COMPGI15: Information Retrieval and Data Mining

Course Project: News Stance Detection

Achillefs Sfakianakis*
(SN: 17074448)
UCL
ucabasf.17@ucl.ac.uk

1. INTRODUCTION

This paper deals with stance detection, which constitutes one aspect of the wider "Fake News" detection problem. "Fake News", which are defined as "made-up stories written with the intention to deceive" by New York Times, are considered one of the main means of propaganda today and a direct threat to democracy. The rise of blogs, websites and social media platforms has generated a huge amount of textual information, making it almost impossible for an individual to verify the validity of all those sources. During the last years, there have been extensive efforts and research on how Artificial Intelligence and Machine Learning could be used to provide automated solutions for this problem. One of the main initiatives is the Fake News Challenge Competition (FNC)[1]. Assesing the truthfulness of a news story is an extremely challenging problem, even for trained experts. Fortunately, this task can be broken down to separate stages. One first helpful step consists of understanding what other sources are saying about the same topic. Therefore, the first part of FNC (e.g. FNC-1), focuses on this exact problem, called stance detection. Stance detection involves estimating the relative perspective (or stance) of two different text pieces on the same topic and can be classified as either *agree*, *disagree*, *discuss* or *unrelated*. For this specific task, the stance between pairs of headlines and relative articles is requested to be estimated. In this paper we use the publicly available FNC-1 dataset footnote[2]. This dataset is split into a training and test set with a ratio of approximately 2:1 (49972 and 25413 pairs of headlines and articles/bodies respectively). Training dataset is extremely unbalanced since 73% of the data correspond to the unrelated stance, 18% to the discuss stance, 7.4% to the agree stance and only 1.6% to the disagree stance. We implement from scratch vector representations of headline and bodies and experiment with feature engineering to test how Linear Regression and Logistic Regression (also implemented from scratch) perform. We then propose three models (e.g. Random Forest, XGBoost and Multilayer Perceptron), all of which perform better than the official baseline [2].

2. APPROACH AND RESULT

2.1 Train - Validation Split

At first, the training set was split to a training and a validation set with ratio 9:1. Before splitting the training set, the four stances were separated and each subset was shuffled to further increase randomness. Care was taken to split training set in a stratified manner, so the distribution of the four stances is similar in both subsets produced. Stratified sampling is of decisive importance, especially in cases of such extreme class imbalance, and it provides a better scheme than random sampling in terms of both bias and variance. The statistics of each stance at the new sets are presented in Table 1.

Table 1: Training and Validation Sets Stance Distribution

	Training Set	Validation Set	ratio
unrelated	32890	3655	8.99
discuss	8018	891	8.99
agree	3310	368	8.99
disagree	756	84	9.00

2.2 Vector Representations

Before the vector representation part both headlines and bodies were processed. All text was converted to lowercase, only alphabetical characters were kept, extraneous whitespaces were stripped, stopwords were removed and the remaining words were lemmatized using the WordNet Lemmatizer. Finally each piece of text was transformed to a list of tokens. After preprocessing, vector representations of headlines and bodies were extracted. We used three different representations, namely Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF) and Word2Vec. For all cases, a vocubulary of 300 words was created, resulting to 300-dimensional vectors. It worths mentioning that the choice of the vocabulary size was based on related projects, however further experimentation on its size could lead to interesting insights. For the construction of the vocabulary, headline and bodies were concatenated and then preprocessed. Below, a brief description of each representation is given:

- **Bag of Words:** Each document is represented as a vector where the elements correspond to word occurence / counts. Although this model is considered over-simplistic, since it doesn't take into account word order and is purely statistical, it has been proven to work suprisingly well in a plethora of natural language processing applications.

- **Term Frequency - Inverse Document Frequency (TF-IDF):** This model tries to take into account the

[1]http://www.fakenewschallenge.org/#page-top
[2]https://github.com/FakeNewsChallenge/fnc-1-baseline

relevant frequency of each word, rather than only the counts of words in a document. The basic intuition behind, is that as a word keeps appearing in more and more documents the less meaningful or informative this word will essentially be. Therefore, the counts of a term t in a document d, also known as term frequency, is multiplied with inverse document frequency.

