An Exploration Into Improvement of the FNC-1 Baseline Program
COMP9417 Machine Learning Project 2019T2

Tyson Subasinghe z5167970, Timothy Gonzales z5131551, Peter Hehir z8372276, David Shvarts z5112955

Summary

By modifying the baseline program of the FNC-1 competition using a culmination of a variety of techniques, our team was able to attain a result which would have placed us 12th out of 50 or more competitors. The following report contains a synopsis of our findings, with further results and evidence in an appendix appended to this document.

Context

The FNC-1 (Fake News Challenge 1) was a competition run in 2016 with the goal of creating a Stance Detection program which utilised a machine learning algorithm to determine whether the headline of an article agrees with, disagrees with, discusses or is unrelated to the content of the article's body. It was envisaged that this Stance Detection could be fed into a pipeline of larger fact checking programs which can predict whether the article was real or fake.

Analysis of the Training Data

The training data consists of 1,683 articles and 49,972 headlines related to these headlines. Headlines and articles are related by a Body ID. Headlines also include a stance which can take the values *agree, disagree, discuss* (collectively, the *related* stances) and *unrelated.*

The distribution of these values in the training data is

Stance	Frequency	Percentage
Agree	3,678	7.54%
Disagree	840	1.68%
Discuss	8,909	17.83%
Unrelated	36,545	73.13%

The distribution is highly skewed with many more *unrelated* stances and, among the *related* stances, many more *discuss* stances. This encourages classification models that are not practical but maximise the score. For example, consider a classifier that predicts only *unrelated* and *discuss.* If one can achieve a 99% accuracy classifying *unrelated* and *related*, then it will achieve a score of 79.17%. This is the same argument as the cancer lab test where the application of Bayes rule means there is still a higher likelihood of not having cancer even if the result is positive.

Analysis of the Baseline Implementation

The provided baseline implementation included code for pre-processing text, splitting data carefully to avoid bleeding of articles between training and test, k-fold cross validation, and scorer. The hand-crafted features include word/ngram overlap features, and indicator features for polarity and refutation.

The text files are pre-processed and results saved for reuse to speed up testing different model parameters. The articles are separated into a training set and a holdout set. The training articles are further divided into k-folds. The headlines/stances are also separated into the same groups as their associated articles.

The baseline implementation was able to achieve a score of 3538 out of 4448.5   (79.53%) on the holdout dataset.

The baseline implementation was able to achieve a score of 8761.75 out of 11651.25 (75.2%) on the competition test dataset versus the winning score 9556.0 (82.02%). We ran this at the end of our assignment.

After analysing and reviewing the baseline, below are the key strengths and weaknesses of the features used:

Strengths

- The baseline program does a significant amount of linguistic preprocessing before any features are generated. Techniques such as removing stopwords reduces both noise and amount of data feed in the model.. Other techniques such as lemmatization and lowercasing transform the words into a standard format. Lemmatization also acts as a first step in identifying similar words.
- The baseline program not only relies on comparing single instances of words between the headlines and bodies but also looks at comparing consecutive characters and words. Looking at consecutive characters and words in the given dataset is helpful as English is a very structured language wherein the ordering of words have an impact on how it is interpreted. Having this feature helps identify double negative words or word expressions that a simple word frequency feature cannot do.

Weaknesses

- As seen in the feature importance chart , there are a lot of features that do not contribute anything to the learning model. These features creates noise and is a waste of the model's run time and memory usage (*Medium, 2019*).
- The feature refuting words uses a small amount of preselected words which does not cover all possible words with a refuting tone. Having a preselected set of refuting words would result to the features effectiveness to vary among different datasets of headlines.
- All of the features used in the baseline is focuses on the frequency of words in both the headline and text but does give weight on the relevance of each word.

