FAKE NEWS CHALLENGE STAGE 1 (FNC-I): STANCE DETECTION

Andreas Louka
Department of Computer Science
MSc Data Analytics
COM6513 Natural Language Processing
alouca1@sheffield.ac.uk

Abstract

Fake news is an emerging threat to our society in the 21st century, mainly because fake news spread easily across social network platforms. The Fake News Challenge (FNC1) encourages the development of a system to perform 'stance detection' (Stance Detection' is the relation between the headline of the article with the main body). The system classifies the headline as one of the four options: 'agrees', 'disagrees', 'discusses' or its 'unrelated' to the article. Natural Language Processing (NLP) concepts have been applied to create a stance detection model. The final scoring result obtained is 77.31% with the Decision Tree Classifier.

1 Introduction

Living in the 21st century, there are many ways which new media can be made available to people, including newspapers and social media. As easy it is for media to spread around the world, so it is for fake media as well. These false stories can create confusion and spread the wrong message to the people reading them. This creates a serious challenge to the news industry, and as technology (and particularly Artificial Intelligence) advances, it is time for us to start using our resources to resolve this problem.

The aim of the Fake News Challenge [1] is to use Artificial Intelligence (AI) to automate the process of identifying news that are fake. As explained in the challenge, the first step one can follow is to identify what other resources and news are saying about the same topic [1]. By using this method, stance detection emerges as a solution, where based on several titles and articles of the same topic one can classify how relevant the main body is to the headline with four stances, which are: "agree", "disagree", "discuss" and "unrelated". This approach does not necessarily define if an article is fake or not, but it informs the reader the stance the article takes relevant to the title. If for example the stance is detected to be "disagree", then the reader knows that the article is at least related to the headline, but expresses the opposite opinion than the headline.

Moving on to the following sections of this report, the stance detection task will firstly be explained along with an overview of the data set used. Further to that, baseline and improved system descriptions will be analysed along with the methods used for the creation of both systems. Learning curves and results on both development and testing sets will be provided, to allow for a detailed and accurate conclusion analysis. Finally, potential improvements will be discussed.

The code is available from the Github page: https://github.com/AndreasLouka/fakenews

2 Stance Detection Task & Dataset Overview

The aim of the stance detection task is to identify the perspective of the main body text to the headline of the article. The relationship is either related or unrelated, and if related to what extent. As mentioned in Introduction, the four stances are *Unrelated, Discuss, Agree, Disagree*. See figure 1 for visualisation and better understanding.

The dataset provided by FNC1 can be downloaded from their Github page [2]. (Development data were also provided in order to represent the test data which would have been released towards the end of the submission; can be found in their Github page [3]). The dataset contains 1648 unique headlines and 1669 body texts, which

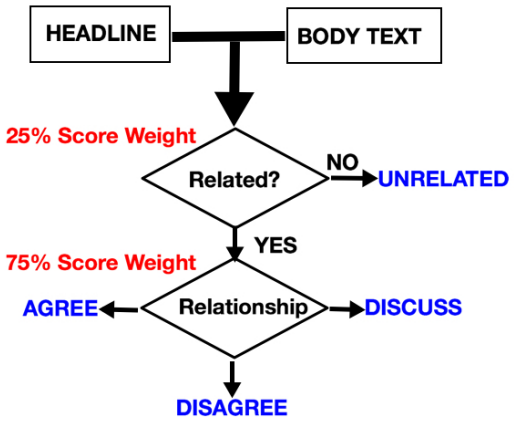

Figure 1: Stance Explanation

have been paired together to create about 50,000 (49,972) headline-body pairs. Each article-body pair contains a label, which is the stance. It should be noted that labels provided for each pair are unbalanced, since there are about 18% Discuss, 73% Unrelated, 7% Agrees and 2% Disagrees.

A scoring metric for stance detection is also provided from the Github page [2]. As shown in figure 1, the most important part is defining the relationship (agree, disagree or discuss), which is given a weighting score of 75%. Therefore, determining whether is related or not is only given weighting a score of 25%.

3 Baseline System

The baseline system was based on term frequency-inverse document frequency (tfidf), which is a statistical approach that reflects the importance of a word or n-gram (contiguous sequence of n items from a sequence [5]) in a corpus.

3.1 Bag of Words

Bag of words of words is a a fixed-length feature representation of text without grammar or word order which is performed during pre-processing of the dataset. The text was tokenised using nltk.word_tokenize [6]. Removal of stop-words was then applied with nltk.corpus stopwords [7]. The output was a list of lists containing the pre-processed dataset. A list of the labels (stances) was also extracted that will be used in the classification procedure (section 3.3).

3.2 TFIDF

Collection of raw documents was converted to a matrix of tfidf features, by importing TfidfVectorizer from sklearn [8]. The pre-processed data (bag of words) were fed to TfidfVectorizer with the n-gram parameter adjusted to include n-grams of range 1-3. Resulting output is a matrix with occurrence counts for all the features (n-grams).

