Fake News Stance detection using Deep Neural Networks

Vyas Anirudh Akundy
Electrical and Computer Engineering
University of Waterloo
ID: 20765080
vaakundy@uwaterloo.ca

Atish Telang Patil
Electrical and Computer Engineering
University of Waterloo
ID: 20793593
ahtelangpatil@uwaterloo.ca

Abstract

Fake news is a very common problem these days which can lead to the spread of misinformation and false facts. The task of assessing the veracity of a news article is a complicated one. However, a first step that can be taken in identifying fake news is to detect the stance of two pieces of text, i.e estimating the relative perspective of two pieces of text. This problem is part of an open challenge called ”Fake News Challenge Stage 1 (FNC-I): Stance Detection” This paper presents a neural network model that classifies whether a given news headline and article are related or unrelated. The model takes in as input the BERT(Bidirectional Encoder Representations from Transformers) embeddings of the headline and article text, Tf-idf(Term-Frequency-Inverse Document Frequency) vectors, distance metrics and few other features, and predicts the class of the stance. Our model achieves an accuracy of 82%(current and Codalab submission score) on the competition data set.

1 Introduction

Information, whether it is numerical or textual, is a vital means of communication between interacting parties. People rely on information, which can be a news article or information about the weather, to keep themselves informed. However when the information is false or ”fake”, it can cause a great deal of confusion and disruption of peoples lives which is a serious problem. This project aims at developing a solution to this problem using machine learning and natural language processing.
The problem can be broken down into simpler steps or stages, the first of which is part of an open challenge known as ”Fake News Challenge Stage 1 (FNC-I): Stance Detection”. Given a body text from a news article and a headline, the goal of this challenge is to classify the stance of two pieces of text, i.e the body text may agree, disagree, discuss or be unrelated to the headline.
The baseline model uses a Gradient Boosting Classifier along with few features extracted from the data to classify the data points into the four categories. We present a neural network model that uses, in addition to the baseline features, few other features we have constructed which will be explained in detail later. Our model takes in two sets of inputs. First is the BERT embeddings and Tf-idf vectors of the headline and body. The second set is the baseline and similarity features. For the similarity features we used a combination of multiple distance metrics(cosine, euclidean, cityblock etc.) and few fuzzy string matching features. The model produces two softmax outputs, the average of the two being our final prediction.
The remainder of the paper is organized as follows. Section 2 discusses some of the previous work done to solve this problem. Section 3 describes our approach, the model architecture and feature engineering in detail. Section 4 covers the experiments that we have carried out,the results we have obtained and Section 5 concludes the paper with a discussion on what we have learned, further improvements and future work.
The code for the project can be found in the [1]Github link provided in the footnote

2 Background/Related Work

The FNC-I challenge involves taking a pair of inputs - {headline,body} and classifying them into one of four categories - ’agrees’, ’disagrees’, ’discusses’, ’unrelated’. Our project is an extension of the work done by [1], which uses a gradient boosting classifier and a set of hand-crafted fea-

[1]https://github.com/Anirudh42/MSCI641-Project

tures to perform the classification task. They extract relevant features such as the number of common words in both headline and body, the number of common ngrams(uni, bi and chargrams) between the two texts and few other features. This model acheives a score of 75.2% on the competetion set.
There have also been other attempts to solve this problem. [6] rely only on the bag-of-words(BOW) representation of the headline and body. They use these as their input features to their multi-layer perceptron (MLP) model and achieve a good performance of 81.72% which is on par with many other more elaborate ensemble approaches.
Attention [2] is another powerful technique, especially helpful in machine translation, in which the decoder can attend to/focus on specific words in the encoder when translating from one language to another. [5] developed their model based on the attention mechanism which they have built on top of a BOW and LSTM model. In this setting, the headline and the body are processed separately in two LSTMs. The final hidden state of the headline LSTM is sent as input to the hidden state of the body LSTM. The attention mechanism then operates over the headline LSTM outputs and the final hidden state of the body LSTM.
Apart from the baseline, we also took inspiration from [7], who developed an ensemble which consists of 50-50 weighted average of a Deep CNN model and a Gradient-Boosted Decision Trees (GBDT) model which made them the top submission in the FNC-I challenge in 2017.
The following sections explain our approach and experiments in detail.