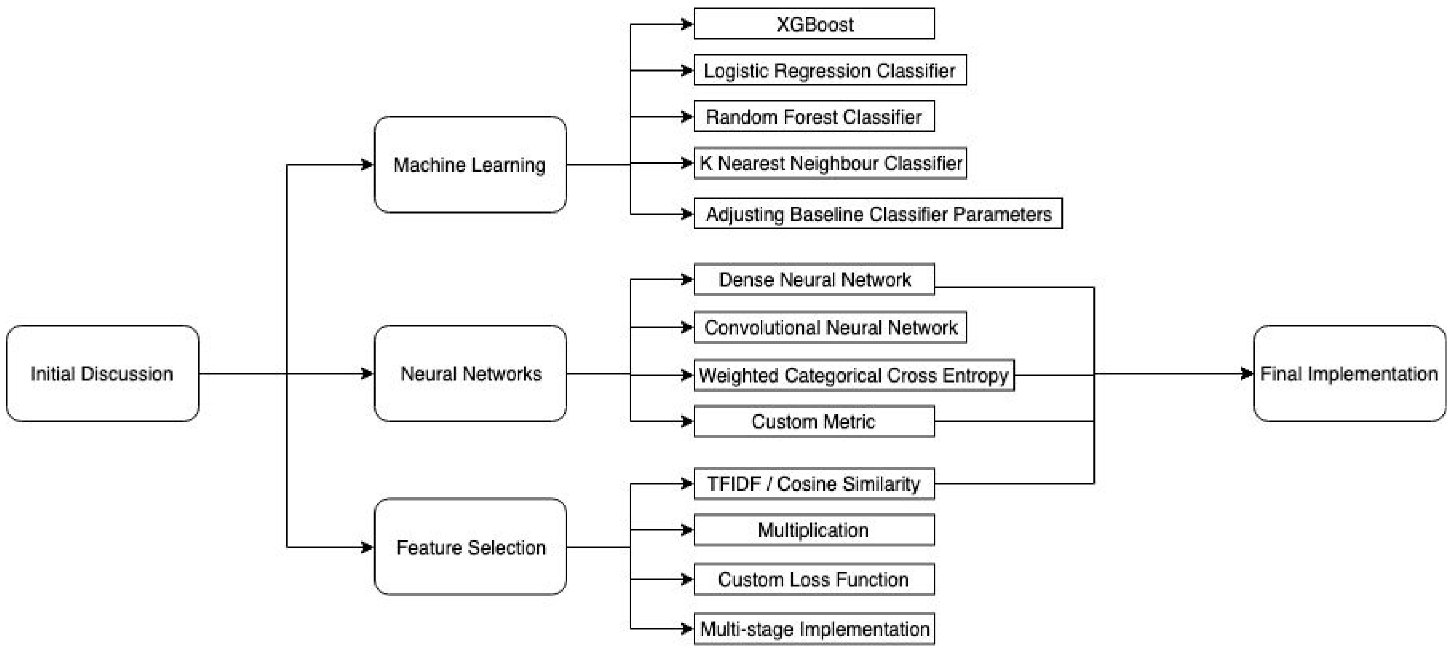
Assignment Approach

We wanted to mimic the competition conditions and so we removed the reporting of competition test data from the baseline solution and no decisions were based on the test data.

We also decided to use the baseline solution as our starting point. The team could then investigate independent improvements which could then be merged into the final implementation.

The primary investigations conducted were on:

1. Machine Learning
2. Feature Selection
3. Neural Networks
4. Combining Different Methods


Initial Discussion
Machine Learning
Neural Networks
Feature Selection
XGBoost
Logistic Regression Classifier
Random Forest Classifier
K Nearest Neighbour Classifier
Adjusting Baseline Classifier Parameters
Dense Neural Network
Convolutional Neural Network
Weighted Categorical Cross Entropy
Custom Metric
TFIDF / Cosine Similarity
Multiplication
Custom Loss Function
Multi-stage Implementation
Final Implementation

Machine Learning

GradientBoostingClassifier

This is the model in the baseline implementation. A grid search of the n_estimators and max_depth parameters resulted in setting these to 100 and 4 respectively (from 200 & 3) but only improved the accuracy by less than 0.5% to 79.81%

Logistic Regression

We also experimented with different solvers within the logistic regression classifier. As the problem is multiclass, sklearn LogisticRegression only provides 'newton-cg', 'sag', 'saga' and 'lbfgs' as valid solvers. Each of these were implemented and the experiment was performed using the standard class weights of [4, 4, 4, 1].

Results for Different Solvers:

newton-cg:	80.347%
saga:	80.235%
lbfgs:	80.145%
sag:	80.121%

These results all surpassed the baseline of 79.53% in a relatively short time with a maximum score increase of approximately 1.0. Newton-cg was the best performer, which can be explained as it uses Newton's Method with a quadratic approximation of the cost function, which is one of the best forms of approximation, however it is relatively computationally more expensive to calculate.

There is not much other distinction in the top performers. Basic L1 and L2 regularization reduced the scores for this dataset.

Random Forest Classifier

We tested this model but found that it produced similar results (79.16%), but offered no improvement when compared to the baseline's GradientBoostingClassifier, and thus this classifier was not explored further.

K Nearest Neighbours

To check the performance of other classifiers, we ran KNN using k = 3, 5, 10. Results were

k	**Dev Set Score (%)**
3	74.35
5	75.83
10	77.28

This did not present any improvement over the baseline's classifier, and thus this was not explored further.

XGBoost

This is another implementation of the GradientBoostingClassifier. The results (accuracy = 79.64%) were consistent with the GradientBoostingClassifier (79.53%).

A grid search of the best n_estimators parameter improved the accuracy to 79.65% when n_estimators = 50.

Conclusions

The GradientBoostingClassifier and XGBoost proved to the best of the standard classifiers.

Classifier Improvements

We looked at several techniques to improve the performance of the classifiers. These use the classifiers we examined above but extend them in the following ways.

Weighting the Input Data

The scoring function gives 4 times more value to a correct prediction for a *related* than an *unrelated* stance. Neither GradientBoostingClassifier nor XGBoost allow weighting of the training data. To give the *related* stances more weight that the *unrelated* stance we appended the training data with 3 additional copies of each headline with a *related* stance. The improved the accuracy to 81.42% for the GradientBoostingClassifer and 81.10% for xgboost an increase of nearly 2%.

