Fake News Stance Detection
MSCI 598 Final Project

Nino Spasik: 20720701
Cem Gunay: 20726091

Abstract

In this modern age of technology where nearly everyone has an internet connection, we often find ourselves victims of fake news. This fake news can come from a plethora of different sources, such as social media, news websites, forums and chats. The distribution of fake news has a negative impact on our society as it can spread around false facts, claims or accusations. Which can hurt various people, organizations or groups. The Fake News Challenge (FNC-1) is a worldwide challenge that encourages people to build models in order to learn the headlines and body of a corpus of news articles and classify them into one of the 4 following categories. These categories are called stances, which can be "agrees", "disagrees", "unrelated" or "discusses". The challenge includes a baseline approach that has an weighted accuracy of 75.08% and a weighted score of 8748.75 (out of 11651.25). The training data set for this challenge is a corpus of news with pairs of headline and body text with the correct stance. The testing data set is pairs of headline and body text without stance labels, as we will be using our model to predict the stances. This paper will explore our approach to create a model that predicts the stance of an article in an attempt to beat the score from the baseline. The data preprocessing, model architecture and testing methods will be outlined in this report for the 2 models that were created. The best model formulated was the TF-IDF Vectorizer combined with Word2Vec Embedding fed through a Feed Forward Neural Network, which had an accuracy of 65.64% on the competition dataset.

1 Background/Related Work

The FNC-1 Fake News Challenge has been released for several years now, meaning that many people have had a chance to build a model that best predicts the stances of headline and body pairs. The purpose of this challenge is to improve the accuracy from the baseline model. The baseline model provided in this challenge uses a gradient boosting classifier. A gradient boosting classifier is an additive model in a forwards pass stage manner, allowing it to optimize the loss function at each stage. For each stage, there are regression trees that are fitted on a negative gradient of the binomial deviance of the loss function. It combines this with k-fold cross validation. In addition, it uses the following features: polarity features, hand features and word overlap. All these features in combination compute the number of popular words and common n-grams for each headline body pairing.

While conducting research for this challenge, 2 models were found that related to our task. The first one is a model that was created for this challenge, titled SOLAT IN THE SWEN.[1] This model applies a 1-dimensional convolutional neural network on the headline and body pairings. These pairings are converted into pretrained vectors, then outputted from the CNNs to be sent into a multi-layer perceptron model with the 4 stances output. This model was regulated using a dropout rate of 0.5 for all convolutional neural networks. The architecture of this model was word embedding, CNN layers, dense layers and output layers. The second model explored was not related to the challenge, but the task it was modeling is similar to FNC-1. This model was created to classify tweets that were directed to a target into the 3 following classes: "positive", "negative" and "neutral".[2] A target could be a person, organization or group. This

model uses LSTM encoding that builds out a representation of a tweet that is dependent on who or what the tweet is targeted towards. This paper argues that embedding a tweet dependent on the target of the tweet is more effective than encoding the tweet and target separately. In addition, the authors of this paper were able to further improve their model by augmenting the model with bidirectional encoding.

2 Approach

2.1 Model 1: TF-IDF Vectorizer Combined with Word2Vec Embedding, Fed Through a Feed Forward Neural Network

The TF-IDF Vectorizer and Word2Vec Embeddings are passed through the input layer of the model as features. TF-IDF stands for Term-Frequency-Inverse Document Frequency. This measure quantifies the relevancy of a word for a document from a collection of documents.[3] In our case, a document would be a headline and body, and the collection of documents would be the corpus of the training dataset given to us for this project. This measure works by multiplying the number of times a word occurs within a body or headline, by the number of times a word occurs in the corpus for the headline and body. This number gives us a measure of the weight a word has in comparison to the rest of the words within the headline or body in the training corpus. The TF-IDF measure for a word is calculated using the following equations:

1. $Tf\ idf(t,d,D) = tf(t,d) * tf(t,D)$
2. $tf(t,d) = \log(1 + freq(t,d))$
3. $idf(t,D) = \log(N/count(d \in D:t \in d))$

Where t represents a word, d represents a headline or body, and D represents the training corpus collection of all headlines and bodies.

