COMPGI15 - News Stance Detection

Parham Oghabi
parham.oghabi.17@ucl.ac.uk

1. INTRODUCTION

Online media platforms allow news to be shared in real-time and disseminate much quicker than before, without any moderation or verification. Fake news can have a severe impact on society and can bias the judgements and actions of humans. Therefore, machine learning algorithms can be used to aid fact checkers. An initial step in identifying fake news is to detect the stance of the article body relative to the headline/claim. In this project, the FNC-1 dataset was used. [1] The goal is to classify a headline and article pair as being unrelated, discuss, agree, or disagree.

2. Literature Review

Stance detection is an ongoing research problem and various ideas have been published. There are two approaches to this classification task: either to use feature engineering, which requires domain knowledge, or to use neural NLP and machine translation models.

The FNC-1 baseline involves hand-engineered features such as percentage of overlapping words between the headline and article, binary vector indicating refuting words in the headline, polarity of the article, counting the presence of an n-gram of the headline in the article body. [2] These features are concatenated and inputted into a gradient boosted decision tree. This model produces a relatively good weighted FNC-defined score of 75% on the test set.

In addition to the baseline features, further feature engineering was done by the top 2 winners in the FNC challenge such as non-negative matrix factorization, TF-IDF features, Word2Vec features, and various topic modelling techniques such as LDA.[3][4] Various cosine similarity metrics were computed between the topic model vectors of the headline and article and between the SVD of the TF-IDF vectors of the headline and article. The cosine similarity metrics proved to be very useful in my results as well. FNC challenge winner 1 used an equally weighted ensemble model of inputting hand engineered features into Xgboost and applying 1-D convolution layers on the headline and article word embeddings to determine the stance. FNC challenge winner 2 simply used a 3-layer multi-layer perceptron.

On the other hand, neural encoding networks have also proven to be effective without performing any feature engineering. [5] discusses various neural encoding architectures such as bi-directional RNNs and conditional bi-directional RNNs with attention. The RNN uses the padded headline and article word embeddings to encode the headline and the article independently and then concatenates the encoded results and feeds them into a multi-layer perceptron. The difference with the conditional RNN is that it first encodes the headline and then uses the final state of the RNN weights to initialize the article RNN encoder and then encodes the article. Attention is added to improve the results by identifying the most relevant part of the article for a given headline. Therefore, each article token is assigned an attention weight.

The results of the conditional bi-directional RNN with attention was much higher than the FNC challenge winners but it was evaluated on a hold-out set derived from the training data and not on the actual test set.

3. Dataset, Splitting, and Pre-processing

The following pre-processing techniques were performed on all the headlines and articles: tokenization using regex, removal of stop words and non-ascii characters, stemmed and non-stemmed version of tokens.

The training data was severely imbalanced with the "unrelated" stance being the majority. A validation subset was derived while maintaining the imbalanced ratio of the four classes and having a training:validation proportion of 9:1. The splitting process was as follows: Non-uniform sampling was done on the 4 labels, with probabilities equal to their training data ratios. Then a training sample with the selected label was sampled uniformly from the training set and moved to the validation set. This was repeated until the desired 9:1 proportion was met. The statistics of the original training data and derived validation data are shown in Table 1. We can see that the unrelated stances are downsampled in the new training set during the validation set creation, which is helpful because of the over-occurrence of the "unrelated" class. The "disagree" class occurred only 1% of the times and therefore this means that our model will probably have difficulties in achieving a high recall for this class. There are 1683 unique articles in the training set and 904 unique articles in the test set.

Table 1. Statistics of Training and Validation Data

	Original Training Set	Validation Set	New Training Set
Samples	49,972	4,997	44,975
Unrelated	73.13%	73.9%	65.7%
Discuss	17.82%	17.47%	16.08%
Agree	7.36%	7.26%	6.63%
Disagree	1.68%	1.36%	1.54%

4. Vector Representation

The cosine similarity between the headline and article was calculated for the train, validation, and training set. Both the bag-of-words (BoW) model and Word2Vec was used to represent the vectors, resulting in 2 cosine similarity calculations. Therefore, 2 features were derived from this step.