3 Approach:

3.1 Architecture of the Model

The architecture of our model can be seen in Figure 1. It consists of two branches. The right branch takes input as the Baseline and Similarity features and the left branch takes in the BERT embeddings and tf-idf vectors of the headline and body as its input, details of these features are provided in the subsequent section.
Each branch is passed through a series of dense layers and later concatenated to form one single vector. This is done to balance the number of features on both sides. The Baseline + Similarity features combined have 56 features, whereas the BERT + tf-idf features have over 2000 features due to which the BERT + tf-idf features may dominate the other set of features. So we have to scale them to balance the number of features.
There is a secondary output(shown as "Probability Outputs 1") and primary output(shown as "Probability Outputs 2"), each with its own loss function, which will help it generalize well to the given task. The secondary output is taken from the right branch as seen in the Figure 1. There is a 4-layer dense network just before both the Softmax functions.
The primary output is then combined with the secondary output by taking a weighted average(equal weights were given to two softmax outputs), to give the final predictions. This weighted average boosted our models accuracy by $\approx 2\%$.
The output is calculated as follows:

$$y_1 = [a_1, a_2, a_3, a_4]$$

$$y_2 = [b_1, b_2, b_3, b_4]$$

where y_1 is the the primary softmax output and y_2 is the secondary softmax output and a and b are the probabilities for each class. Hence the final predictions are given as,

$$y_{final} = argmax(0.5 * y_1 + 0.5 * y_2)$$

'argmax' is done here to select the class with the highest probability.

3.2 Feature Engineering

3.2.1 Baseline features

The Baseline consists of 4 features: Word overlap, Refuting, Polarity and Hand features. These four features are explained in detail below:

- **Word overlap feature:** The intersection of the words from the headline and body are divided by the total number of words from the headline and body. This is calculated for each headline and its corresponding body.

- **Refuting feature:** A set of refuting words are predefined, example words like 'fake','fraud','hoax' etc. The headline is tokenized and if the words in the headline are present in this list, then that particular word(in the list) is given a value of 1 else 0. This is done for all the headlines and then passed as a feature to the model.

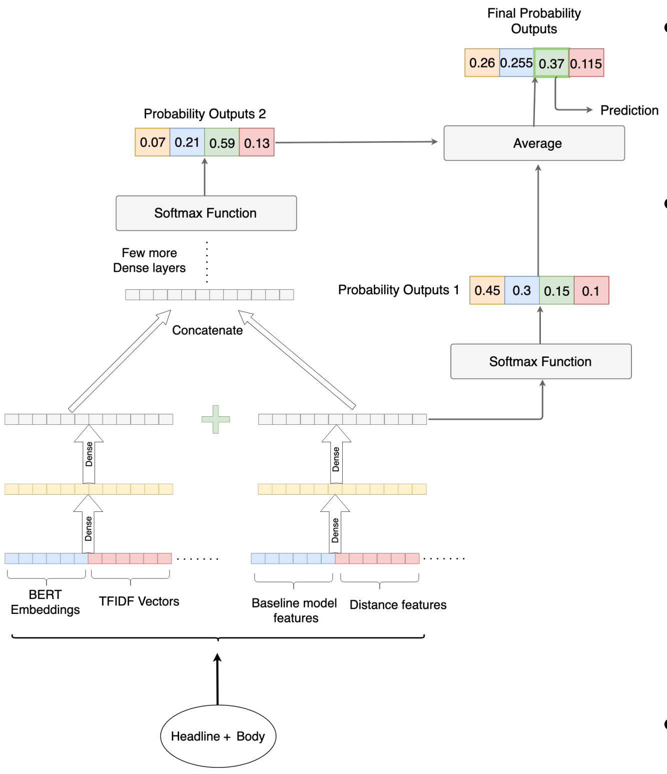

Figure 1: Architecture of Neural Network Model

- **Polarity feature:** The number of refuting words are counted in the headline and its corresponding body, this is then added as a feature to the model.
- **Hand feature:** There are 2 things calculated in this, first the count of a token in the headline appearing in the body is calculated. Second the count of n-grams from the headline appearing in the body are calculated and appended as a feature.

