Stance Detection for Fake News using Deep Neural Networks

Aishwarya Krishna Allada
Electrical and Computer Engineering
University of Waterloo
Student ID: 20816326
akallada@uwaterloo.ca

Amith Nandakumar
Electrical and Computer Engineering
University of Waterloo
Student ID: 20859891
a4nandak@uwaterloo.ca

Abstract

Fake News is one of the most common problems these days, with the advent of social media and other resources and lack of knowledge with the population on average. Fake news can spread false facts and lead to lots of misinformation. While assessing the veracity of news is difficult, Fake News Challenge Stage 1 (FNC-I): “Stance Detection” is a public challange to assess the data set of news articles consisting of headlines and their respective article bodies and predict the stance of the news. The detection is to identify whether a particular headline “agrees”, “disagrees”, “discusses” or is “unrelated” to its related article body. In this paper, various traditional machine learning algorithms and neural networks from simple to complex architecture with wide multiple vocabularies were implemented to tackle the stance detection and their performance is analysed. Ultimately, the best model tried was found out to be delivering the best performance of 80.87% weighted accuracy. This report presents a detailed analysis of the experiments performed on the models and detailed explanation of the final model chosen, which best classifies the stances, as needed for the challenge.

1 Introduction

Information is a significant part of communication between two entities and it is often prone to spoofing. Be it numerical or textual, it can be modified easily to garner attention from different parties and entities for their benefits. For example, any news article or weather forecast, which is read by millions of people to keep them informed can be easily faked. Having said that, it creates panic and also creates a huge difference in communicating the right information to the right audience which is a serious concern.

The New York times defined fake news as made-up stories written with the intention to deceive and are published in such a way that it would seem like a traditional news article to any common man on the receiving end. These deceiving techniques have reached a new level for spammers which helps in attracting web traffic to their websites and helps in making money through ads. This is also a threat to the journalism standards and reputed websites which serve the public in keeping them informed. Above all, these tactics are used in political propaganda’s and fake news has gained traction in the last elections of the states for increasing political polarization.

The paper is divided into multiple sections where section 2 discusses the related work and some of the previous proposals, while section 3 explains about the approach taken to solve this problem and also explains the model architecture along with the feature engineering in detail. Section 4 covers the experiments, where the details of a dataset, mode configuration, evaluation metrics and other experiments performed are elaborated. Section 5 elaborates about the results and analysis by interpreting the performance measures and finally, Section 6 concludes the paper with a discussion on our learnings and future scope which is then followed by the references used to complete the project.

This code for this project can be found in the GitHub link[1] provided in the footnote.

[1] https://github.com/AishwaryaAllada/msci-text-analytics-s20/tree/master/Final%20Project

2 Background/Related Work

Many researches took place to come up with the best model to find the stance of the given information. The FNC-I challenge is about picking a headline and its respective article body and to classify it into a stance category to which they belong. The model achieved for this project is upon the improvisation of the baseline model [6] which achieved an accuracy score of 79.53%. The baseline model uses gradient boosting classifier along with k-fold cross-validation and some features like Word overlap, Refuting, Polarity and Hand features, where each of the features calculates the number of common n-grams, common words in a selected headline and body, etc to carry out the classification task.

Apart from the above baseline model, a similar task of stance detection on smaller texts which predicts if a tweet is "positive", "negative" or "neutral" by Augenstein et al. [1] uses a pair of LSTMs in various arrangements. Here, when one LSTM encodes the target string (independent encoding), the other LSTM encodes the tweet. The final hidden state vectors of the output of both the LSTMs is fed to a single feed-forward architecture having a softmax activation function for prediction. Another model was created, where two LSTMs were arranged sequentially, known as the "conditional encoding" model where the final hidden state of the target encoder was used to initialize the tweet encoder state while the final hidden state of the tweet encoder is used for prediction. A modification was made, making it "bidirectional conditional encoding", where the encoding happened in both the directions and the two final hidden state vectors from the bidirectional LSTM was used for prediction. This particular experiment showed that ”conditional encoding” performed better than ”independent encoding”.