Customised Loss Function

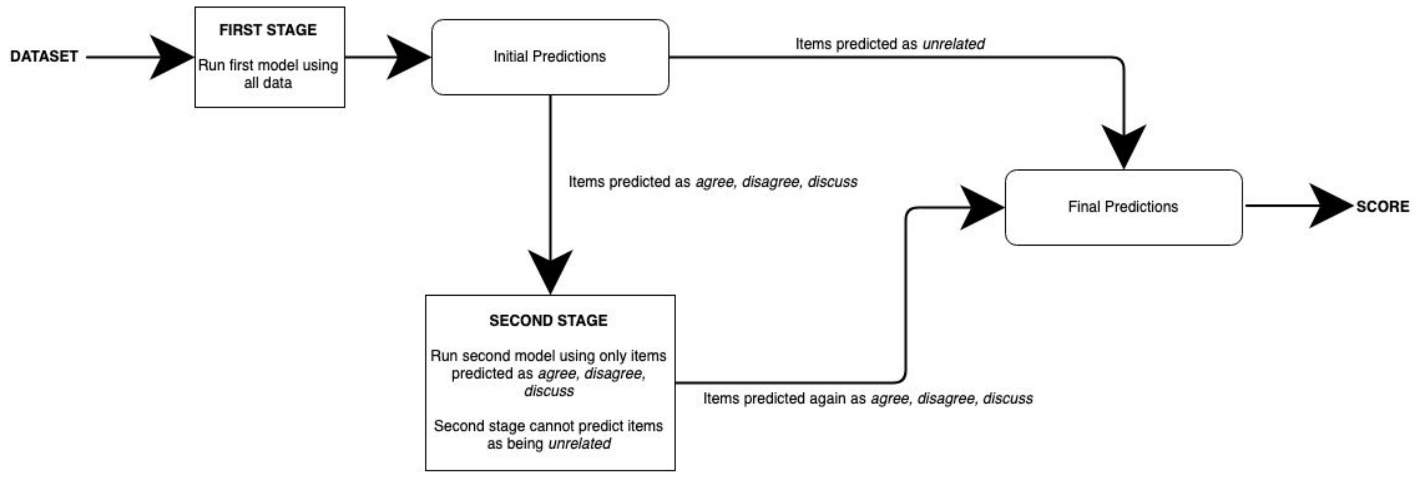
We integrated our own softmax function with XGBoost. This objective function exactly matched the results of XGBoost using the inbuilt objective function (softprob).

XGBoost allows one to change the gradient/hessian of the loss function but not the loss function itself. We were unable to improve the accuracy by Increasing the gradient/hessian for the *related* stances without also modifying the loss function itself.

We also customised the prediction method to make it more likely to predict a *related* stance by predicting *unrelated* only if its probability was > 0.5 and predicting the *related* stance with greatest probability otherwise. This approach performed poorly compared to the standard predictor.

Multi Stage Model

Due to the distribution of data within the training dataset, it was thought the large number of *unrelated* stances would influence the classification of the *related* stances. We trained two model, the first using all the data and the second using just the *related* stances. The second model was used to predict the stance if the first model predicted it was a related stance. This approach made almost no improvement to the accuracy compared to the single stage model (81.13% versus 81.10%)


DATASET
FIRST STAGE
Run first model using all data
Initial Predictions
Items predicted as *unrelated*
Items predicted as *agree, disagree, discuss*
Final Predictions
SCORE
SECOND STAGE
Run second model using only items predicted as *agree, disagree, discuss*
Second stage cannot predict items as being *unrelated*
Items predicted again as *agree, disagree, discuss*

It can be concluded that the weighting of the input data provides the only significant improvement to the standard model.