In the BoW model, the vocabulary used was all the pre-processed and stemmed tokens of the training articles which resulted in 15,619 unique words. The term frequencies of tokens for the headline and article were also included in the vectors, instead of using a binary BoW vector.

For the Word2Vec model, I used pre-trained word vectors from Google News which was trained on 100 billion words and has a vocabulary of 3 million words. Each word vector has 300 dimensions. For each of the headline and article pair, the word vectors of the tokens in each were multiplied together in order to come up with two 300-dimensional vectors, one for the headline and one for the article. The cosine similarity was then found between the vectors. The product was used instead of adding or averaging the token word vectors because it performed better. The un-stemmed tokens were used to see if a token exists in the Google News pre-trained vocab. If not, then that token was replaced with an "<UNK>" token which was randomly initialized as noise with 300-dimensional vector between -1 and 1.

5. KL-Divergence

The KL-divergence for each headline and article pair was calculated. For each training, validation, and test headline and article pair, the stemmed tokens of the headline were used to create the unigram query language model while the tokens of the article were used to create the unigram document language model. Each pair has its own query and document language model. Dirichlet smoothing was applied to the document language model in order to estimate probabilities for missing headline words in the article. For the collection language model in the interpolated model, the entire training article was used to count the occurrence of the unique vocab words. The full equation is show below. μ is set to be 1000, N is the number of word occurrences in the article, and |C| is the total number of tokens in the training articles.

$$\mathrm{H}(\mathrm{M_Q}||\mathrm{M_D}) = -\sum_{w} \frac{f_{w,Q}}{|\mathrm{Q}|}.\log\left(\frac{N}{N+\mu}.\frac{f_{w,D}}{|D|}\right) + \left(\frac{\mu}{N+\mu}.\frac{f_{w,C}}{|\mathrm{C}|}\right)$$

KL-divergence was a good indicator of distinguishing unrelated and related pairs. Completely unrelated pairs had zero occurrences of a headline token in both the article and the entire training collection. This caused the value to be infinity (log(0)). A lower KL-divergence means that the two probability distributions are closer and indicates that the headline and article are related in some sense.

6. Alternative Features

Two more features were also implemented without the use of libraries: TF-IDF and BM25.

TF-IDF is a feature which was used by all the top 3 winners of the FNC challenge. The term frequency portions computes how many times a query token appears in the article text while the inverse document frequency captures the notion of how discriminative a term is. The IDF was calculated using the number of unique training articles each stemmed vocab token appeared in. N is 1683 training articles. If a term occurs in many articles then it is probably a common word and has a lower IDF weight whereas a token which occurs rarely has a higher weight. For example, the words "said" and "report" have the lowest IDF weights with values 0.18 and 0.20 respectively. That is as we expect because these words have a high probability of occurring in news articles which indicate that they act as corpus specific stop words. On the other hand, the stems "alphanumer" and 'ai' have IDF weights of 3.22 since they rarely appear in the training articles.

$$\mathrm{TF.IDF(Q,D)} = \sum_{t\in(Q\cap D)} tf_t.\log\left(\frac{N}{n_t}\right)$$

BM25 is normally used for probabilistic ranking and is based on the binary independence model but it can also give a sense of how related the headline is to the article. Large values indicate that the headline and article are more closely related. The value for each headline and article pair is calculated using the equation below. k_1 to 1.2, k_2 to 100 and b to 0.75. K is calculated using those values and the average article length which is 216.

$$\mathrm{BM25(Q,D)} = \sum_{i\in Q} \frac{1}{\frac{n_i+0.5}{N-n_i+0.5}}.\frac{(k_1+1).f_i}{K+f_i}.\frac{(k_2+1).qf_i}{k_2+qf_i}$$

7. Distribution of 2 Important Features

Four features have been discussed: cosine similarity of term frequency BoW vectors, KL-divergence, tf-idf, BM25. The figure below shows distributions on the training data for the four stances for each of the four features. The minimum and maximum values are shown by the vertical line whereas the mean feature value for each of the stances is shown by a small horizontal line.