Inverse document frequency takes into account the total number of documents n_d and the number of documents that contain term t and it has many variants. In this project we used the following calculation to comply with Python's scikit-learn's library implementation and compare results for proof of concept:

$$idf_t = log\frac{1+n_d}{1+df_t} + 1 \tag{1}$$

Finally, the resulting vectors are normalized by the Euclidean norm.

- **Word2Vec:** It was published by Google back in 2013 and is a neural network implementation that learns distributed representations of words, also known as word embeddings. Provided that there is a large amount of training data (e.g in the order of hundreds of thousands and more), these representations can capture word meanings, semantic relationships and different contexts they are used in. Since each document is represented by multiple words, one challenge lies at the way of combining the independent vectors. To the best of our knowledge, two of the most common ways include averaging and multiplication. For the purposes of this project we used the plain averaging. Due to the large size of training data and article size, a different approach was followed for this type of vector representation. Based on the intuition that a human writer would use strong indicators of agreement or disagreement in the initial or last few sentences, the bodies were clipped and only the first and last 5 sentences were included.

After the extraction of vector representations, for both headline and bodies, the cosine similarity was used to compare the pairwise vectors. Cosine similarity returns the angle between two vectors and is calculated as showed in Equation 3. The main advantage of cosine similarity is that two pieces of text with similar senses will have a high similarity (e.g. small distance) regardless of the absolute frequencies of words in the text.

$$sim(\boldsymbol{u}, \boldsymbol{\nu}) = \frac{\boldsymbol{u} * \boldsymbol{\nu}}{|\boldsymbol{u}| * |\boldsymbol{\nu}|} = \frac{\sum_{i=1}^{n} u_i \nu_i}{\sqrt{\sum_{i=1}^{n} u_i^2} * \sqrt{\sum_{i=1}^{n} \nu_i^2}} \tag{2}$$

Figure 1 illustrates the results. It can be seen that for the BoW and TF-IDF models, the cosine similarity provides clear distinction between unrelated and related (i.e. agree, disagree, discuss) stances, however there in no clear distinction between the related classes. This indicates that cosine similarity on its own could effectively help the classification between unrelated and related classes, but possibly struggle for the separation of the related classes between each other. In addition, the cosine similarity for Word2Vec model doesn't seem to provide any kind of separability between the four stances. This most probably indicates that clipping the bodies doesn't perform well in this context. However, it was impossible to avoid, due to time constraints.

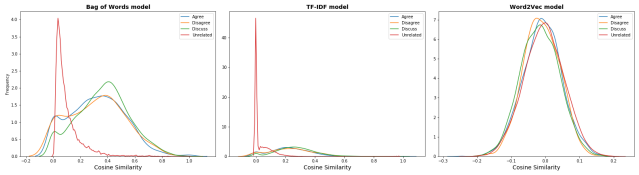

Figure 1: Cosine Similarity between Headline and Body for BoW, TF-IDF and Word2Vec models

2.3 Language Model

Language model representations were extracted for headlines and bodies,using the query-likelihood and the document likelihood models respectively. Laplace smoothing was used in both cases. In the document likelihood model for the bodies, in the case of related stances (e.g. agree, disagree, discuss), a subset of the body's total words was added to the headline to compensate for the imbalance between body and headline length. Then the KL divergence between those models was computed. Results are presented in Figure 2. KL divergence for unrelated stance seems to be slightly dissimilar to other distributions, taking some larger values and having a wider spread.