Word2Vec embedding simply groups the vectors of similar words together through a vector space.[4] Word2Vec transforms a word into a vector that comes in the form of a numerical representation. The architecture of this model is as follows:

1. Feature 1: TF-IDF Vectorizer embeddings on the headline and body
2. Feature 2: Word2Vec Embeddings on the headline and body
3. The two features are sent to a Feed Forward Neural Network
4. Passed through "ReLU" activation layer
5. Passed through dropout layer
6. Passed through output dense layer consisting of 4 classes, each representing one stance

2.2 Model 2: BERT Embedding Matrix Fed Through Bidirectional LSTM

The hidden embedding matrix outputted by the BERT model is used as the features for the input layer of this model. BERT creates a hidden state vector of already established size that corresponds to each word in a headline or body from the training corpus.[5] The architecture of this model is as follows:

1. Feature 1: Word embedding matrix created from the BERT model
2. This feature is sent to the Bidirectional LSTM layer
3. Passed through dropout layer
4. Passed through the dense output layer with a softmax function, consisting of 4 classes, each representing one stance

3 Experiments

3.1 Dataset

The training dataset provided from the FNC-1 challenge consists of 49972 rows. The columns in the dataset are Headline, Body and Stance. The distributions of the stances are as follows:

Stance	Count
Agree	3678
Disagree	840
Discuss	8909
Unrelated	36545

Table 1: Training Dataset Stances Distribution

We can see that most of the stances are unrelated, suggesting many news articles within the training dataset have bodies and headlines that are unrelated to each other. Suggesting the abundance of fake news within the dataset. In the following figure blue represents "unrelated", orange represents "agree", green represents "disagree" and red represents "discuss".

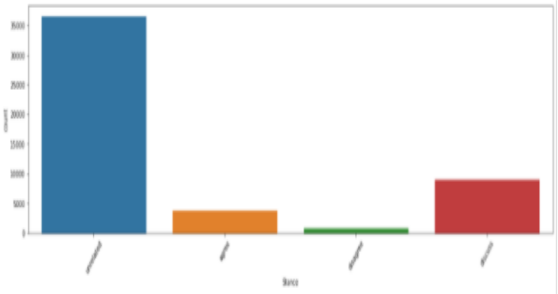

Figure 1: Visualization of Training Dataset Stances Distribution

The max length of a headline is 40 words. Where the average headline length was 11.12 words. Below we can visualize the distribution of headline length. We can see than most headlines lie within the 7 to 13 range.

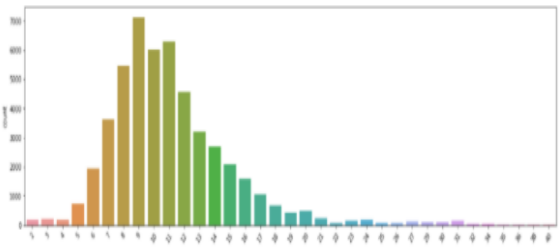

Figure 2: Visualization of Training Dataset Headline Length Distribution

4 Running The Experiment

4.1 Data Preprocessing

For the optimal model (Model 1), various data preprocessing steps were done to both the headline and body data. First, the text within this data was cleaned up by converting all text to lower case text. In addition, punctuation, numbers and special characters were removed. The cleaned-up data was sent to both respective embedding functions (TF-IDF and Word2Vec). Secondly, the text was tokenized. The process of tokenization involves splitting up a text into singular words, called tokens. An identifier is created for each unique token and stored in the code.

4.2 Model Configurations

The optimal model had 6 different parameters that needed tuning. These parameters are number of epochs, hidden units, batch size, type of activation function in the dense layer, dropout rate and the learning rate of Adam algorithm optimizer. The optimal values for the parameters are as follows:

Epochs: 20
Hidden Units: 100
Batch Size: 128
Activation Function(s): ReLU, Softmax
Dropout Rate: 0.2
Learning Rate of Adam Optimizer: 0.001

The batch size was increased from 64 with multiples of 8.

The “tanh” activation for bidirectional LSTM and “ReLU” for the feed forward neural network activation function was used. Dropout layers were also added to avoid over-fitting. The full architecture can be found in Figure A.1.