3.2.2 Similarity Features

We have used various distance metrics to compute the similarities between the headline and body embedding. Each distance measure is used for different situations. Hence we have captured various aspects of similarity between the headline and body. Apart from the distance features, we have also used the fuzzywuzzy string matching library to estimate the similarities between two sentences in various ways. Below are the list of distance features used, where a, b, x, y are the two vectors for which similarity is being calculated

- Cosine distance

$$\cos\theta = \frac{\bar{a}.\bar{b}}{\mid \bar{a} \mid\mid \bar{b} \mid}$$

- City block distance

$$\sum_{i=1}^{n} \mid x_i - y_i \mid$$

- Jaccard distance

$$s_{ij} = \frac{p}{p+q+r}$$

where,
p = number of variables that are positive for both objects
q = number of variables that are positive for the i^{th} objects and negative for the j^{th} object
r = number of variables that are negative for the i^{th} object and positive for the j^{th} object
s = number of variables negative for both objects

- Canberra distance

$$d(x,y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}$$

- Euclidean distance

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- Minkowski distance

$$(\sum_{i=1}^{n} \mid x_i - y_i \mid^p)^{1/p}$$

3.2.3 TFIDF

A Term Frequency is a count of how many times a word occurs in a given document (synonymous with bag of words). The Inverse Document Frequency is the number of times a word occurs in a corpus of documents. Tf-idf is used to weight words according to how important they are. Figure 2 gives an example of tf-idf. Each word will have a vector representation of length 500(chosen by us). We constructed vector representations for both headline and body. We have then combined these 2 vectors to a form a single tf-idf feature.
tf-idf of a given word is calculated as follows:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

$$idf(w) = \log(\frac{N}{df_t})$$

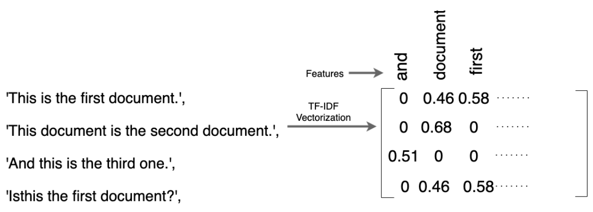
Features
and
document
first
'This is the first document.',
'This document is the second document.',
'And this is the third one.',
'Isthis the first document?',
TF-IDF Vectorization
0 0.46 0.58
0 0.68 0
0.51 0 0
0 0.46 0.58

Figure 2: TF-IDF example

Combining the above two equations we get the tf-idf score($w_{i,j}$) for a given word in a document,

$$w_{i,j} = tf_{i,j} \times \log(\frac{N}{df_t})$$

Here $tf_{i,j}$ is the number of occurrences of i in j. df_i is the number of documents containing i and N is the total number of documents

3.3 BERT

BERT [3] stands for Bidirectional Encoder Representations from Transformers. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its original form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is used. We have used a pre-trained BERT model which will provide us with a vector representation of each word. In our case we pass the headline and body into the pre-trained BERT model and it gives us a 786 dimensional vector representation for both headline and body, which is used as a feature in our model.
The BERT representation is an essential feature as the contextual meaning is added as a feature to the model. Knowing the meaning of the headline and body will help the model capture the stance between the two texts.

4 Experiments

4.1 Dataset

The dataset, obtained from [1], consists of a training set and a competition set on which the submission is to be made. The data is organized into two files, one containing the news headlines, article body ID and the corresponding stance(agree,disagree,discuss,unrelated). The other file contains the actual articles and their appropriate ID, mapping the ID's from the first file.