Another experiment[5] was performed with multi-layer perceptron (MLP) where the input to the model was a bag-of-words (BOW) representation of headline and body. Here, the accuracy of 81.72% was achieved, by which the model performed reasonably well while matching the performance of other dynamic approaches.

Yet another approach[3] was made, which was one of the top submissions in the FNC-1 challenge in 2017, where a Deep CNN model and a Gradient-Boosted Decision Trees (GBDT) model are used using a 50-50 weighted average.

The upcoming sections elaborate our approach and other models experimented for the challenge.

3 Approach

3.1 Architecture of the Model

The architecture we used here has two branches, which is quite different from the usual, as two different vector representations of the headline and article body are fed to each of the branches. The output of these two branches is later fused, before sending it to the feed-forward neural network.

The headlines and the article bodies are first tokenized into text sequences using Keras library which is then fed to the left branch and pre-trained Word2Vec model of 300 dimensions is used to create a vector representation of the tokens for both the headlines and the article bodies. These vector representations of headline and body are then fed to a bidirectional LSTM with 100 hidden units. The extracted features from the headline and body are sent to a dot layer where the similarity between the features are obtained.

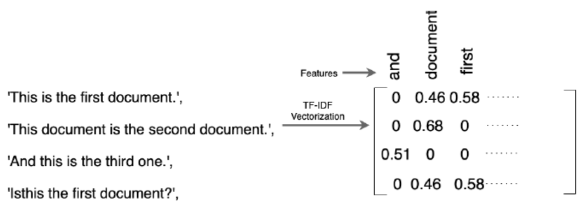
Similarly, for the right branch of the network, word representation for each token in headline and body are determined using Term Frequency- Inverse Document Frequency (TF-IDF) which is sent separately to a feed-forward neural network and the extracted features from headline and body is further sent to a dot layer where the similarity between the features are obtained.

Finally, the extracted features for the headline and the body from the left branch is fed to a bidirectional LSTM along with their similarity features, which are concatenated with the extracted features of headline, body and similarity from the right branch. The resultant features of the concatenated layer are finally fed to the feed-forward neural network having 100 hidden units and have a softmax activation function which produces the probabilities of which stance a specific article body and a headline belongs to.

3.2 Feature Engineering

3.2.1 Term Frequency- Inverse Document Frequency

TF-IDF is a measure which tells how relevant a particular word is for a document in a collection of documents. By multiplying the number of times a word appears in a document (Term Frequency) and the number of times a word occurs in a number of documents(Inverse Document Frequency), this statistical measure is used to weight words based on their importance amongst the rest of the words. Figure 1 is an apt example for TF-IDF.



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 1: TF-IDF Example

Hence each sentence will have a representation depending upon the importance of each word in the sentence. Various values of max features were experimented and a value of 150 was chosen for both article body and the headline. The calculation of TF-IDF for a word is as given below:

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

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

With the above two equations the score for every word can be obtained, which is represented as:

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

where the $tf_{i,j}$ is the number of times a word i appear in document$j$, df_i is the number of documents with the word i and the total number of documents is represented by N.

4 Experiments

4.1 Dataset

The dataset for the experiment is obtained from [1], which has a training set and a competition set to evaluate the model. Each of the sets has two files respectively, where the first file consists of the news headlines along with the article body ID and the corresponding stance (agree, disagree, discuss, unrelated) while the other file has the articlebodies along with the body ID. The IDs from both the files are used to map the headline, body and stances correctly.

Totally, there are 49972 training data points, where 36545 are labeled as 'unrelated', 3678 are 'agree', 8909 are labeled 'discuss' while the rest of the 840 data points are labeled as 'disagree'. As we can see, there is a high imbalance in the data, as the majority of the data is classified into "unrelated".

