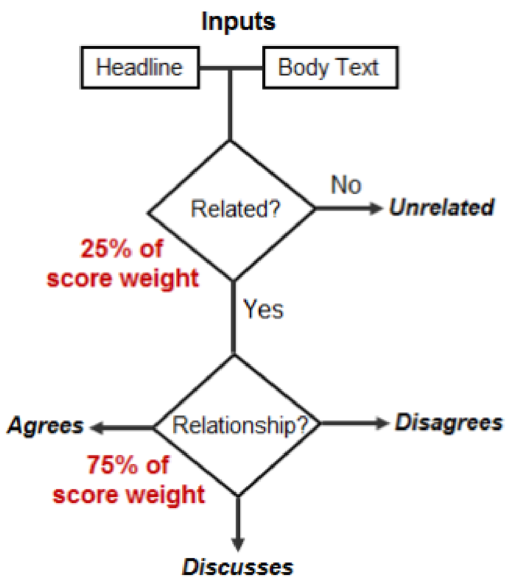

Fig. 4. Score Calculation Flowchart[28].

Each sample is a headline-article body pair that is categorized into one of four classes: 'agree', 'disagree', 'discuss' and 'unrelated'. If the class is either 'agree', 'disagree' or 'discuss', then the headline-body pair can be said to be 'related'. Out of these, 36,545 data points are labeled 'unrelated'. This means around 73% of the data belongs to a single class. Thus, we have biased data. Using such a biased data to train may result in the model being a binary classifier acting as a decider between the majority class and the rest. This was indeed observed when BERT was trained on this data.

To overcome this problem, a more balanced subset of the data was prepared. It was observed that in the training data, 13,427 sampled were from rest of the three classes. Data was grouped by headlines and subsequently by the stance. For each headline, all the samples that belonged to either 'agree', 'disagree' and 'discuss' were chosen, and at most equal number of 'unrelated' samples were chosen. Thus, our final data had 26,854 samples where half of the samples were labeled 'unrelated' and rest belonged to the three classes. Though this data set is still not perfectly balanced, the bias is much less than in the original data. Also, any further attempts to balance the data set would have resulted in a much smaller training space which again, is not desirable. Hence, data set prepared in the above manner was chosen as the training data.

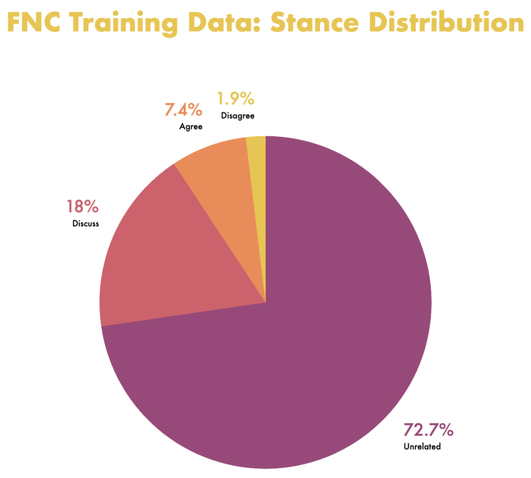

Fig. 5. Training Data: Stance Distribution[29].

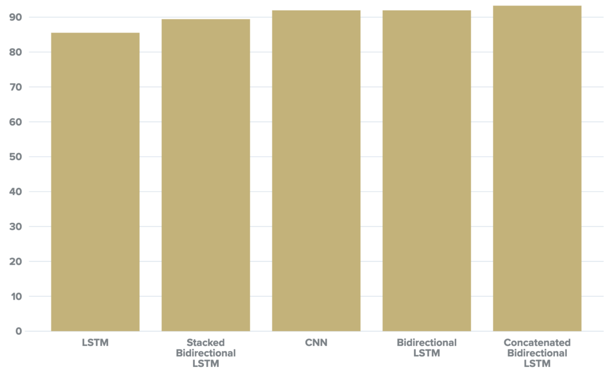

Fig. 6. Deep Neural Networks: Validation Accuracy [29].