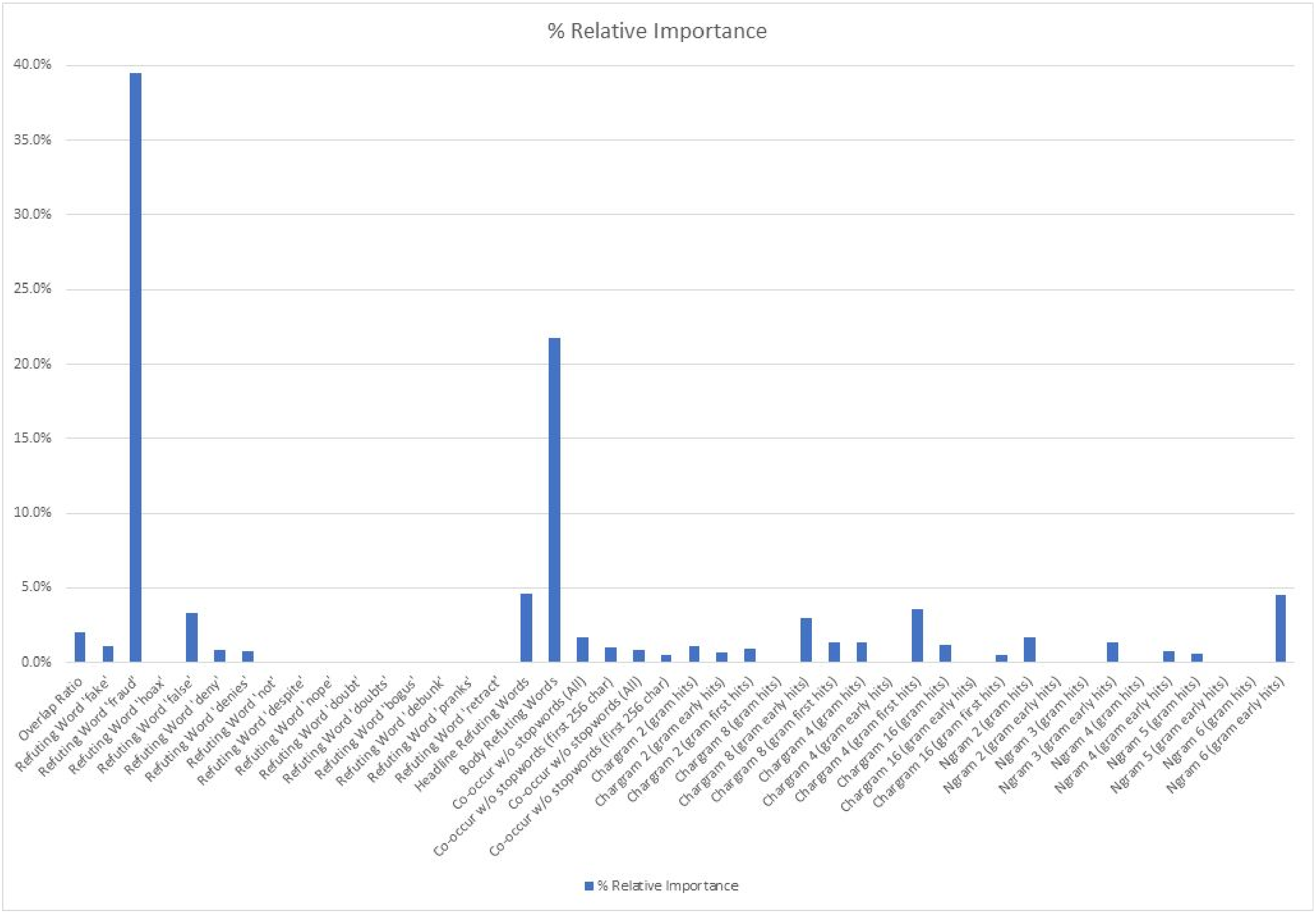
Feature Selection

Aside from building the right model, determining the right features to use to train the model is one important step to achieve an accurate stance detection classification (*Comp 9417 Classification Slide, 2019*)

The goal of this experiment is to transform the given dataset into a set of features that the model can easily learn and interpret to produce better accuracy in classification.

Since the competition has given us a baseline program to start with. The first approach of the experiment is to review and understand how the baseline transformed the training data sets into useful features. The analysis of the features are discussed in the first part of the document.

After running the baseline model using XGboost, below is a bar graph that shows the importance of each feature:



Reducing Dimensionality of Features

The next approach is to determine the importance of each feature in the model and remove irrelevant features. Irrelevant features negatively impacted model because adding more features increases the dimensions and makes the data sparse which will lead to unreliable results in the model (Medium, 2019). Removing features also improves the training time of the model.

After looking at the importance of each attribute, features that had a 0% importance were removed.

The amount of features that have 0% importance in our model may affect the model's efficiency and accuracy. To remedy this, we remove unimportant features. After running the model without those features, the model provided the same run time and predictions.

This disproves our hypothesis. The reason that removing unimportant features had no effect because the gradient boosting classifier in the baseline utilises a method that ensembles multiple weak decision trees to produce a better performing decision tree. Removing irrelevant features in this model does not have any effect on the learning of the model as the model greedily selects the most importance features in splitting the tree (*DeepAI, 2019*).

Cosine Similarity with Bag of Words

Another way of improving the current baseline model is to add more relevant features in the model. After researching various text features, cosine similarity was chosen as an appropriate feature to be added to the baseline. The reason for this is that cosine similarity is effective in determining the distance/similarity of text even if there is a huge difference in the number of words used *(Machine Learning Plus, 2019*).

Since adding new features is another way of improving the accuracy of the model, a cosine similarity feature was added to the model. This uses a count vectorizer to convert words into vectors. Running this version of the model showed a score increase of 2% (from 79.2% to 81.2%) in the holdout set.

The cosine similarity feature is typically used in sentiment analysis models and thus improved our model because analyses the sentiment of the article (as a vector direction) and headline as opposed to just vector distance (*Quora, 2019*)

Cosine Similarity with TFIDF

The next approach was to find ways to improve the cosine similarity feature. The way we decided to improve this feature was to look at different ways to transform text data into vectors. The original feature used a “bag of words” transformation which converts text data to a vector of the frequency of words.

Using bag of words transformation becomes less effective when looking at similarities between different sets of text as this transformation ignores word importance and only considers word frequency.

To negate this, a better transformation called Term Frequency Inverse Document Frequency (TFIDF) is used to improve the cosine similarity feature *(Kdnuggets.com, 2019)*.