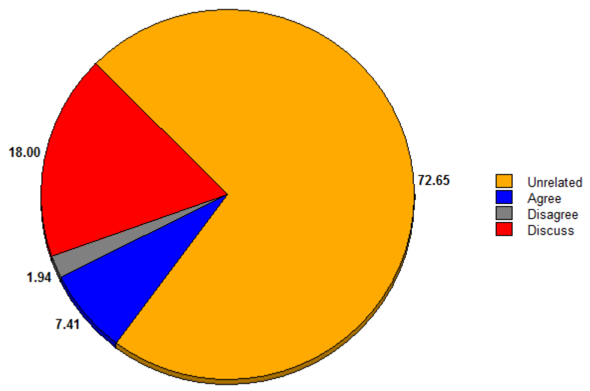


Figure 2: Distribution of the Data

The distribution of the data points between the 4 categories is as shown in the Figure 2.

In the competition data set, there are totally 25413 data points, where we used this data set as validation data for our model. When we initially split the training data set into 80/20 (train-val split) the model performed extremely good but performed poorly when it came to testing dataset as the number of samples were more. Hence for better performance, the model was trained on entire training samples and the performance of the model was evaluated on testing test.

4.2 Details

4.2.1 Model configuration Experiment

The Figure 3 shows the detailed architecture of the model which performed the best, along with all the hidden units used in the model. To arrive at the best model, different activation functions and hidden units were experimented for the hidden layer and it was found that ”tanh” activation for bidirectional LSTM and ”ReLU” activation for feed forward neural networks worked the best. In order to avoid over-fitting of the model on train data and to make the model more robust dropout layers were added by fine tuning the values for dropout.

4.2.2 Pre-Processing

Pre-processing was performed on both the headlines and article body, where the texts are converted to lower first and then the punctuation, special characters, new lines and numbers are replaced with space. The performance of the model improved upon pre-processing the text.

4.2.3 Design Choices

The number of epochs, activation functions used, dropout rates, hidden units, batch size variations are tabulated in Table 1. The best parameters upon tuning were found to be 20 for number of epochs, and the best batch size was 64. The loss function used was “Categorical Cross-Entropy” as the problem is of a multi-class classification with more than two exclusive targets and the targets are encoded as one-hot vectors. The metrics used is “accuracy”, where even precision, recall and F1 score is monitored. In order to save the best model, early stopping is employed where the model with best validation accuracy is saved, which ensures that even if the model starts to overfit during training, the best model will be preserved.

Epochs	5,10,15,20,25,30,40,50
Batch size	32,64,128,256,512
hidden units	50,100,150,200
Activation Function	ReLU, Softmax, tanh
Dropout	0.1,0.2,0.3....1