4.3 Data Directory Configurations

There were 4 configurations needed to optimize for the data embedding for the input layer of the model. These configurations were max length size, max vocabulary size, LSTM dimension and embedding dimension. The optimal values of the parameters are as follows:

Max Length Size: 150
Max Vocabulary Size: 40000
LSTM Dimension: 256
Embedding Dimension: 300

4.4 Training

The LSTM models are implemented in the Keras framework and trained in Google Colab. Google Colab support GPU and TPU computation and due to the heavy nature of the models, model 1 was run on the GPU and model 2 was run on the TPU since they are both substantially faster than CPU. The google news vectorized corpus was used with pre-trained vectors to create a vector representation of the tokens. Zero padding was introduced to make all the sequences of body and headline fit in the max sequence length conserved in the corpus. The LSTM models had the number of hidden layer units set to 100. The extracted features from headline and body will then have the similarity between the features attained along with

the similarity attained from the features sent through the feed-forward neural network using TF-IDF. The final extracted features from both the TF-IDF and LSTM are concatenated into a layer and fed through the neural network with 100 hidden units and SoftMax activation function to produce the final prediction probabilities. The training for the BERT model was made possible with the use of SpaCy transformers and their "en_trf_bertbaseuncased_lg" pretrained transformer model.

4.5 Learning Rate

The model learned the data through the process of 20 epochs. The average epoch run time was 863.7 seconds, which is equivalent to 14.39 minutes. The epoch run times, loss, accuracy, validation loss and validation accuracy were as follows:

Epoch	Time (s)	Loss	Accuracy	Validation Loss	Validation Accuracy
1	860	.4920	.8230	.6217	.7664
2	862	.1876	.9305	.7865	.7576
3	861	.1194	.9557	.8026	.7765
4	862	.0881	.9664	.9300	.7657
5	864	.0685	.9738	1.1032	.7710
6	864	.0545	.9792	1.2069	.7721
7	864	.0458	.9827	1.4088	.7749
8	862	.0401	.9846	1.5148	.7658
9	863	.0357	.9862	1.6642	.7662
10	866	.0301	.9884	1.4997	.7658
11	865	.0258	.9902	1.4008	.7716
12	865	.0245	.9912	1.7348	.7799
13	867	.0217	.9917	1.6799	.7720
14	863	.0201	.9926	1.6527	.7724
15	864	.0172	.9940	1.7161	.7673
16	864	.0140	.9945	1.8875	.7702
17	865	.0135	0.9952	1.7727	.7760
18	865	.0148	0.9949	2.1333	.7726
19	864	.0106	0.9960	2.0620	.7777
20	864	.0113	0.9960	2.0284	.7723

Table 2: Learning Rate at Each Epoch for Model 1

4.6 Training Time

The total time it took to train Model 1 was 17274 seconds. Which is equivalent to 287.9 minutes, which is roughly 4.8 hours. We initially wanted to have 40 epochs in our Model 1, although we realized that this would mean the model would take much longer to train, which was unfeasible for us.

5 Evaluation Metrics

The evaluation process for this challenge is distributed between two levels. Level 1 is to classify headline and body text as "related" or "unrelated" with a 25% score weighting. Level 2 is to classify pairs as "agrees" or "disagrees" with a 75% scoring weighting. The score weighting first goes to level 1 where it asks if the headline and body are unrelated, if it's a no, then the stance is unrelated, if yes, the scoring system proceeds to level 2. Level 2 asks what the relationship is between the article and body, as at this point, we know that they are related. At this level, the stance can either be "agree", "disagree" or "discusses". The scoring system is set up like this because the related/unrelated classification task is an easier classifier task than classifying the relationship between a headline and body pairing. In addition, the classification task of if are they related or not is a less relevant task for the detection of fake news. The scoring weight flowchart can be seen below:

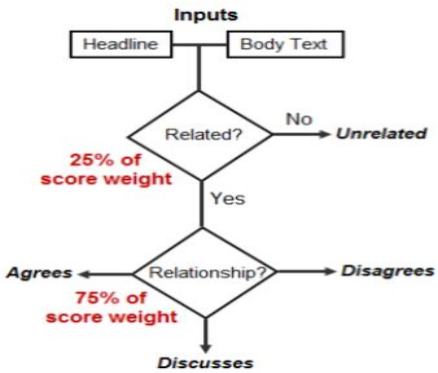

Figure 3: Scoring Process Flowchart

6 Results

6.1 Evaluation

We can look at the quantitative evaluation for both methods by analyzing the validation accuracies and the training accuracy. If we graph Table 2, figure 4 clearly shows us that as the training accuracy improves, the accuracy on the validation set remans around the same value. Therefore it can be said that even though certain precautions were taken to avoid overfitting, the model ended up doing so as the epoch with the highest validation accuracy saved into the checkpoint file was epoch 12.

The confusion matrices can also be used to see where the model is making the biggest mistakes in Figure 5. The model does a great job at predicting unrelated stances with an F1 score of 90% however the disagree and agree stances are poorly predicted with an F1 score of respectively 3% and 36%.