4.3. Result Analysis

All the deep learning networks were trained using Adam Optimizer and with Categorical Cross Entropy loss function. Due to the reduction in training data, it was proposed to train the models on more epochs. However, it was observed that the models would tend to over fit on the training data as evident by the validation accuracy which is randomly selected from the training data itself for every epoch.

Sr. No.	Models	Training Time per epoch(sec)
1	CNN	51
2	LSTM	148
3	Bidirectional LSTM	180
4	Stacked Bidirectional LSTM	179
5	CNN with Fasttext 300-d	100
6	Concatenated Bidirectional LSTM	741

Table 1. Training time for Deep Learning Models.

Amongst the Deep Learning models, training time is least for CNNs. This is due to the fact that CNNs are able to utilize GPUs for parallel processing and faster training. Training time increase for RNN based models as they input data in sequential models. Concatenated Bi-directional LSTM took the longest to train. Training times are subject to change in computing hardware. We used Google Colab for our training.

Conventional Neural Network models failed to cross the baseline score. This is possible as they are only extracting general features from the text. Baseline model on the other hand uses specific features like overlaps, refuting words, polarity etc. which are providing more information than the features extracted by the Deep Neural Networks on the data. Another reason behind this can be explained by the embeddings. Conventional Neural Network models are using Glove and Fasttext embeddings. They suffer from two major problems according to [30], first is that they use very shallow language models, and hence have a limit to the amount of information that can be captured. The second one is that such models are not capable of truly understanding the context of a word. Hence, these embeddings already put a limit on the amount of information extracted.

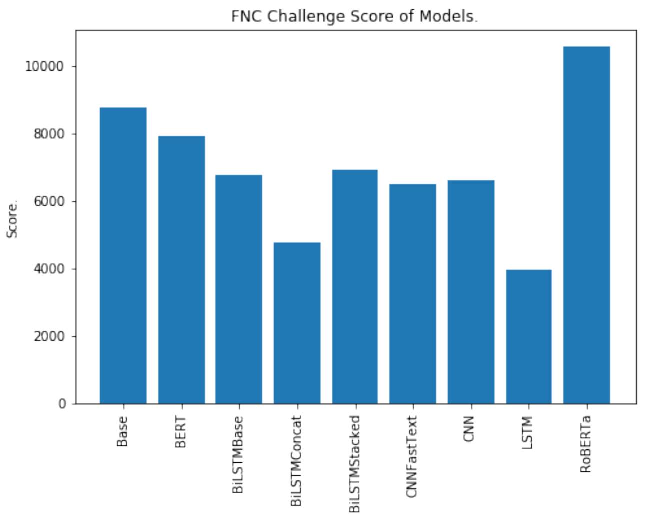

Fig. 7. FNC Challenge Score.

Transformer based architectures performed much better than conventional neural networks. BERT performed better than conventional neural network models. RoBERTa model on the other hand, performed best amongst all the models considered. It scored a high test accuracy of 93.91% and a score of 10542.50 which is quite impressive. RoBERTa performed best in identifying all four labels. Its stellar performance can be attributed to it's architecture and specially pretraining. Due to its pretraining which enables it to have a deep

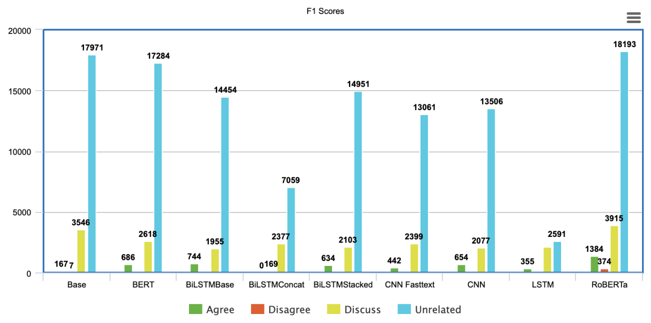

Fig. 8. F1 Score.

understanding of the English language and grammar, as well as about context of words in a sentence. When it is fine tuned on the FNC dataset, it is able to find patterns based on its previous knowledge which helps it in proper classification of the relationship between headline and article body. Best results were obtained on a sequence length of 512 and a learning rate of 1e-5, when trained over five epochs.