There are a total of 49972 training data points out of which 36545 are labeled as **'unrelated'**, 8909 are labeled **'discuss'**, 3678 are **'agree'**, 840 are labeled as **'disagree'**. The data is highly imbalanced and more biased toward the 'unrelated' label. Figure 3 represents the distribution of the data among the four classes. There are 25414 datapoints in the competition dataset.
For our project, we used an 80/20 split, i.e we used 80% of the training set for training our model and 20% of the training set as a validation set on which we have tuned the model.

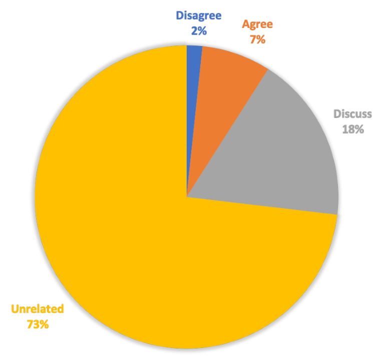

Figure 3: Distribution of the Data

4.2 Details

4.2.1 Mode configuration & Experiment

Figure 4 shows a detailed overview of the neural network model configuration along with the number of hidden units. All the different parameters which include the number of hidden units for each layer, the values of each individual dropout layer, activation functions were experimented with extensively to arrive at the best model.

The output of the architecture consists of two sets of softmax probabilities. We took a weighted average of the two outputs. The weights were varied and it was observed that giving equal weight to both the probabilities resulted in a good performance. And this ensemble was found to boost the accuracy of the model when compared to a single output model.

The number of training epochs, batch size, number of hidden units for each dense layer, dropout and the activation function were varied as shown in Table I. Upon experimentation, we found that the best number of epochs was 15 and the best