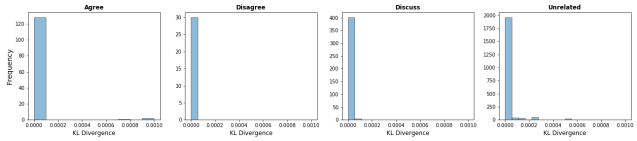

Figure 2: KL Divergence Distribution for all stances

2.4 Feature Engineering

Different distances and global features (e.g. features directky extracted by headline and body rather than their vector representations) that could potentially increase the performance of the stance classification were implemented. As far as distances are concerned we experimented with *euclidean*, *manhattan* and *canberra* distance. Furthermore, the following global features were considered:

- *Overlapping n-grams ratio:* Computes the common n-grams between the headline and body and divides by the total number of unique n-grams in the pair. It is calculated as:

$$overlapping\_ratio = \frac{H \cap B}{H \cup B} \tag{3}$$

where H is the set of words in headline and B is the set of words in the body.

- *Word co-occurrence:* Counts the total number of occurrences of any word of the headline in the body.

- *Refuting words count:* Counts the number of refuting words in both the headline and the body. Refuting words were taken from the FNC-1 baseline without any change and include words like "fake", "fraud", "hoax", "deny", "doubts" etc. The presence of such words in either headline or body can potentially provide an indication for the relationship between headline and body.

- *Discuss words count:* Counts the number of discuss words in headline and body. Discuss words were taken from [3] and include words like "according", "reports", "claims", "investigating", "allegedly" etc. The presence of such words may also indicate a relationship between headline and body.

- *Polarity and Subjectivity:* It seems reasonable that an article which agrees or disagrees with a headline tends to use strong language and sentiment to express its position. In the same sense those type of articles have intrinsically higher levels of subjectivity compared to the discuss and unrelated stances. We used Python's TextBlob package [3] provides a simple API to perform common NLP tasks, including methods for polarity and subjectivity extraction from text.

2.5 Feature Exploration

All the implemented distances were combined with the three different vector representation models and their distributions with respect to the four stances were explored. The same happened for the global features. The exploration results indicate that euclidean distance for the tf-idf model, co-occurence, unigram and bigram overlap ratios are the features that provide the best separability between the distributions of the four stances. All of them seem to provide a clear separation between related and unrelated stances, while the unigram and bigram overlapping ratios indicate a substantially different distribution for the disagree stance. Euclidean distance for the tf-idf vectors also indicates a different distribution for the agree stance with respect to disagree and discuss. Results are displayed in Figure 3.

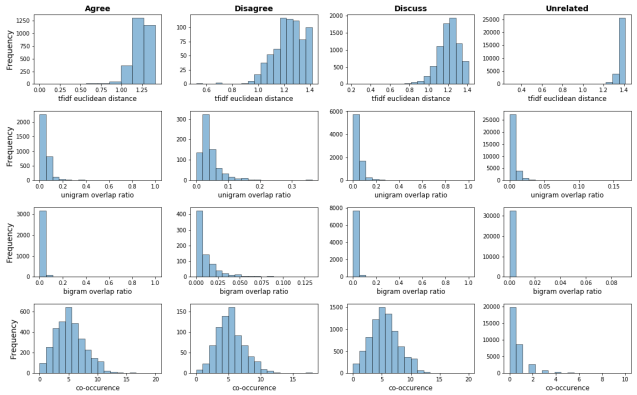

Figure 3: Euclidean distance, N-gram Overlap and Co-occurence Distributions for all stances

[3]https://github.com/sloria/TextBlob

2.6 Linear and Logistic Regression

In this section a subset of the aforementioned features was selected and was given as input to a Linear Regression and a Logistic Regression model. More specifically, we used BoW and TF-IDF cosine similarity, TF-IDF euclidean distance, unigram and bigram overlap ratio, co-occurence and count of refuting and discuss words in the body. Each model is explained below.