After replacing the count vectorization / bag of words transformation with the TFIDF transformation, the baseline code with cosine similarity feature produced the below results:

Adding this new version of the cosine similarity feature, an increase of 2.1% in the dev set compared to the baseline program, which is a substantial improvement.

Using TFIDF results in higher weights given to more unique terms that give more ‘meaning’ to the text, and it was able to improve the model accuracy significantly because it counts similarities in words both inside the article body and between the headline and the body, as opposed to just counting word similarities like count vectorisation (*Beyondthelines.net, 2019*).

From the original baseline, adding a cosine similarity feature using the TFIDF transformation has made a substantial improvement in the accuracy of our model.

Neural Networks

The initial idea for using a neural network arose after coming to the conclusion that common machine learning methods like the gradient boosting classifier used in the baseline may not perform at the same level as a deep learning method such as a neural network when something as abstract as the ‘stance’ of an article is being determined.

Whereas machine learning cannot understand the nuances of english language and the complexity of being ‘related’ to something else, multi-layer deep learning networks have the capacity to learn when relationships cannot be described linearly. This greater flexibility lead to us testing the viability of using a neural network to attain better results.

Dense Neural Network

Using the Keras API with TensorFlow as the backend, basic Neural Networks were constructed in an attempt to improve over our machine learning models. TensorFlow was compiled from source to enable the use of extra CPU features. GPU acceleration was not used.

In the initial experiment, the features used were again kept the same as the baseline to examine the effect of neural networks compared to the machine learning methods tested.

The initial neural network that provided some increase was constructed as a basic Dense Neural Network (DNN) with an input layer of the length of the baseline features (44), one hidden layer consisting of 32 perceptrons, and an output layer of 4 neurons with softmax activation for categorical cross-entropy loss.

Using a Keras Tokenizer, the features were replaced with TFIDF analysis of the top 1000 words in the headline and the body concatenated together. This was replaced later with a standard FREQ analysis in the same dimensions, which provided better results. To compensate for the increase in the feature dimension, the model was expanded to contain 512 nodes in the input and hidden layer.

To avoid overfitting, Dropout and L2 regularisation on bias and kernel were added. Dropout is a popular regularisation technique commonly applied to DNNs (G. Hinton et al., 2012) and can significantly improve generalisation. The new preprocessor was also multithreaded to speed up development.

Features found in our feature analysis were then concatenated back onto the feature vector along with the baseline features. however the model quickly began displaying issues similar to the aforementioned machine learning classifiers. Specifically, the issue of items rarely being predicted as having a stance of disagree was present. Additionally, a custom loss function (weighted cross-entropy) and a custom scoring metric based on FNC-1 were integrated into the model (see next sections). Finally, to take advantage of the new metric, Model Checkpointing and reloading was added. After some minor tuning to minimise potential generalization, this became our final DNN model.

Weighted Categorical Cross Entropy

A custom loss function to relate the performance of the model closer to the uneven distribution of the competition scoring function was needed.

A solution was found in a GitHub issue under the official Keras repo. This applies a weight matrix to the outcomes of the standard categorical cross-entropy function based on a multiplicative penalty for certain results, effectively individually penalizing true/false positives/negatives.

Weighting categorical cross-entropy steers the loss function more in line with the FNC-1 scoring system but cannot represent exact scores as the weights are scalar vector multipliers, which lead to minor gains.

Custom Metric

To practically analyse the performance of the model in real time, a custom keras metric was developed to add a score that almost exactly matches the default scoring function. Due to the intricacies of the Keras backend and how it handles Tensors, this was accomplished by unconventional means. Code can be found in `utils/score_loss.py`.

```
Reduce each of the batch tensors (3D) to argmax tensors (2D).

For each score category (correct/incorrect related/unrelated):

        Reduce argmax tensors (2D) to boolean tensor (1D)

        Cast boolean tensor to int32 tensor

        Sum int32 tensor (0D)
```

Convolutional Neural Network

As we can describe the data in the form of an analysis on the headline and the body, a two-dimensional Convolutional Neural Network was seen as appropriate to fit the data, despite being applied generally to image data. The X input was organised in two dimensions containing freq analysis results, one for the headline, and one for the body. Results are recorded in the appendix.

Final Program

The final program used a dense neural network, and incorporated weighted categorical cross entropy, a custom metric, and used TFIDF.

The final program produced the below results for the holdout dataset:

Holdout Score: 3882.75 out of 4448.5 (87.2822299651569%)

	Agree	Disagree	Discuss	Unrelated
Agree	**495**	**0**	**244**	**23**
Disagree	**84**	**0**	**77**	**1**
Discuss	**201**	**2**	**1537**	**60**
Unrelated	**27**	**1**	**75**	**6795**

The final program produced the below results for the competition dataset:

Competition Score: 9208.0 out of 11651.25 (79.03014697993777%)

	Agree	Disagree	Discuss	Unrelated
Agree	**918**	**2**	**844**	**139**
Disagree	**296**	**0**	**271**	**130**
Discuss	**859**	**1**	**3243**	**361**
Unrelated	**136**	**1**	**297**	**17915**

These results would have placed us 12th in the official FNC-1 out of at least 50 competitors.