Table 1: Parameters search

4.2.4 Evaluation Metrics

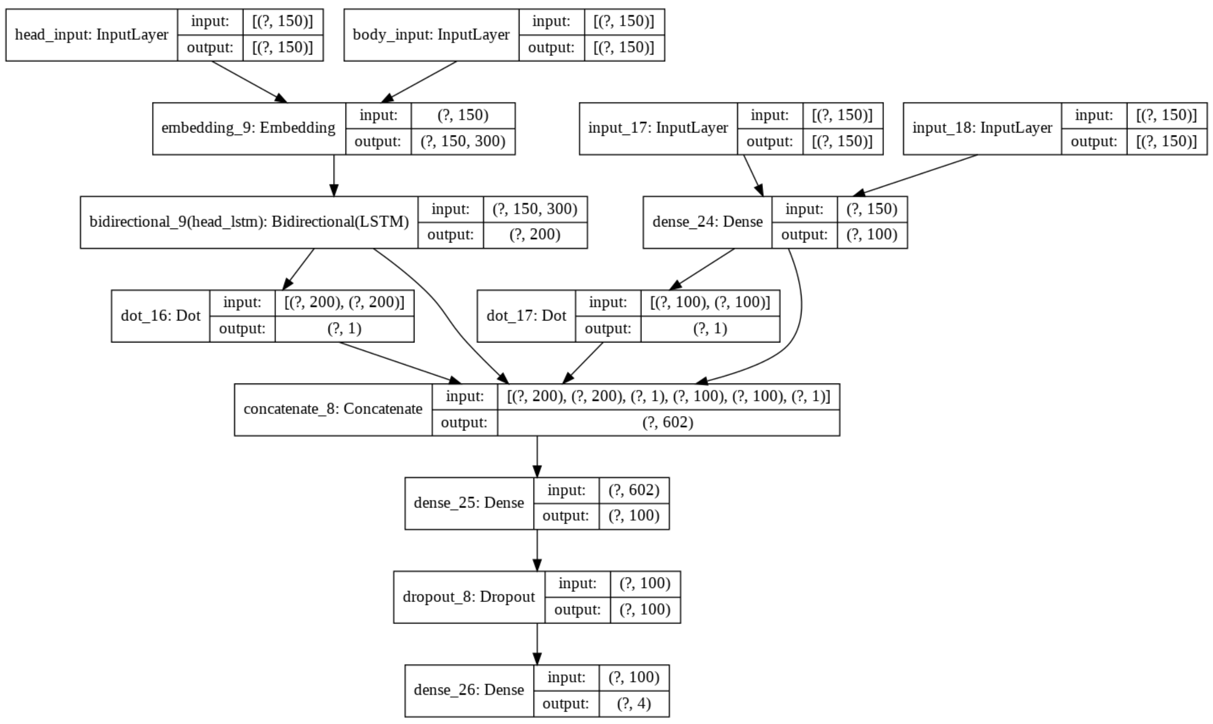
The metrics we used to evaluate all our experimented models is illustrated in the flowchart shown below in Figure 4. To elaborate the whole scoring system, as the labels are highly unbalanced, the result obtained over prediction will be a measure of weighted accuracy. We can see from Figure 4 that there are two levels, where if a label is correctly classified as “unrelated” for a Headline-Body, then the score is added by 0.25, else if the Headline-Body is related, (i.e) “discuss”, ”agree” or “disagree”, then the score is added by 0.75. The high value is because it is easy to predict the “unrelated” class when compared to the rest of the classes.

There is a confusion matrix which gives us the details about the number of correct and incorrect classifications that happened under each category for the data set.

4.3 Various Models Experimented

Pretrained word embeddings capture the semantic and syntactic meaning of a word as they are trained on large datasets. Word2Vec is trained on the Google News dataset, which is a model consisting of 300-dimensional vectors for 3 million words and phrases[5]. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase an interesting linear substructure of the word vector space [6]. FastText [7] on the other hand, was developed by Facebook, which represents each word as an n-gram of characters.

Bidirectional LSTMs with pre-trained word-embeddings like Word2Vec[7], GloVe[4], FastText[2] were implemented, where the model with GloVe embedding performed better which is followed by FastText and Word2Vec. Another model with Bidirectional LSTM with BERT embedding was created from SpaCy library. Another CNN+LSTM model was also implemented, where Word2Vec embeddings were used and



head_input: InputLayer | input: [(?, 150)] | output: [(?, 150)]
body_input: InputLayer | input: [(?, 150)] | output: [(?, 150)]
embedding_9: Embedding | input: (?, 150) | output: (?, 150, 300)
input_17: InputLayer | input: [(?, 150)] | output: [(?, 150)]
input_18: InputLayer | input: [(?, 150)] | output: [(?, 150)]
bidirectional_9(head_lstm): Bidirectional(LSTM) | input: (?, 150, 300) | output: (?, 200)
dense_24: Dense | input: (?, 150) | output: (?, 100)
dot_16: Dot | input: [(?, 200), (?, 200)] | output: (?, 1)
dot_17: Dot | input: [(?, 100), (?, 100)] | output: (?, 1)
concatenate_8: Concatenate | input: [(?, 200), (?, 200), (?, 1), (?, 100), (?, 100), (?, 1)] | output: (?, 602)
dense_25: Dense | input: (?, 602) | output: (?, 100)
dropout_8: Dropout | input: (?, 100) | output: (?, 100)
dense_26: Dense | input: (?, 100) | output: (?, 4)