Linear Regression

Linear Regression assumes a linear relationship between the features and the target variable and uses a hypothesis function of the form:

$$h_w(x) = \mathbf{w}^T \mathbf{x} \tag{4}$$

It tries to minimize a sum of squared error loss function of the form:

$$L(\mathbf{w}) = \sum_{i}^{N} [y^i - h_w(x^i)]^2 \tag{5}$$

The solution of the Linear Regression problem lies to the determination of the optimal vector $\mathbf{w}$ that minimizes the loss function L. This happens using gradient descent. Gradient descent approximates the global minimum of the loss function by performing multiple iterations using the following update rule:

$$\boldsymbol{w}_{j+1} = \boldsymbol{w}_j - a \frac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w}) \tag{6}$$

The parameter a is called *learning rate* and is of crucial importance for the successful minimization of the loss function. If a is too small, then convergence might be too slow, whereas if a is too large, gradient descent can overshoot the global minimum (e.g. it may fail to converge or even diverge). Therefore, tuning learning rate is essential. Note that the sum of squared error loss function is a convex function, which means that it has a global minimum so there is not the case of convergence to a local minimum.
Linear Regression's output however produces values in a continuous range. In the stance detection problem though, we need to predict a category or class. In other words, we need an output that is bounded between 0 and 1 and corresponds to the probability of each class. Therefore, Linear Regression is inappropriate for such a task.

Logitic Regression

Logistic Regression overcomes the problem of bounded output in the range of $[0, 1]$ by mapping the Linear Regression's hypothesis function using the sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

As a result the new hypothesis function is:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^T \boldsymbol{x}}} \tag{8}$$

Due to the fact that $h_{\boldsymbol{w}}(\boldsymbol{x})$ is heavily non-linear, a sum of squared error loss function cannot be used because it won't be convex. As a result, the cross-entropy loss function is used:

$$L(\boldsymbol{w}) = \sum_{i=1}^{N} [y^i log(h_{\boldsymbol{w}}(x^i)) - (1 - y^i) log(1 - h_{\boldsymbol{w}}(x^i))] \quad (9)$$

Logistic Regression is used in principle for binary classification. In the stance detection problem, however, we have to deal with a multi-class classification problem. In order to handle multiple classes an *"One-vs-Rest (OVR)"* model was implemented. OVR transforms the multi-class classification problem into multiple independent binary classification problems. A separate model is trained for each class, predicting whether a sample belongs to this class or not.

2.7 Model Evaluation

For the evaluation of our models, three different metrics were used.

- **Metric 1 (FNC-1 Accuracy)**: The official evaluation metric of FNC-1 which is a two-level system. The first level contributes 0.25 points per pair to the overall evaluation metric and checks if classification was correct in terms of related versus unrelated stances. Then for the related pairs, additional 0.75 points are added if classification is correct between agree, disagree and discuss stances. The logic behind is that it is much easier to separate related versus unrelated stances than separating the related stances between each other.
- **Metric 2 (F1-score)**: We used F1-score per stance and total weighted F1-score to compensate for the class imbalance. F1-score is a much more robust measure than plain accuracy, especially in cases of such severe class imbalance, since it takes into account both precision and recall.
- **Metric 3 (Confusion Matrix)**: Confusion matrix enables to dig deeper into model performance per stance and inspect true and false positives and negatives per stance. Due to space constraints, the confusion matrices for the various models could not be included in the report.

The validation set was used to tune the learning rate parameter. We tried 100 values in the range $[0.001, 5]$. Results are displayed in Figure 4.The optimal learning rate for this problem was 3.1 (which is generally considered as a large value). Using this value we evaluated the Logistic Regression model in the test set. Results for both validation and test sets are presented in Table 2. We can see that Logistic Regression performs almost similar with the FNC-1 baseline for the validation/hold-out set and is almost 2% better for the test set (the corresponding baseline scores were 79.53% and 75.20% respectively).We need to mention though that the baseline implementation uses a k-Fold cross-validation method for the hold out set. Furthermore, we see that the model performs significantly well for the discuss and unrelated stances with F1-scores 71.7% and 96.22% respectively, but struggles with the less frequent agree and disagree stances. Especially for the disagree stance, the model doesn't make a single prediction for a sample to belong in this stance, so essentially it is like totally ignoring its presence. Further experimentation with features could provide more balanced results per stance. We could try for example using as input the vector representations themselves, or combine them with the engineered features, but then feature exploration and interpretation would be much less intuitive. In question 10, we experiment with more models and algorithms to improve these results.