3.3 Classification

Classification was performed by importing four different classifiers from sklearn (Decision Trees, Gaussian Naive Bayes, Support Vector Machine (SVM) and Logistic Regression) [9]. Classification is the procedure of identifying in which category a new observation belongs to, based on the observations of a training set [10]. Matrix output from TfidfVectorizer along with the list of labels was applied to the classifiers for training. Predictions were then made from the classifier using the development and testing datasets. The predictions were finally applied to the scoring metric provided by FNC1, along with the correct labels.

As the highest score achieved from the baseline system (using Logistic Regression) was 50.20%, more features needed to be extracted and therefore an improved system was developed which is explained in the section that follows (all results are shown in section 5).

4 Improved System

4.1 Word Overlap

The first improvement was to generate world overlap features, which are features present both in the headline and in the article body. Two operations were used to perform this step which are *intesection & union* [11]. *Intersection* returns a set with elements common in the two parameters applied, and union returns a set with all elements in the two parameters. The preprocessed data (bag of words) of article bodies and headlines were used as the parameters for these operations, and the output was provided by dividing the intersection result with the union result.

4.2 Binary co occurence

Similar to word overlap, binary co occurrence finds tokens which are present both in the headline and the body text, but returns a number of counts as a feature. The counts were saved as a vector which was later on combined with all the other features extracted.

4.3 Doc2Vec

Doc2Vec is an unsupervised algorithm which learns continuous distributed vector representations from text [12]. The most common fixed-length feature representation is Bag of Words as previously explained in section 3. Bag of words has two main weaknesses which are resolved with Doc2Vec. Bag of words ignores the order the words appear in text and secondly they ignore semantics such as "very" and "strong". Some ordering of the words is kept when using n-gram features in the tfidf step, but still this ordering is for either 1,2 or 3 words depending on the n-gram size. Doc2Vec overcomes these weaknesses since every paragraph is represented by a column matrix, were every word in the paragraph is mapped to a vector which is represented by another matrix.

The Doc2Vec model was instantiated with a vector of size 50 and a train vocabulary was created through 150 iterations. The minimum word count was set to 2, for the purpose of allowing higher frequency words to have more weighting. The train vocabulary was saved and could be loaded later on for creating the Doc2Vec array representations of the datasets. By saving the model it greatly reduced the execution time of the system, since training the Doc2Vec model took about two hours.

4.4 Cosine Similarity

Cosine similarity measures the cosine angle between two vectors, therefore determining how similar the two vectors are with a score from 0 to 1 [13]. This feature extraction was performed by extracting headlines and body articles from the datasets and transforming them to matrices using tfidf in a similar way explained in section 3.2. The cosine angle between them was calculated by multiplication of the matrices with their inverse, and the output result was converted to an array to be later on concatenated with the rest of the features.

5 Results & Discussion

All features extracted as explained in the previous sections were concatenated together to be passed to the classifiers for training.

5.1 Baseline System Results

Baseline system scoring results obtained from the scoring metric provided can be seen in table 1. As shown below, the highest score (50.20%) obtained was by using the development dataset with Logistic Regression.

Classifier	Dev Data Score (%)	Test Data Score (%)
Decision Trees	43.08	43.45
SVM	40.70	39.10
Gaussian Naive Bayes	42.58	40.35
Logistic Regression	50.20	47.82

Table 1: Baseline system scoring results for all Classification Algorithms implemented.

5.2 Improved System Results

Improved system scoring results obtained from the scoring metric provided can be seen in table 2.

Classifier	Dev Data Score (%)	Test Data Score (%)
Decision Trees	73.66	77.31
SVM	70.28	72.47
Gaussian Naive Bayes	47.45	44.01
Logistic Regression	71.44	69.61

Table 2: Improved system scoring results for all Classification Algorithms implemented.

For a more accurate representation of the results, the confusion matrices for two of the classifiers (using the test dataset) are shown in the tables below (table 3 & 4).

	agree	disagree	discuss	unrelated
agree	0	0	259	65
disagree	0	0	76	13
discuss	0	0	852	145
unrelated	0	0	204	3417

Table 3: Confusion matrix for Decision Trees with the test dataset

	agree	disagree	discuss	unrelated
agree	102	66	144	12
disagree	3	48	38	0
discuss	85	105	791	16
unrelated	332	296	751	2242

Table 4: Confusion matrix for Logistic Regression with the test dataset

As it can be seen from tables 2, 3 & 4, even though Decision Tree provided a higher overall score, the correct classifications obtained were only for the discuss and *unrelated* stances. On the other hand, the Logistic Regression classifier has made classifications in all the stances. Decision Tree fails to predict correctly *agree* and *dis-*

agree stances; which might be due to the unbalanced dataset provided, as explained in section 2.