Discussion

Real World Usability of Our Program

Real world applications of our findings are limited due to the scope of the competition. Despite being one of the highest scores, 79.03% is not an acceptable real world accuracy.

As well as this, the few discrete stance categories are not accurate reflections of the stances headlines can take in the real world. For example, instead of being agree or disagree, a headline may strongly disagree or mildly agree; it is more akin to a continuum.

Real world distributions of headline stances are also vastly different to those provided in the competition. In the competition, 73.13% of headlines had an *unrelated* stance, whereas in the real world this number would be much less, as the ratio of real to fake news articles would not be the same. Additionally, the chance of a fake news article having a headline vastly different from the body text in real life is also quite low.

In real world applications, there would also be more parameters from which our program could be improved. For example, source of the headline (website), popularity (likes/views/shares). Real world applications such as Facebook's fake news filter, which may have used headline stance as a parameter to determine authenticity, would have had access to these parameters and so would be able to outperform our program due to having access to more useful data.

Deep vs Machine Learning

Please note that for the scope of this assignment "deep learning" encompasses neural networks and "machine learning" refers to commonly used machine learning classifiers such as random forest, gradient boosting, k nearest neighbours, etc.

Neural networks provided a significant improvement in the accuracy of the program over machine learning techniques tested. After modifying the feature selection process, the difference in improvement was reduced, but even so, neural networks always outperformed our implementation of machine learning algorithms.

These findings reflect a statement by Dean Pomerlau, one of the organisers of FNC-1, who stated "we quickly realized machine learning just wasn't up to the task" *(The Verge, 2019)*

Machine learning models are not able to handle abstract information such as text easily since they have a hard time distinguishing context of words, and cannot understand irony and other literary techniques often employed in articles. In neural networks, however, feature extraction can be done by network itself and therefore can 'understand' patterns in meaning better (*Research Gate, 2019*).

Future Work

Based on our findings during the timespan we had to optimise the program, we came to conclusions regarding which avenues of improvement would be most beneficial to pursue in the future and which would not.

Deep learning should be explored further, especially convolutional neural networks, since they excel at finding relationships in 2D array formats, which could easily be implemented to compare headlines and bodies.

On the other hand, echoing the sentiment of Dean Pomerlau, we would likely not explore traditional methods of machine learning (Nearest Neighbour, Random Forest, etc) further, since we found these classifiers to perform worse than the baseline's gradient boosting classifier.

We also feel that the gradient boosting classifier as we have utilised it is near its optimal implementation within the scope of the competition, and thus would not explore this option further.

Work in the field reflects our conclusions, and currently revolves around deep learning, such as at Facebook with their face news classifier rather than their prior machine learning approach.

Conclusion

Through the application of a combination of a neural network and feature selection techniques, our team was able to place 12th amongst the competitors of FNC-1, and our journey to this result led to a deeper level of understanding of machine learning, deep learning and their applications.

References

Bain, M. (2019). *COMP9417 Classification lecture Slide (1)*.

Beyondthelines.net. (2019). *TF-IDF – Beyond the lines*. [online] Available at: https://www.beyondthelines.net/machine-learning/tf-idf/ [Accessed 10 Aug. 2019].

DeepAI. (2019). *Gradient Boosting*. [online] Available at: https://deepai.org/machine-learning-glossary-and-terms/gradient-boosting [Accessed 10 Aug. 2019].

Dictionary.cambridge.org. (2019). *LEMMATIZATION | meaning in the Cambridge English Dictionary*. [online] Available at: https://dictionary.cambridge.org/dictionary/english/lemmatization [Accessed 10 Aug. 2019].

Hinton, G.E., et.al, Improving neural networks by preventing co-adaptation of feature detectors,

Kdnuggets.com. (2019). *WTF is TF-IDF?*. [online] Available at: https://www.kdnuggets.com/2018/08/wtf-tf-idf.html [Accessed 10 Aug. 2019].

Machine Learning Plus. (2019). *Cosine Similarity - Understanding the math and how it works? (with python)*. [online] Available at: https://www.machinelearningplus.com/nlp/cosine-similarity/) [Accessed 10 Aug. 2019].

Medium. (2019). *The Curse of Dimensionality!*. [online] Available at: https://medium.com/diogo-menezes-borges/give-me-the-antidote-for-the-curse-of-dimensionality-b14bce4bf4d2 [Accessed 10 Aug. 2019].

Medium. (2019). *Feature Selection Techniques in Machine Learning with Python*. [online] Available at: https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e [Accessed 10 Aug. 2019].

Medium. (2019). *Word to Vectors — Natural Language Processing*. [online] Available at: https://towardsdatascience.com/word-to-vectors-natural-language-processing-b253dd0b0817 [Accessed 10 Aug. 2019].

Research Gate. (2019). *Why Deep Learning method perform better than other traditional Machine Learning methods*. [online] Available at: https://www.researchgate.net/post/Why_Deep_Learning_method_perform_better_than_other_traditional_Machine_Learning_methods [Accessed 10 Aug. 2019].