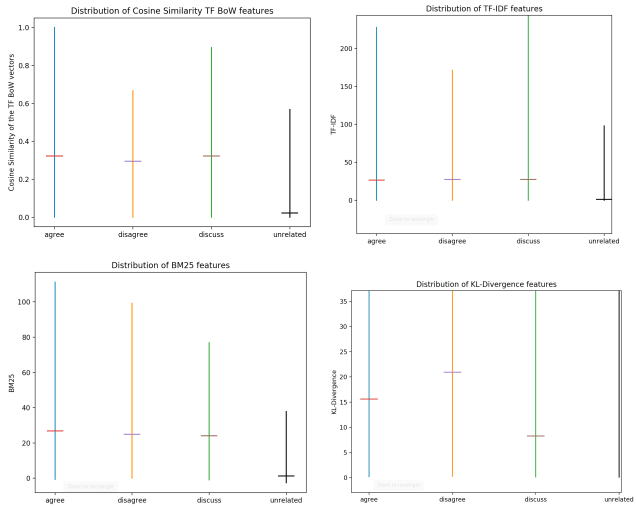

Figure 1: Distribution of the various features for the four stances.

For the distributions of the cosine similarity between the TF BoW headline and article vectors, we can see that the average cosine similarity for the "unrelated" stance is close to 0 while for the related stances it is much greater (which indicates that the article is somehow related to the headline). Furthermore, the maximum values for the "agree" and "discuss" stances are very close to 1 and their average values are higher than that of the "disagree stance".

For the TF-IDF distributions, we can see that it clearly discriminates the related and unrelated stances since the average value of the TF-IDF for the unrelated stances is close to 0 but is not able to further discriminate pairs in the related stance. The maximum value of the TF-IDF for the "disagree" stance is much

lower than the values for the "agree" and "discuss" stances but the distribution of the average values are more indicative of how the models will perform. Similarly, for the BM25 distributions, the values for the unrelated stance is close to zero but it is hard to distinguish between the other stances. The value for the agree stance is slightly higher but they are all virtually the same.

The distribution of the KL-divergence shows different average values for the related stances. The average value for the unrelated stance is very high and can be thought of being infinity. Smaller values indicate that the headline and article are more closely related, which is the case in our results. The average values of the KL-divergence for the "agree" and "discuss" stances are much lower than that of the "disagree" class.

Based on the results above, I conclude that cosine similarity between the term frequency BoW vectors and the KL-divergence are important features for stance detection and can help the model. The former is able to distinguish between the unrelated/related stances while the latter is able to discriminate between related stances even further.

8. Linear Regression and Logistic Regression

For multi-class classification in logistic regression, the one-vs-rest (OVR) method was implemented. The cross-entropy log loss was used as the cost function and the models were trained by adjusting the weight parameters to minimize the cost function. Regularization was also implemented in gradient descent to avoid overfitting. The best parameter was found by performing grid search and comparing the evaluation metrics on the validation set. We found the value to be 0.5 in our case. Four binary logistic regression models were trained, one for each of the four stances. Therefore, for each model training, one of the stances was set to class 1 while the other three stances were set to class 0. The four models resulted in a four-dimensional output prediction vector for each (headline, article) test sample, representing the probabilities of each class. The class with the highest probability was selected as being the stance for that sample pair. For linear regression, decision rules were created in order to round the regression predictions up or down to the nearest class.