5. CONCLUSION

Different architectures were tried for the FNC problem and their performances were observed. Transformer based architectures performed much better than conventional neural network based architectures. RoBERTa performed the best with a score of 10542.50 and a test accuracy of 93.91%.

In our analysis, we have used straight out of the box implementations of Text Classifiers using BERT and RoBERTa encodings. An avenue for further research would be using BERT and/or RoBERTa encodings with different customized models. They can also be used with other handcrafted features for the baseline model and evaluating its performance. ELMO and ULMFiT embeddings can also be used for Deep Learning models as an option.

Github Link:
https://github.com/rkarwayun/MSCI-641-Final-Project

6. REFERENCES

[1] P. Bahad, P. Saxena, and R. Kamal, “Fake news detection using bi-directional lstm-recurrent neural network,” *Procedia Computer Science*, vol. 165, pp. 74–82, 02 2020.

[2] M. Qazi, M. U. S. Khan, and M. Ali, “Detection of fake news using transformer model,” in *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pp. 1–6, 2020.

[3] S. Singhania, N. Fernandez, and S. Rao, “3han: A deep neural network for fake news detection,” 11 2017.

[4] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] M. Schuster and K. Paliwal, “Bidirectional recurrent neural networks,” *Trans. Sig. Proc.*, vol. 45, pp. 2673–2681, Nov. 1997.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018. cite arxiv:1810.04805Comment: 13 pages.

[7] FakeNewsChallenge, “Fakenewschallenge/fnc-1-baseline.”

[8] A. Agarwal, M. Mittal, A. Pathak, and L. Goyal, “Fake news detection using a blend of neural networks: An application of deep learning,” *SN Computer Science*, vol. 1, 05 2020.

[9] R. Kaliyar, A. Goswami, P. Narang, and S. Sinha, “Fndnet- a deep convolutional neural network for fake news detection,” *Cognitive Systems Research*, vol. 61, 06 2020.

[10] Y. Yang, L. Zheng, Z. Jiawei, Q. Cui, Z. Li, and P. Yu, “Ti-cnn: Convolutional neural networks for fake news detection,” 06 2018.

[11] M. d. Koning, “How to build a recurrent neural network to detect fake news,” Feb 2020.

[12] P. Bahad, P. Saxena, and R. Kamal, “Fake news detection using bi-directional lstm-recurrent neural network,” *Procedia Computer Science*, vol. 165, pp. 74–82, 02 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[14] M. Allard, “What is a transformer?,” Mar 2020.

[15] P. J. S. at Analytics Vidhya with multidisciplinary academic background. Experienced in machine learning, “Transformers in nlp: State-of-the-art-models,” Jun 2020.

[16] P. Suleiman Khan, “Bert, roberta, distilbert, xlnet-which one to use?,” Oct 2019.

[17] R. Horev, “Bert explained: State of the art language model for nlp,” Nov 2018.

[18] M. S. Z. R. computer science graduate, “What is bert: Bert for text classification,” Jun 2020.

[19] “Bert (language model),” Jul 2020.

[20] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Ro{bert}a: A robustly optimized {bert} pre-training approach,” 2020.

[21] V. Slovikovskaya, “Transfer learning from transformers to fake news challenge stance detection (fnc-1) task,” 2019.

[22] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation.,” in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[23] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, “Fasttext.zip: Compressing text classification models,” 2016. cite arxiv:1612.03651Comment: Submitted to ICLR 2017.

[24] A. Mohnot, “Using nlp to detect fake news,” Mar 2020.

[25] “Understanding lstm networks.”

[26] T. Rajapakse, “https://simpletransformers.ai/.”

[27] “hugging face – on a mission to solve nlp, one commit at a time.”

[28] “Fake news challenge stage 1 (fnc-i): Stance detection.”

[29] “Make social graphics, short videos, and web pages to stand out-in minutes: Adobe spark.”

[30] M. S. Z. R. computer science graduate, “What is bert: Bert for text classification,” Jun 2020.