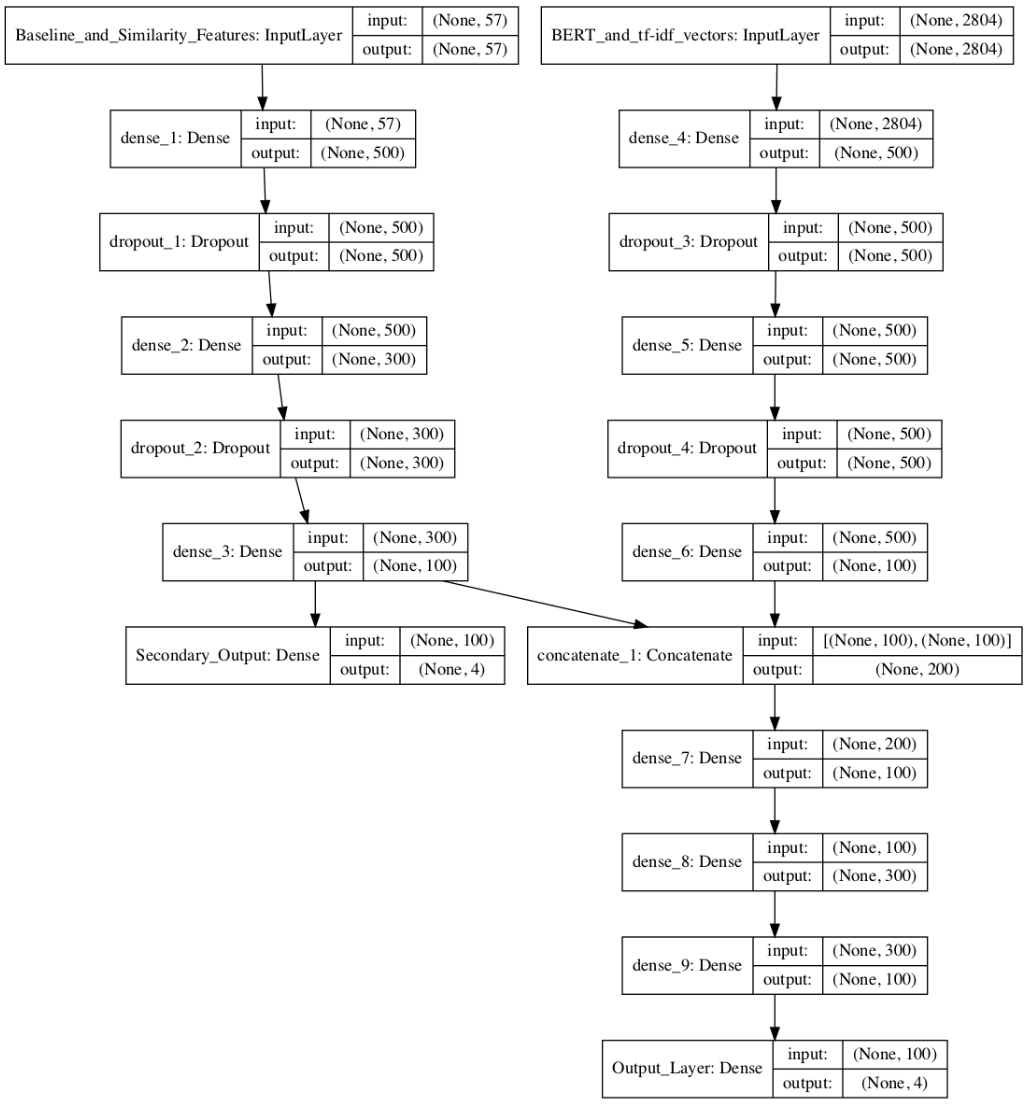

Figure 4: Model Configuration

Epochs	5,10,15,20,25,30
Batch Size	32,64,128,256,512
Hidden Units	100,200,300,400,500
Activation Function	tanh, sigmoid, ReLU
Dropout	0.1,0.2,0.3,....1

Table 1: Parameter search

batch size was 128.
Using a single activation function resulted in a poor performance. Hence through experimentation we found that the use of a combination of ’tanh’ and ’sigmoid’ activation functions achieved a good accuracy.
The optimizer we used was the Adam optimizer with a learning rate of 0.001 which was kept constant throughout the experiment.

The model was trained on an Nvidia GTX 1060 GPU which is hosted in the eceUbuntu4 machine of the Electrical and Computer Engineering Department. The training time for the model is 5 minutes and 16 seconds.

4.2.2 Evaluation Metrics

The metric used to evaluate the performance of our model can be illustrated in a flowchart shown in Figure 5, taken from the fake news challenge website [4].
Briefly, the scoring system is a simple classification accuracy but one which is biased towards giving a higher score when the classes ’Agree’, ’Disagree’ and ’Discusses’ are labeled correctly. A score of 0.25 is awarded when the ’Unrelated’ class is labeled correctly and a score of 0.75 is awarded when the other classes are identified cor-

rectly since it is much easier to predict the 'Unrelated' class than the other classes.
A confusion matrix is constructed to evaluate the classification over all four classes which is shown in the Results section.
The confusion matrix will give us details about the number of correctly and incorrectly predicted classes in each individual category.

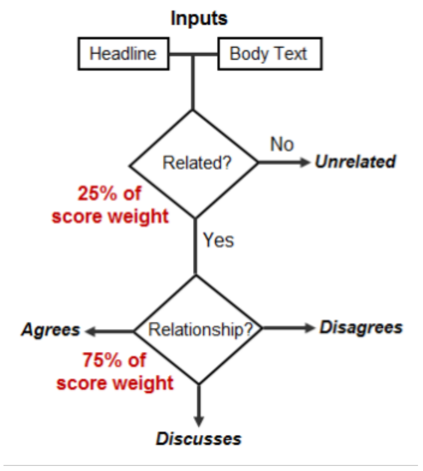

Figure 5: Scoring System

4.3 Results

4.3.1 Quantitative Results

Figure 6 shows the variation in the training and validation accuracy with the number of epochs. The overall performance of the model can be