The Verge. (2019). *Why AI isn't going to solve Facebook's fake news problem*. [online] Available at: https://www.theverge.com/2018/4/5/17202886/facebook-fake-news-moderation-ai-challenges [Accessed 10 Aug. 2019].

The Verge. (2019). *Why AI isn't going to solve Facebook's fake news problem*. [online] Available at: https://www.theverge.com/2018/4/5/17202886/facebook-fake-news-moderation-ai-challenges [Accessed 10 Aug. 2019].

Quora. (2019). *Advantages of Euclidian distance and cosine distance, respectively*. [online] Available at: https://www.quora.com/What-are-the-advantages-of-Euclidian-distance-and-cosine-distance-respectively [Accessed 10 Aug. 2019].

Appendices

Running Our Models

Our main file is `fnc.py`, which handles flags to clear cache `[-c]` as well as an optional experiment argument in order for the reader to confirm our results `[-e “experiment_name”]`. Our experiments exist in mostly independent files. Our final submission does not require experiment name flags and exists in `fnc_dnn.py`. Programs were developed using Python3 version 3.6.8 and were confirmed working on this version. Please allow sufficient time for preprocessing for some of the later experiments. Steps to initialise the project are as follows:

1. `git submodule init`
2. `git submodule update`
3. `pip3 install -r requirements.txt`
4. `python3 fnc.py`

Experiment names can be found in `fnc.py`.

Competition Details

Competitors were provided a training dataset contains 2 files, one with article bodies and another with the headlines & stances linked by an article number. There are 1,683 articles and 49,972 headlines so there are multiple headlines for each article. In the test data there are 904 articles and 25,413 headlines.

They were also provided a baseline implementation which uses the GradientBoosting Classifier in scikit-learn.

Results were evaluated based on a 2 level scoring system using a separate testing dataset that was never revealed to the competitors. If the implementation successfully classified the stance as unrelated or related (related being one of agree, disagree or discuss) it was given 0.25 points. Further, if it correctly predicted the stance of the article, it was given an additional 0.75 points. The actual score was divided by the maximum possible score to give an accuracy rate.

Competitors were not provided the test dataset but were given 6 opportunities to run their programs on the test dataset 48 hours prior to the close of the competition. This prevented competitors fitting their implementation to the test data.

Scoring Function

The scoring function gives much greater weight to a correct related stance than to the *unrelated* stance. Further, it score 0.25 even if it stance is incorrect but is related. A standard 4 class classification model assumes equal weighting for each classification and all mismatches are treated as errors.

We can show algebraically the scoring function is given by

$$\frac{np}{4}\left[(1-e_1)(4-3e_3)-(1-e_2)\right]+\frac{n(1-e_2)}{4}$$

where n is the number of headlines, p is the proportion of *related* stances in the headlines, e_1 is the false negative error (*unrelated* predicted as *related*), e_2 is the false positive (*related* predicted as *unrelated*) and e_3 is the misclassification error of the *related* stances. The maximum score is when all the errors are zero and is given by $\frac{3np}{4}+\frac{n}{4}$.

If the scoring function was 0.25 in a standard classification model, the score is given by

$\frac{np}{4}\left[(1-e_1)(1-e_3)-(1-e_2)\right]+\frac{n(1-e_2)}{4}$ which reduces to $\frac{n}{4}$ when all the errors are zero.

This difference implies we need to adjust the model to align with scoring function of the competition.

Baseline Model Preprocessing

The baseline first cleaned and standardised the words used in the dataset using the below techniques:

Linguistic Pre-prorocessing Technique	Description
Lowercasing	All words in the dataset are lowercase
Removing stop words	Remove useless words/stop words from the dataset.
Lemmatize	Reducing the structure of words into its based form (Dictionary.cambridge.org, 2019)

After standardising the words, words are then transformed into several feature vectors as per below:

Feature name	Description
Word Overlap	gets the percentage of common words out of the total number of words in the headline and body
Refuting Words	A total of 16 binary features wherein each feature focuses on whether a refuting word is in the headline or body
Polarity	Feature that counts the total refuting words in the headline. Another feature that counts refuting words in the body
Binary co-occurance	After linguistic pre-processing word, count the number of times a word appears from both the headline and body
Char gram	For each consecutive n characters, count the number of times it appears in both the headline and body
N-grams	For each consecutive n words, count the number of times it appears in both the headline and body

Results of the Baseline Model

Running the baseline features with the GradientBoostingClassifier model used in the baseline implementation give the following results with the holdout data:

Score: 3538 out of 4448.5 (79.53%):

	Predicted			
Actual	Agree	Disagree	Discuss	Unrelated
Agree	**118**	**3**	**556**	**85**
Disagree	**14**	**3**	**130**	**15**
Discuss	**58**	**5**	**1,527**	**210**
Unrelated	**5**	**1**	**98**	**6,794**

Running the baseline features with the GradientBoostingClassifier model used in the baseline implementation give the following results with the competition data:

Competition Test Set Score: 8740.25 out of 11651.25 (75.02%)