Figure 3: Model Configuration

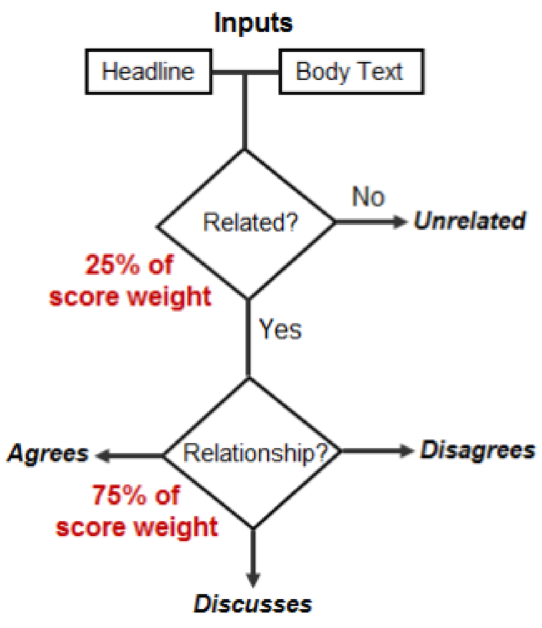


Figure 4: Scoring System

the model is mainly made up of Conv1D layer and an LSTM layer. The results obtained upon experimenting various neural networks are tabulated in the Table 3.

Classification using classical methods like Random Forest, Linear SVM, SVM, XG-Boost and Decision Tree classifier were implemented and the results are tabulated in Table 2. From the results obtained the neural network models performed better than the classical methods.

Models	Testing Accuracy	Weighted Score
Decision Tree	40.75%	4748.0
Random Forest	40.90%	4766.25
XGBOOST	40.91%	4767.25
Linear SVM	46.04%	5364.5
SVM	41.31%	4813.75

Table 2: Classical Models Performance

5 Results

5.1 Quantitative Analysis

Figure 5 shows the training and validation accuracies with respect to the number of epochs. Since the testing set was used as validation it is evident from the graph that even when training accuracy increases there was only a gradual increase in testing accuracy. It can be inferred that the model started to over-fit

Models	Training Accuracy	Testing Accuracy	Weighted Accuracy
Similarity Features(Word2Vec)+ Bidirectional LSTM	98.11%	77.31%	9008
CNN + LSTM	86.67%	46.54%	5422.75
Word2Vec+ Bidirectional LSTM	93.09%	51.67%	6020.25
GloVe+ Bidirectional LSTM	93.52%	55.62%	6480.75
FastText+ Bidirectional LSTM	92.37%	51.45%	5995.25
BERT embeddings+ Bidirectional LSTM	87.61%	60.73%	7352.75

Table 3: Various Neural Network Model's Performance

the train data however since checkpoints were used the best model is saved. Overall, the performance of the model is interpreted from confusion matrix tabulated in Table 4 which can also be visualized from Figure 6.

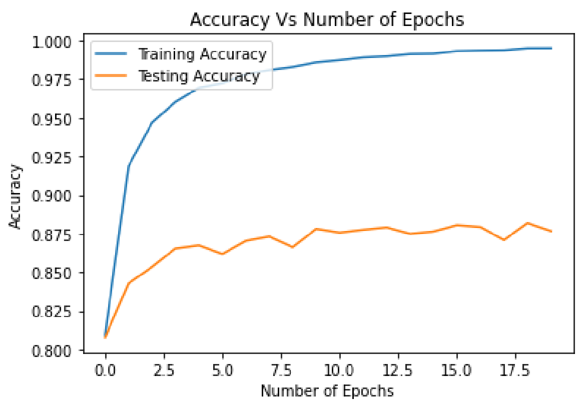


Figure 5: Accuracy vs number of epochs

	Agree	Disagree	Discuss	Unrelated
Agree	949	177	534	243
Disagree	85	305	171	136
Discuss	293	98	3388	685
Unrelated	136	55	391	17767

Table 4: Confusion Matrix

5.2 Qualitative Analysis

From the Table 4, we can see from the confusion matrix that the stance ”Disagree” has the least number of predictions amongst the rest, which is mainly due to the high variation in the samples of training data. As ”Disagree” constitutes a very small portion of the dataset, the model couldn't train as there were fewer examples of that particular stance.