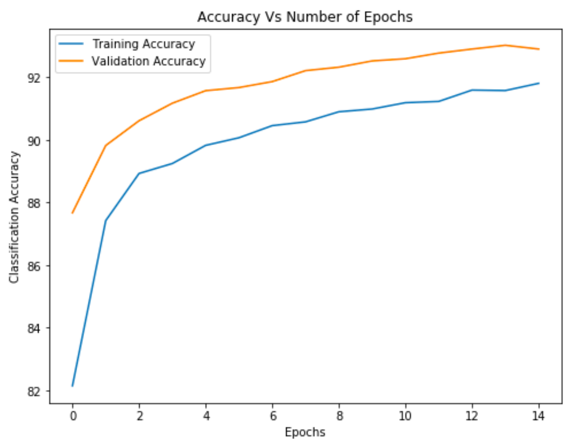

Figure 6: Train and Validation Accuracy

shown in the form of a confusion matrix as shown in Table 2, which is also illustrated in the bar graph shown in Figure 7

4.3.2 Qualitative Results

Two examples of the type of predictions made by the model is shown here.

Example 1

Headline: Fowler Falls For Fake Website, Makes Crazy Claims About Tennis Star
Body: SEOUL, Feb. 5 (Korea Bizwire) ,Äì A powerful robot vacuum cleaner caused an unlikely accident involving a Korean housewife, and required the intervention of a couple of paramedics. On January 3, a woman in her fifties had her hair sucked up into a robot vacuum at her home in the city of Changwon, South Korea.
Stance:'unrelated'

Example 2

Headline: EXCLUSIVE: Apple To Unveil The Long-Awaited Retina MacBook Air At Its ,ÄúSpring Forward,Äù Event
Body: Last week, Apple sent out the invites for its ,ÄúSpring Forward,Äù event, slated to be held at the Yerba Buena Center for the Arts in San Francisco on March 9th.....previous reports.
Stance:'agree'

	Agree	Disagree	Discuss	Unrelated
Agree	1038	0	721	144
Disagree	277	0	293	127
Discuss	645	0	3520	299
Unrelated	15	0	272	18062

Table 2: Confusion Matrix

By looking at the confusion matrix, we can see that there were no predictions for the 'disagree' class. The reason is that due to the high imbalance in data, with 'disagree' being approximately 2% of the entire dataset, the model does not see enough examples for the 'disagree' class.

4.3.3 Ablation Studies

We have conducted few experiments where the model was run each time with only one of the four input features,i.e we fed the BERT embeddings, Tf-idf vectors, similarity and baseline features individually to the network to observe the performance of the network and to evaluate the significance of each feature. The results of this study are shown in Table 3.
It is interesting to see from the table that for certain features like 'Similarity' and 'tf-idf', when taken individually performed very

Feature	Training Accuracy(%)	Test Accuracy(%)(Competition Set)
Baseline Features(Gradient Boosting Model)	87.31	75.20
Baseline Features(Neural Network Model)	90.10	76.15
Similarity Features	80.46	31.23
tf-idf	79.25	42.68
BERT	85.25	77.28
Baseline+Similarity features	87.54	79.25
Baseline + Similarity + BERT	91.93	78.90
Baseline + Similarity + tf-idf + BERT	**91.94**	**82.03**