	Agree	Disagree	Discuss	Unrelated
Agree	**96**	**11496**	**310**	**310**
Disagree	**26**	**1**	**417**	**253**
Discuss	**114**	**2**	**3626**	**722**
Unrelated	**1**	**0**	**335**	**18013**

GradientBoostingClassifier

Hyperparameter Changes: Results

```
Scores on the dev set for clf:
GradientBoostingClassifier(random_state=14128,n_estimators=100,max_depth=4)
```

	agree	disagree	discuss	unrelated
agree	122	6	554	80
disagree	17	3	128	14
discuss	60	5	1533	202
unrelated	5	0	94	6799

```
Score: 3550.25 out of 4448.5   (79.80780038215129%)
```

Random Forest Classifier

```
Scores on the dev set for clf:
RandomForestClassifier(random_state=14128,n_estimators=300,criterion='entropy',n_jobs=-1)
```

	agree	disagree	discuss	unrelated
agree	141	1	529	91
disagree	25	3	118	16
discuss	106	6	1486	202
unrelated	8	0	109	6781

```
Score: 3521.5 out of 4448.5    (79.16151511745532%)
```

Weighting - GradientBoostingClassifier

Score: 3622 out of 4448.5 (81.42%):

	Predicted			
Actual	Agree	Disagree	Discuss	Unrelated
Agree	**124**	**2**	**593**	**43**
Disagree	**17**	**1**	**136**	**8**
Discuss	**65**	**4**	**1,643**	**88**
Unrelated	**17**	**1**	**277**	**6,599**

We were able to increase accuracy by nearly 2% (81.42% v 79.53%) using the same features and model.

Weighting - XGBoost

Score: 3607.75 out of 4448.5 (81.10%):

	Predicted			
Actual	Agree	Disagree	Discuss	Unrelated
Agree	**139**	**2**	**568**	**53**
Disagree	**16**	**4**	**135**	**7**
Discuss	**96**	**3**	**1,602**	**99**

Unrelated	**16**	**2**	**249**	**6,631**

Multi Stage Model - XGBoost

Score: 3609.25 out of 4448.5 (81.130%):

	Predicted			
Actual	Agree	Disagree	Discuss	Unrelated
Agree	**145**	**3**	**561**	**53**
Disagree	**14**	**4**	**137**	**7**
Discuss	**97**	**6**	**1,598**	**99**
Unrelated	**17**	**4**	**246**	**6,631**

Reducing Dimensionality of Features

Dev set Score: 3521.75 out of 4448.5 (79.16713498932225%):

	Agree	Disagree	Discuss	Unrelated
Agree	**79**	**3**	**588**	**92**
Disagree	**12**	**0**	**136**	**14**
Discuss	**37**	**0**	**1550**	**213**
Unrelated	**2**	**0**	**101**	**6795**

Cosine Similarity (Bag of Words)

Dev set Score: 3613.0 out of 4448.5 (81.21838822074857%)

	Agree	Disagree	Discuss	Unrelated
Agree	**86**	**0**	**621**	**55**
Disagree	**14**	**0**	**136**	**12**
Discuss	**46**	**1**	**1623**	**130**
Unrelated	**1**	**0**	**99**	**6798**

Cosine Similarity (TFIDF)

Dev set Score: 3618.0 out of 4448.5 (81.330785658087%)

	Agree	Disagree	Discuss	Unrelated
Agree	**86**	**1**	**627**	**48**
Disagree	**13**	**0**	**141**	**8**
Discuss	**41**	**1**	**1620**	**138**
Unrelated	**1**	**0**	**73**	**6824**

Dense Neural Network

N.B. Results vary slightly due to randomised starting values for weights

```
> python3 fnc.py -c -e “nn_basic”
```

Holdout Score: 3585.25 out of 4448.5   (80.59458244352028%)

	Agree	Disagree	Discuss	Unrelated
Agree	**92**	**0**	**625**	**45**
Disagree	**14**	**0**	**140**	**8**
Discuss	**58**	**2**	**1635**	**105**
Unrelated	**11**	**0**	**293**	**6594**

Dense Neural Network (Weighted Categorical Cross Entropy)

N.B. Results vary slightly due to randomised starting values for weights

```
> python3 fnc.py -c
```

Holdout Score: 3882.75 out of 4448.5   (87.2822299651569%)

	Agree	Disagree	Discuss	Unrelated
Agree	**495**	**0**	**244**	**23**
Disagree	**84**	**0**	**77**	**1**
Discuss	**201**	**2**	**1537**	**60**
Unrelated	**27**	**1**	**75**	**6795**

Convolutional Neural Network

N.B. Results vary slightly due to randomised starting values for weights

```
> python3 fnc.py -c -e “nn_cnn”
```

Holdout Score: 3647.25 out of 4448.5 (81.98831066651681%)

	Agree	Disagree	Discuss	Unrelated
Agree	**283**	**0**	**431**	**48**
Disagree	**105**	**0**	**46**	**11**
Discuss	**229**	**0**	**1456**	**115**
Unrelated	**14**	**0**	**62**	**6822**