5.3 Learning Curves

Learning curves have been plotted for the highest scoring classifier of the baseline (Logistic Regression) and improved systems (Decision Tree). Learning curves allow the performance comparison of the model to be seen as the number of training points increase [14]. Cross validation was used to create the learning curves, which is a technique used to asses how well a model generalises. Y-axis indicates the score, which is the opposite of error. Therefore, the higher the score, the lower the error. X-axis represent the number of training examples.

Note: red lines indicate training score, and green lines indicate cross validation score.

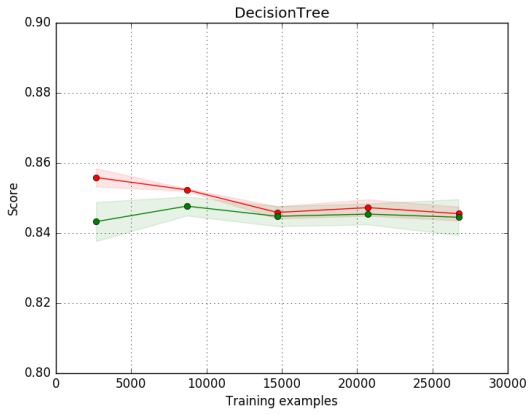

Figure 2: Decision Tree learning curve.

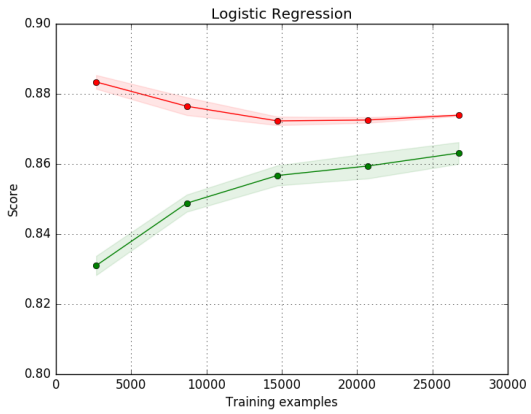

Figure 3: Logistic Regression learning curve.

As shown from the learning curves, the Logistic Regression classifier has higher score on the training line than Decision Trees, which might indicate a better performance with less training data. On the other hand, the score for the cross validation line is higher on lower training data for the Decision Trees. Overall, Logistic Regression converges at a higher score, but the two lines of Decision tree are very close together. It should be noted that the ideal learning curve is the one that generalises to new data, the two lines have a small gap between them and also the lines converge at similar values. Therefore, the Decision Tree performs better overall on the particular dataset.

6 Conclusion

In this work, Natural Language Processing concepts have been applied for the stance detection task proposed by the fake news challenge. The best performing model included feature extractions of tfidf, world overlap, binary co occurance, Doc2Vec and cosine similarity. The highest score obtained was 77.31%, using the Decision Tree classifier.

As for future work to further improve the performance of the system, the precision of the relationship can be improved as it is given the highest weighting score. A possible solution can be to implement hand-crafted features in combination with all the features already been used. A token for words not in the embedding dictionary could also help improve the precision. Further to that, obtaining additional more balance training data could help the model generalise better. Finally, syntactic features can be further improved by the addition of grammatical dependancies or named entity labels.

7 References

[1] D. Pomerleau and D. Rao (2016). Fake news challenge, Available from: http://www.fakenewschallenge.org/

[2] https://github.com/FakeNewsChallenge/fnc-1

[3] https://github.com/sheffieldnlp/fakenewschallenge-teaching

[4] Wikipedia, The Free Encyclopaedia, tfi-idf, Available from: https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[5] Wikipedia, The Free Encyclopaedia, n-gram, Available from: https://en.wikipedia.org/wiki/N-gram

[6] NLTK 3.2. documentation, ntltk.tokenize package, Available from: http://www.nltk.org/api/nltk.tokenize.html

[7] NLTK, Accessing Text Corpora and Lexical Resources, Available from: http://www.nltk.org/book/ch02.html
[8] scikit-learn, TfidfVectorizer, Available from: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[9] scikit-learn, Classifier comparison, Available from: http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
[10] Wikipedia, The Free Encyclopaedia, Statistical Classification, Available from: https://en.wikipedia.org/wiki/Statistical_classification
[11] The python Standard Library, Data Types, Unordered collections of unique elements, Available from: https://docs.python.org/2/library/sets.html
[12] Quoc Le, Tomas Mikolov, Distributed Representation of Sentences and Documents, Google Inc, 1600 Amphitheatre Parkway, Mountain View, CA 94043
[13] Wikipedia, The Free Encyclopaedia, Cosine similarity, Available from: https://en.wikipedia.org/wiki/Cosine_similarity
[14] Christepger Meek et al.(2002), The Learning-Curve Sampling Method Applied to Mode-Based Clustering, Microsoft Research One Microsoft Way Redmond, WA 98052-6399, USA