Table 3: Ablation study results

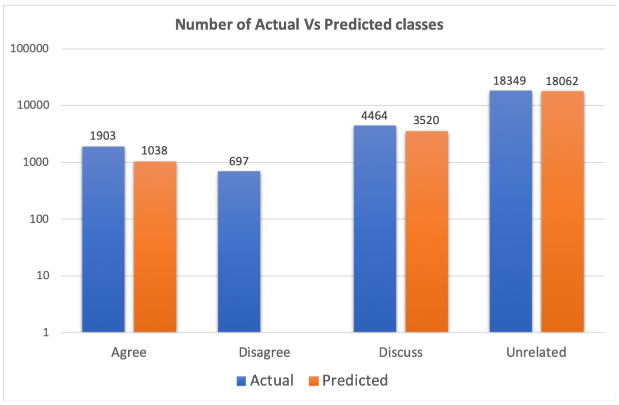

Figure 7: Bargraph of the Confusion Matrix

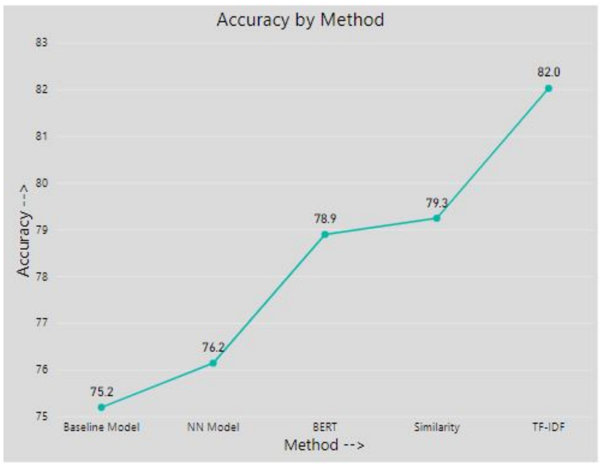

Figure 8: Accuracy by features/method

poorly, but produced good results when taken in combination with some of the other features. We can say that each feature is capturing a particular "meaning" from the headline and body, each of which needs to be combined to capture the whole picture. The BERT embeddings only provides a 2% increment in the accuracy giving 77.28% when used alone. However, it gives a 7% boost to the accuracy from 75.20% to 82.029% when it is appended as an additional feature to the 'Baseline + Similarity + tf-idf' feature set, which shows that word vector representations, and especially BERT, adds the meaning of the sentences which is essential in determining the stance between the headline and body.

Figure 8 shows how the models accuracy improved by an incremental addition of each feature. The x-axis indicates each feature being added to the model on top of the previously existing features. The first point is the accuracy for the baseline model and the second point is the accuracy for our neural network when used with the same baseline features.

5 Conclusion

This paper presents a neural network approach to address the challenge of identifying the stance of two pieces of text. Given a news headline and an article, the goal was to classify the relation between the two into one of four classes. Our approach was based on a neural network model which takes in two sets of features, one being the concatenation of BERT embeddings and the tf-idf vectors of the headline and body, second being the concatenation of the similarity and baseline features. We obtained two softmax probability outputs from the network, which were averaged to get the final prediction. Our model achieved an accuracy of 82% on the competition set, the baseline model score being 75.2%.

This project helped us understand the importance of feature engineering and word vector representations in natural language processing. We gained insights on the neces-

sary steps involved in preprocessing textual data(tokenizing, lemmatizing, removing special characters etc.).
In the future, we would like to experiment with different types of optimizers and different learning rates, conduct a more detailed and comprehensive feature engineering, provide a solution to overcome the data imbalance problem and also experiment with other types of neural networks, for example with siamese networks, to further boost the performance of our current model.

References

[1] H. van Veen B. Galbraith H. Iqbal et al. *A baseline implementation for FNC-1, 2017*. URL: https://github.com/FakeNewsChallenge/fnc-1-baseline.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL: http://arxiv.org/abs/1409.0473.

[3] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[4] *Fake News Challenge*. URL: http://www.fakenewschallenge.org/.

[5] Stephen R. Pfohl. “Stance Detection for the Fake News Challenge with Attention and Conditional Encoding”. In: 2017.

[6] Benjamin Riedel et al. “A simple but tough-to-beat baseline for the Fake News Challenge stance detection task”. In: *CoRR* abs/1707.03264 (2017). arXiv: 1707.03264. URL: http://arxiv.org/abs/1707.03264.

[7] Doug Sibley Sean Baird and Yuxi Pan. *Talos Targets Disinformation with Fake News Challenge Victory*. URL: https://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html.