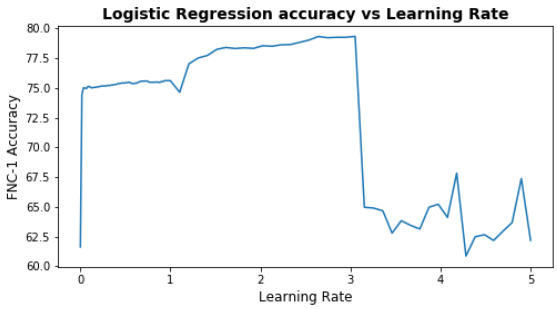

Figure 4: Learning Rate Tuning in Validation Set

Table 2: Logistic Regression Evaluation on Validation and Test Sets

	Validation Set	Test Set
FNC-1 Accuracy	79.34%	76.98%
Weighted F1	83.38%	81.89%
F1-Agree	3.11%	2.20%
F1-Disagree	0%	0%
F1-Discuss	71.70%	70.40%
F1-Unrelated	96.22%	96.06%

2.8 Feature Importance

Using the weights of the separating hyperplanes produced by Logistic Regression we can visualize feature importance per individual classifier in the One-vs-Rest model. Figure 5 illustrates that the euclidean distance is the most important feature for separating all stances except agree. Bag of words cosine similarity and co-occurence also play a significant role for the Discuss vs Rest and Unrelated vs Rest classifiers. Finally unigram overlap is the most important feature for the Agree vs Rest classifier.

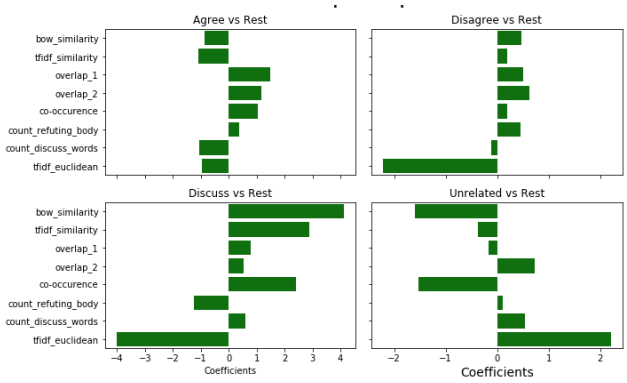

Figure 5: Feature Importance per One-VS-Rest Classifier

2.9 Related Work

During the last years there have been significant efforts in the stance detection problem. For the purposes of this paper we reviewed a large amount of literature and here we present the most interesting findings. Riedel et. al. [4] propose a model of a multi-layer perceptron with one hidden layer, that uses as features the term frequencies of headlines and bodies and the corresponding tf-idf cosine similarities. Despite the seeming simplicity of this model, it managed to rank 3rd out of 80 participants at the FNC-1 competition. Chaudhry et. al. [1] used pre-trained GloVe word embeddings to feed an LSTM-based bidirectional conditional encoding model that achieved more than 97% accuracy on the dev set. Finally, Mrowca et. al. [3] implemented a conditioned bidirectional LSTM using global, co-occurence, polarity and refutation features that resulted into a 9.7% improvement in the FNC-1 evaluation metric.