TF-IDF Word2Vec Model

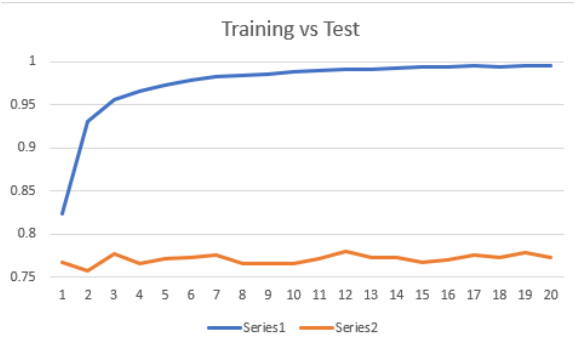

Figure 4

	agree	disagree	discuss	unrelated
agree	817	27	422	637
disagree	186	12	170	329
discuss	891	12	2071	1490
unrelated	687	25	661	16976

	precision	recall	f1-score	support
agree	0.32	0.43	0.36	1903
disagree	0.16	0.02	0.03	697
discuss	0.62	0.46	0.53	4464
unrelated	0.87	0.93	0.90	18349
accuracy			0.78	25413
macro avg	0.49	0.46	0.46	25413
weighted avg	0.77	0.78	0.77	25413

Figure 5

We can see in the BERT model from our train vs test results through the epochs that the loss and accuracy for the validation/test remain stagnant after the 20 epochs that we did. From the confusion matrix, it is interesting to point out the fact that disagree was never predicted using the BERT model making its score for that category 0%. Furthermore, there were a lot more "unrelated" guessed that were incorrect, lowering its total score. The final score on the competition dataset for this model was 41%. This model however was much more efficiently with an average epoch time of around 1 minute compared to our model 1 of 15 minutes.

BERT

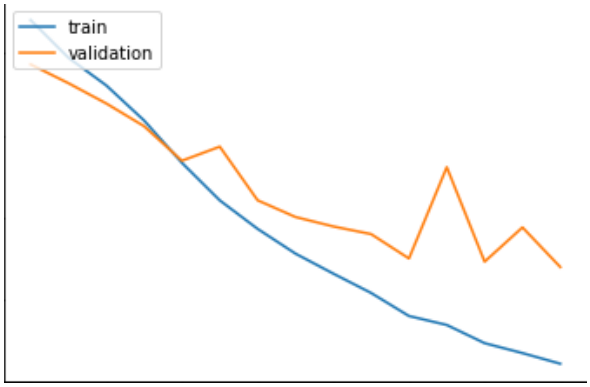

Figure 6 - loss

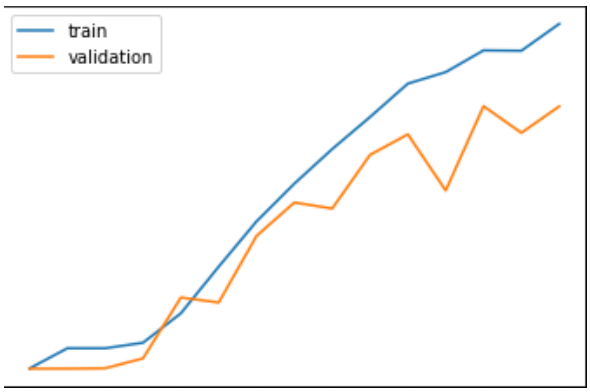

Figure 7 – accuracy

	agree	disagree	discuss	unrelated
agree	215	0	157	1531
disagree	79	0	50	568
discuss	245	0	995	3224
unrelated	1263	0	3358	13728

Figure 8

7 Conclusion

This report outlines 2 models that were created and fitted on the data provided to us by the FNC-1 challenge. This data includes a corpus of headline and body pairings, along with a stance between the pairing. This stance can either be "agree", "disagree", "related" or "discuss". The purpose of our models was to best predict the stance of a