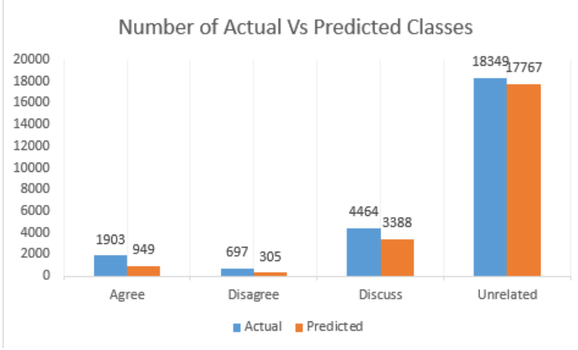


Figure 6: Bargraph of the Confusion Matrix

5.3 Interpretation of Performance Measures

As we can see from the Table 5, ”Unrelated” stance has the highest precision value and Recall value, as the whole data set is dominant with this particular category. Also, in case of F1-Score, which is the weighted average of precision and recall, ”Unrelated” stance has value of 0.96 which is approximately equal to 1.

	Precision	Recall	F1-Score	Support
Agree	0.65	0.50	0.56	1903
Disagree	0.48	0.44	0.46	697
Discuss	0.76	0.76	0.76	4464
Unrelated	0.94	0.97	0.96	18349

Table 5: Performance Measures

6 Conclusion

This paper presents a neural network to predict the stances of fake news, where two texts are given as input, namely headlines and article body. The aim of this challenge is to classify the given two inputs into one of the stances, (i.e) agree, disagree, discuss and unrelated. The various experiments performed clearly depicts that recurrent neural network models with LSTMs which have "memory enabled" units outperform non-recurrent model for this particular task. The approach we used here is different from usual as the model takes two sets of features. One of them is the concatenation of the Word2Vec embeddings of both headline and body, while the second input is the concatenation of the TF-IDF vectors of the headline and body. After concatenation, the vector is sent to a fully connected layer with "ReLU" activation function and then to a dropout layer with 0.2 dropout rate. Finally, the vector is sent to the output Dense layer with 4 nodes, which categorizes the headline and body to it's respective stance category. Our model achieved a final accuracy of 80.87 % on the competition dataset, while the baseline model gave 79.53% on the competition dataset.Before finalizing this model, many other models and architectures were experimented.

With this project, we gained valuable knowledge and insights of various embeddings and architectures to be used. Also, we understood how valuable preprocessing of text data (special characters removal and tokenization etc) is in Natural Language Processing Tasks.

This project can be improvised in the future by concatenating many features and by fine-tuning alternative complex architectures. Comprehensive feature engineering and overcoming the imbalance of the data can also be looked into to get better performance model for this particular problem. Also different neural networks like siamese architecture can be implemented by experimenting with different optimizers and hyperparameters values.

References

[1] Isabelle Augenstein et al. *Stance Detection with Bidirectional Conditional Encoding*. 2016. arXiv: 1606.05464 [cs.CL].

[2] *Fasttext - Library for efficient text classification and representation learning*. URL: https://fasttext.cc/.

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

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. URL: https://nlp.stanford.edu/projects/glove/.

[5] Benjamin Riedel et al. *A simple but tough-to-beat baseline for the Fake News Challenge stance detection task*. 2017. arXiv: 1707.03264 [cs.CL].

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

[7] *Word2Vec - Tool for computing continuous distributed representations of words*. URL: https://code.google.com/archive/p/word2vec/.