Table 2. Decision Rules for Linear Regression Output

Predicted Value	Rounded Value
[0, 0.5)	Stance 0 (unrelated)
[0.5, 1.5)	Stance 1 (Discuss)
[1.5, 2.5)	Stance 2 (Agree)
[2.5, ∞)	Stance 3 (Disagree)

9. Evaluation Metric and Model Performance

Various evaluation metrics were used to assess the performance of the linear regression and logistic regression models. All evaluation metrics were implemented from scratch. The models were initially assessed by accuracy. However, that is not very indicative of the model's performance since the dataset is severely imbalanced and predicting all the test sample pairs as being "unrelated" will give an accuracy of 72%. Therefore, the weighted F1 score was also used as the evaluation metric. The weights were chosen as the ratio of the four classes in the training set. The F1 was computed using the precision and recall for each class.

Furthermore, in order to compare the model's performance with the FNC-1 challenge winners, we also used the FNC scorer metric which is shown in Figure 2. The score is increased by 0.25 if the relatedness portion is predicted correctly, while the other 0.75 is for the correct prediction between agree, disagree, and discuss stances. The score is then normalized by the maximum possible score. We can see that the majority of the score is given upon predicting the correct relationship while discriminating between related and unrelated is less weighted, which is appropriate for our imbalanced dataset.

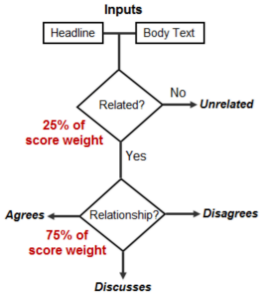

Figure 2: FNC-1 Score Metric for Stance Detection. [1]

As a final additional metric, the confusion matrix along with the precision and recall of each class was computed in order to exactly locate the strengths and weaknesses of the model such as figuring out which classes are predicted poorly. This helps in adjusting the features to make the model more robust.

The input into the logistic and linear regression models are the previously discussed four features. All features were normalized to have 0 mean and unit standard deviation. The learning rate affects the rate at which the cost function decreases during the gradient descent parameter update. A small learning rate causes the training to require a large number of iterations in order for the cost function to converge and reach a global or local minimum. On the other hand, a very large learning rate might either cause the cost to not decrease steadily and slowly or not decrease at all. Figure 3 shows two different learning rates for the linear and logistic regression models. We can see that a learning rate of 0.01 for logistic regression decreases the cost very slowly and even after 600 iterations, the cost function has not converged. On the other hand, a learning rate of 0.5 causes the cost function to converge quickly. The model performance is also affected by the learning rate since if the cost function is not at its lowest, then the weight parameters are not optimal. The weighted F1 score and confusion matrices show that the logistic regression models with a larger learning rate performed better. The models with a smaller learning rate did not make any attempts to predict the "agree" and "disagree" stances. For the linear regression, the cost converges with both learning rates so it doesn't affect the model performance.

Table 3 also shows the performance of the two models based on the discussed evaluation metrics. The rows and columns of the confusion matrices are the same as the ones in Figure 3. Both models achieved a higher FNC score than the baseline, which got 75% on the test set. We can see that the logistic regression model achieved a higher accuracy. However, that is a result of predicting more of the test samples as being the majority "unrelated" class. The linear regression model achieved a higher weighted F1 and FNC score. Looking at the confusion matrix, we can see that the logistic regression did not predict any test sample pairs as being a