headline, body pairing. The first model was a TF-IDF vectorizer combined with Word2Vec embedding, fed through a feed forward neural network. The second model was a BERT embedding matrix fed through a bidirectional LSTM. The first model achieved a weighted score of 65.64% with codalab score of 7648.25 and the second model achieved a weighted score of 41% The key difference between the two models is that the first model takes the concatenation of two features as in the input. These features are the TF-IDF vectors and Word2Vec embeddings. In contrast, the second model only created one feature as input. The feature for the second model is an embedding matrix created from the BERT model. Throughout this project, we were able to apply our knowledge on string embeddings as input layer for a classifier model. The Word2Vec embedding model was familiar to us prior to this project, although TF-IDF and BERT embeddings were new to us, and we were able to gain valuable insights on why they should be used when building a classifier model. In addition, we gained more knowledge about model architecture, as it was up to our discretion to build out the model without any direction from the FNC-1 challenge. We had to conduct our own research on how to best build out our models. Finally, we learned the importance of data preprocessing for classifier models, as we implemented the removal of stop words, tokenization, and special character removal as part of our data preprocessing. Even though we were unable to improve the accuracy of the baseline model, we still gained important insights and had the opportunity to apply our knowledge of natural language processing in a real-world scenario.

Acknowledgments

Thank you to Professor Vechtomova and to TA Peyman Kiasari for a wonderful course. The most memorable course for us in our last semester at University of Waterloo. We wish you all the best!

References

[1] Sean Baird. *Solat In the Swen – deep_learning_model.* 2017. https://github.com/Cisco-Talos/fnc 1/tree/master/deep_learning_model

[2] Andreas Vlachos, Kalina Bontcheva, Isabelle Augenstein and Tim Rocktaschel. *Stance Detection with Bidirectional Conditional Encoding*. 2016. https://arxiv.org/pdf/1606.05464.pdf

[3] Bruno Stecanella. *What is Tf IDF.* 2019. https://monkeylearn.com/blog/what-is-tf-idf/

[4] Chris Nicholson. *A Beginner's Guide to Word2Vec and Neural Word Embeddings.* 2017. https://wiki.pathmind.com/word2vec

[5] Raman Kumar. *All You Need to know about BERT.* 2021. https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-bert/

[6] "English · Spacy Models Documentation." English, https://spacy.io/models/en.

A Supplementary Material

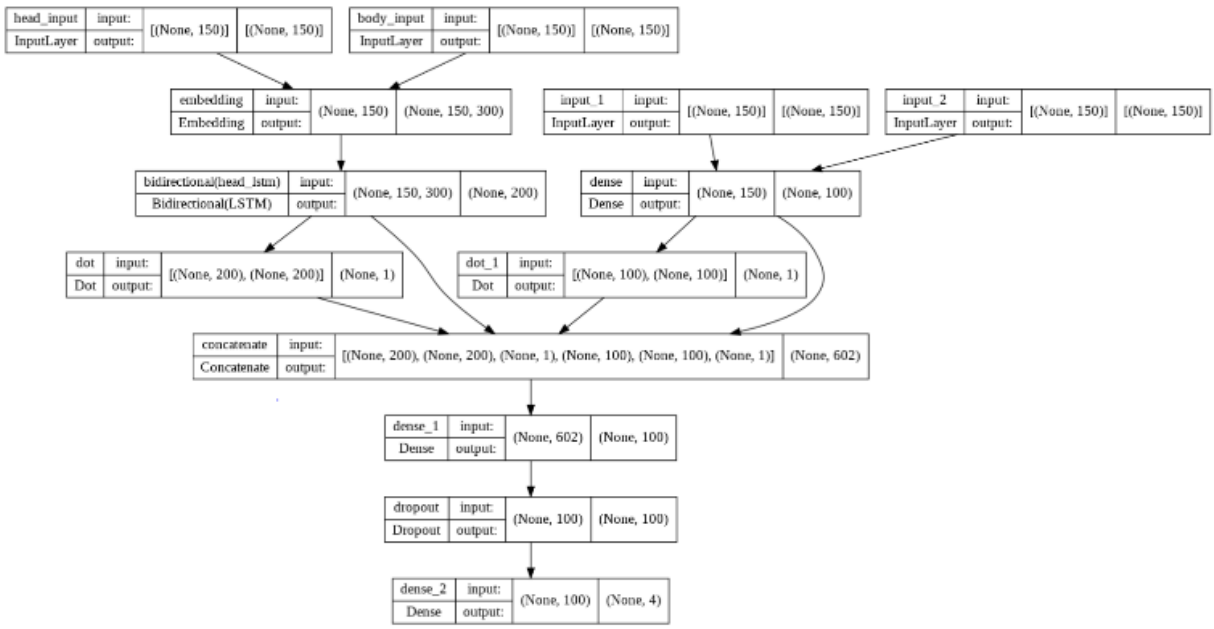

Figure A.1

GitHub Link: https://github.com/YE5BOSS/msci-nlp-w22