2.10 Model Improvement

After implementing everything from scatch, we experimented with "off-the-shell" packages to explore more models and try improve the classification performance. More specifically we used a *Random Forest* classifier with 500 estimators and max depth set to 7 for each decision tree, an *XGBoost* classifier with also 500 estimators and a *Multilayer Perceptron* with 3 hidden layer of sizes $(256, 64, 32)$ and sigmoid functions for activation. We experimented with those classifiers in two scenarios. In the first scenario, the original data were used as such. In the second scenario, we tried to overcome the extreme class imbalance with *oversampling*. A common technique to deal with the class imbalance problem is to resample the dataset, either by downsampling the majority classes or by upsampling the minority classes. The disadvantage of downsampling is that there is the danger of information loss, whereas for upsampling the replication of the exact same samples makes the dataset prone to overfitting. The choice was to use the SMOTE (Synthetic Minority Over-sampling Technique) algorithm [2], which manages to overcome the overfitting problem by creating "synthetic" examples along the line segments joining the k minority class nearest neighbors, rather than creating actual copies of the data. The upsampling was proportional to the original stance distribution, because otherwise the performance was downgrated. Therefore agree stance was upsampled with 1.50 ratio, disagree stance with a ratio of 3 and discuss class with a ration of 1.25. A key point though to remember is that oversampling must be applied only to the training set. Otherwise, there is potential bleeding of information from the test (or validation) set to the training set and therefore the classification results are misleading, since the model may predict on the exact same samples it was trained on. This can happen even in the case of SMOTE, where no duplicate observations are created, if the nearest neighbors of minority class observations in the training set end up in the validation set. Results are presented in Tables 3 and 4.

As can be seen from Table 3, trying the new models in the original data managed to improve significantly the results for the agree stance, however classifiers still can't detect the disagree class with the exception of XGBoost model. Table 4 indicates that oversampling managed to achieve an impressive performance improvement. All models are well balanced for every stance and outperform the baseline model in both the validation and test set by an average of 1.8% and 0.8% respectively. The best performing model is XGBoost, with an F1-score of 88.14% in validation set and 83.83% in test set.

Table 3: Random Forest, XGBoost and MLP Evaluation - No Oversampling

	RF		XGB		MLP	
	Val.	Test	Val.	Test	Val.	Test
Accuracy	81.37%	77.07%	82.06%	76.92%	79.32%	75.5%
Weighted F1	86.71%	83.6%	87.7%	83.91%	85.57%	83.27%
Agr. F1	26.56%	19.07%	34.94%	24.01%	24%	21.68%
Dis. F1	0%	0%	6.9%	0.84%	0%	0%
Disc. F1	75.21%	69.66%	75.62%	68.81%	72.92%	68.14%
Unrel. F1	97.57%	96.86%	97.81%	96.96%	96.82%	96.5%

Table 4: Random Forest, XGBoost and MLP Evaluation - With Oversampling

	RF		XGB		MLP	
	Val.	Test	Val.	Test	Val.	Test
Accuracy	81.98%	76.53%	81.83%	75.85%	80.23%	75.64%
Weighted F1	87.36%	83.53%	88.14%	83.83%	86.98%	83.84%
Agr. F1	31.4%	22.24%	37.74%	26.17%	32.26%	28.13%
Dis. F1	14.14%	0.56%	25.45%	9.48%	20.13%	10.09%
Disc. F1	75.29%	67.71%	74.88%	66.09%	72.84%	65.4%
Unrel. F1	97.61%	96.88%	97.89%	96.95%	97.47%	96.9%

3. CONCLUSION

In this paper we dealt with the stance detection problem using the data from the "Fake News Challenge" competition (FNC-1). Vector representations of text like Bag of Words and TF-IDF were implemented from scratch and there was extensive feature engineering to create strong predictors for classification. Linear and Logistic Regression were also implemented from scratch and their performance was tested in the validation and test sets. Logistic Regression performed well for the majority classes (e.g. unrelated and discuss), but failed to classify correctly samples of the minority classes (agree and disagree). Finally, we proposed an XGBoost model that uses an oversampled version of the original data that improves the official FNC-1 baseline evaluation metric by 2.33% and 0.65% in the validation and test set respectively.

References

[1] Ali K Chaudhry, Darren Baker, and Philipp Thun-Hohenstein. Stance detection for the fake news challenge: Identifying textual relationships with deep neural nets.

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[3] Damian Mrowca, Elias Wang, and Atli Kosson. Stance detection for fake news identification.

[4] Benjamin Riedel, Isabelle Augenstein, Georgios P Spithourakis, and Sebastian Riedel. A simple but tough-to-beat baseline for the fake news challenge stance detection task. *arXiv preprint arXiv:1707.03264*, 2017.