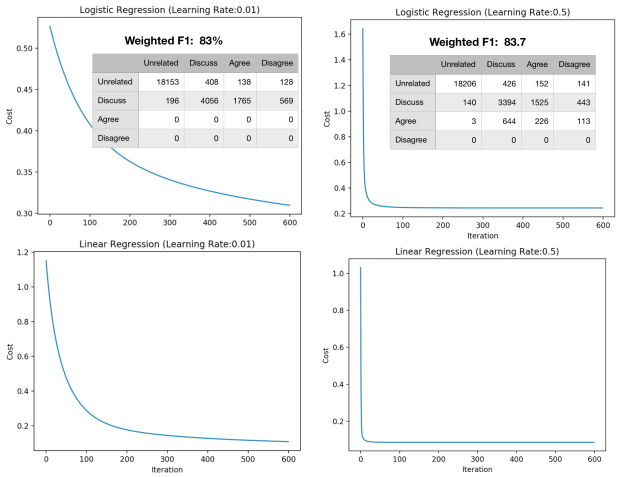

Figure 3: Effect of learning rate on the cost value and model performance

Table 4. Performance of the Models

	Accuracy	FNC Score	Weighted F1	Confusion Matrix
Logistic Model	85.8	75.8	83.6	18206 453 154 151 141 3375 1523 434 2 636 226 112 0 0 0 0
Linear Model	84.8	75.9	84.3	17999 208 76 70 348 2714 960 440 2 1517 846 183 0 25 21 4

"disagree" stance. This is due to the "disagree" stance being a minority class and appearing in only 1.5% of the training set. On the other hand, the linear regression model did predict some of the "disagree" stances correctly and also predicted more "agree" classes than the logistic model. To conclude our results, the linear regression model performed slightly better and table 5 shows the precision and recall of each class for the linear model. Both models perform very poorly for the "disagree" class.

Table 5. Precision and Recall for Each Class for the Linear Model

	Precision	**Recall**	**F1**
Unrelated	98%	98%	98%
Discuss	60.8%	60.7%	60.8%
Agree	33.2%	44.45%	38%
Disagree	8%	0.5%	1.07%

10. Feature Importance

To assess the contribution of each feature, the models were trained with each of the four features independently and the FNC score was recorded, as shown in Table 6. Based on the results, we can see that the cosine similarity between the headline and article BoW vectors and the BM25 features are able to contribute the most to the models.

The FNC score achieved by the logistic regression model for the BM25 feature alone is relatively high (78%). However, upon looking at the confusion matrix, this is due to the model predicting more of the "unrelated" and "discuss" stances correctly but not even attempting to predict any "agree" or "disagree" classes. In fact, the precision and recall for the "disagree" class was zero for all the logistic regression model with each of the four features. However, the BM25 feature helped the linear regression model achieve a recall of 5.2% for the "disagree" class which is much higher than the previous results.

Table 6. FNC Scores for the Models using each Feature Independently

Feature	**Logistic FNC Score**	**Linear FNC Score**
Cosine Similarity of TF BoW	74.5%	74.1%
TF IDF	68.3%	67.15%
KL Divergence	70.6%	73.9%
BM25	78.2%	73.8%

Using the logistic regression model along with the cosine similarity and BM25 features, the performance of the model increases compared to using all four features, as shown in Table 7. However as previously mentioned, this boost in result is due to the model predicting more of the samples as "unrelated" and "discuss" and since they are the top two majority class, the FNC score increases.

Table 7. Performance of Logistic Regression with Cosine Similarity and BM25 Features

	Accuracy	FNC Score	Weighted F1	Confusion Matrix
Logistic Model	87	77.8	83.7	18220 418 152 138 127 3826 1677 514 2 220 74 45 0 0 0 0

11. Improvements

Various improvements were implemented such as using more features and various models. The term frequencies of the 5000 most frequent training article tokens were extracted for each headline and article pair in order to represent each by a 5000-dimensional vector. Latent Dirichlet Allocation was then trained on the training articles to extract 300 topics. LDA was then applied to the headline and article vectors and the cosine similarity between them was computed and used as a feature.

An ensemble model consisting of a gradient boosted decision tree and a multi-layer perceptron was used in order to predict the test stances. The weight given to each model's prediction was equal. Xgboost was used for the decision tree with 300 trees, each with a depth of 3. The multi-layer perceptron had 2 dense layers, with 128 hidden units, and 2 dropout layers with 0.5 probability for regularization to avoid overfitting. Relu activation was used for

non-linearity and Adam optimizer was used for training. The validation data was used to optimize the weights of the multi-layer perceptron. The result of this ensemble model on the test data is shown in Table 8. The FNC score is 5% higher than the baseline model and is also very close to the top 3 winners. The downside is that the precision and recall for the "disagree" class is still 0.

Table 8. Performance of Equally Weighted Ensemble (Xgboost & MLP) Model

	Accuracy	FNC Score	Weighted F1	Confusion Matrix
Xgboost +MLP Model	87.36	80.03	84.15	18030 183 64 61 317 4119 1786 607 2 162 53 29 0 0 0 0

A neural model which was mentioned in [5] was also attempted in order to see if it will perform better than the previous models which require feature engineering. The headline and article were encoded using a two-layer LSTM network. The encoded outputs were concatenated and fed into a multilayer perceptron for classification. The architecture of the implemented model is shown in Figure 4.

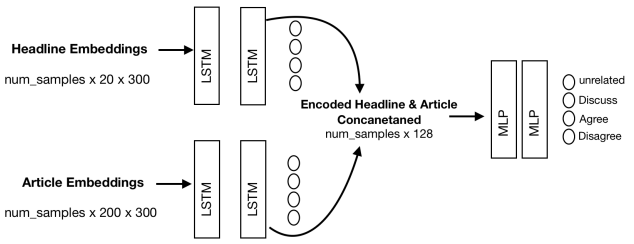

Figure 4: Architecture of the neural model implemented for stance detection.

The Word2Vec, pre-trained on Google News, 300-dimensional word embeddings were used. Out of vocabulary tokens were represented by a 300-dimensional random vector and padding tokens were represented by an array of zeros. A maximum of 20 tokens was used for the headlines and a maximum of 200 tokens was used for the article bodies. In both cases, if the text was longer than the limit, it was truncated and if it was shorter, it was padded with padding tokens.

A two-layer LSTM, with 64 hidden units in each layer, was trained using the previously mentioned headline and article embeddings independently of each other. After training, the output of the intermediate layer (the second LSTM layer) was used as the encodings. The encoded headline and article vectors were then concatenated and a 2-layer multi-layer perceptron, with 128 hidden units and 2 dropout layers, was used as the final classifier. The MLP was also trained again. The results of the model are shown in Table 9.

Table 9. Performance of the Neural Model

	Accuracy	FNC Score	Weighted F1	Confusion Matrix
Neural Model	72.2	39.4	60.5	18349 4464 1903 697 0 0 0 0 0 0 0 0 0 0 0 0

Based on the above results, we can see that the model did not learn anything from the encodings and just predicts the majority class in all the cases. Different training parameters and number of LSTM layers in the encoding network might yield different results. However, due to the large training time required to train the article encoding network, only one the previously mentioned architecture was implemented. Furthermore, for the maximum length of the article, we only chose 200 tokens. Therefore, since many of the articles were much longer than 200 tokens, they were truncated at 200 words. As a result, the encoding of the articles may not be true representations of the articles.

From my results, I conclude that feature engineered approaches combined with neural model yielded better results than only using neural models to learn features automatically.

12. REFERENCES

[1] Fake News Challenge Stage 1 (FNC-I): Stance Detection. Retrieved April 8, 2018 from http://www.fakenewschallenge.org/

[2] FakeNewsChallenge/fnc-1. (June 2017). Retrieved April 8, 2018 from https://github.com/FakeNewsChallenge/fnc-1/

[3] Sean Bird, Doug Sibley, and Yuxi Pan. *Talos targets disinformation with Fake News Challenge victory*. 2017

[4] Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, and Felix Caspelherr. *Description of the system developed by Team Athene in the FNC-1*. 2017

[5] Qi Zeng, Quan Zhou, Shanshan Xu. *Neural Stance Detectors for Fake News Challenge.* 2017.