Fake News Detection with Machine Learning

Vladyslav Maksyk
University of Stavanger, Norway
v.maksyk@stud.uis.no

ABSTRACT

In this paper, We examined whether Natural Language Processing techniques can be utilized in the task of differentiating legit and fake news. We used a labeled data-set to build a classifiers that can decide whether a claim is fake or legit based on the search result which contains several articles related to the claim. The classification is solely based on the content of those articles. The implemented classification models are compared on their performance which is per class accuracy. The project forms a foundation for further studies in fake news detection.

KEYWORDS

Machine Learning, fake news, text classification

1 INTRODUCTION

In the modern world where internet has taken over the role of the main information resource, it is vital to control the spread of fake news. With the use of the social media platforms like Twitter, Facebook etc. nowadays fake news can be disseminated among millions within a short period of time.

With the excessive amount of information nowadays it is not always easy to determine whether the given information is fake or real. While in the past fake news was mainly accidental, these days it is generally published for the purpose of gaining political or financial benefits. Hence, fake news can be a powerful weapon in creating biased opinions that can affect the election outcomes and benefit a certain candidate. With all this in mind, detection of fake news has gained a huge importance in order to prevent its negative influence on the society.

Fake news detection is generally considered as a binary text classification problem. Distinctions can be performed based on the source of the article or the text of the article itself. In this project we were mainly focused on feature extracting from text and machine learning algorithms to classify news articles to legit or false based entirely on the text itself.

2 METHOD AND DESIGN

In this section we explain the sequence of actions taken, that lead from data processing over feature generation to conduction of the research. All steps performed throughout the experiment are describes here with their requirements and results. Hence, it is that the research can be reproduced based on the description and the attached source code.

Supervised by Vinay Jayarama Setty.

Project in Computer Science (DAT550), IDE, UiS
2019.

2.1 Goal and Challenge

The main goal of this study shall be to compare different machine learning algorithm and feature extraction techniques for the purpose of binary text classification. The result should represent a ranking with the best performing combination of the last two.

2.2 Pre-processing

In this project two datasets from two different sources were used. The datasets were gathered from open-data fact checking platforms like Politifact [2] and Snopes [3].

Dataset	Snopes	PolitiFact
Total claims	4341	3568
True claims	1164	1867
False claims	3177	1701
Unverified claims	-	-
Claim sources	-	95
Articles	29242	29556
Article sources	336	336

Table 1: Data statistics

These datasets have the following important attributes in common:

- **Statement:** Claim that had to be verified.
- **Speaker:** The person who made the statement.
- **Search Results:** Contains search result from the internet related to the statement made.
- **Description:** A short summary based on the search result.
- **Credibility:** Labels the statement as true or false.

Both datasets were provided by Vinay Jayarama Setty, the project supervisor.

These datasets were in a form of multiple JSON files that had to be combined into a single file for convenience. Before moving on to Feature Generation the datasets had to be preprocessed. All texts were first lower-cased and stripped from stop-words and non-words characters. After the preprocessing step each entry of the data set consist of the binary label for fake or true and the text of the article.

2.3 Feature Generation

Many different algorithms were considered for the process of feature generation. However only a few of them, appeared to be suitable for text classification and reasonable for our study.

After some research the *Vector Space model* seemed to be the most simple and promising to start with. It is also known as *bag-of-words* or *bag of n-grams*. The main idea of this algorithm is to use the number of occurrences as a feature to train the classifier.

Hence, this algorithm does not preserve any information about the text, such as structure of the text, word order and grammar.

Another approach was to use *Term Frequency - Inverse Document Frequency*. Tf-idf reflects the importance of a term with the respect to a document in a set of documents. Moreover, term frequency is the number of times a word occurred in a document and inverse document frequency is the inverse function of the number of documents in which it occurred. The values are calculated as follows:

$$tf - idf(t,d) = tf(t,d) \times idf(t) \tag{1}$$

$$idf(t) = \log \frac{1 + n_d}{1 + df(d,t)} + 1 \tag{2}$$

where n_d stands for the total amount of documents and $df(d,t)$ for the amount of documents that contain the term.

As a third *Feature Generation* approach we used embeddings generated with *Doc2Vec* [5] model which is based on *Word2Vec*. The purpose of using it was to create a vector representation of each article rather than separate words, as it is done in *Word2Vec*. We feed *Doc2Vec* with preprocessed and comma-separated list of words to produce 300-length embeddings vector for each article. The main reason for using *Doc2Vec* is to preserve information about word order of the documents. Unlike *Word2Vec*, *Doc2Vec* adds a "document vector" to the output representation, which allows the classifier to learn information about the word sequence and contains information about the document as a whole.

2.4 Choice of Algorithms

In order to cover and evaluate a broad field of algorithm, all algorithms used were categorized into three categories:

- **Support vector machines**
- **Probability based classifiers**
- **Linear models**
- **Ensemble methods**

Several algorithms from each category were researched and the best performing one was chosen then implemented and used for the project. As an example of *Support Vector Machines algorithm* [8], Linear Support Vector Classification was tested and implemented. Generally SVM plots each data item as a point in a high dimensional space where each value of the feature is being a value of a specific coordinate .Then a classification is performed by finding the hyperplane that differentiate the two classes the most.

For the Probability based classifiers *Multinomial Naive Bayes* [7] classifier was used. This method is broadly used in text classification and is based on *Bayes's* theorem of independence assumptions between features. This algorithm works on continuous data from the input vectors, and assumes that the values are distributed with Multinomial distribution.

One of the *Linear* models that was chosen for the research is *Stochastic Gradient Descent Classifier* [1]. Its a simple but very efficient algorithm for discriminative learning. The estimator relies on gradient descent for the implementation of regularized linear models. The model updates along the way based on the estimation of the gradient of the loss.

Finally, to represent one of the *Ensemble methods* we have chosen to implement *Random Forest Classifier*. It is a machine learning

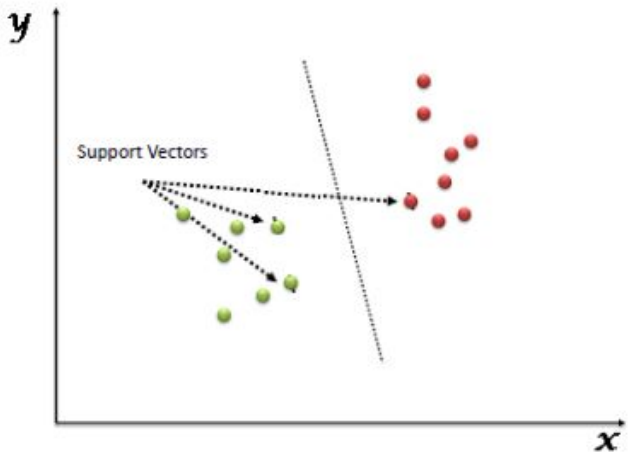

Figure 1: Support Vector Machine

algorithm which builds several decision trees while training the model. It makes predictions from a combination of decisions based on the base model. *Random Forest* solves the over-fitting problem in Decision trees which are caused by huge depth of the tree. *Random Forest* makes use of multiple trees to find the average result.

The implementation of the classifiers is based on *sklearn* [9] library for *Python*

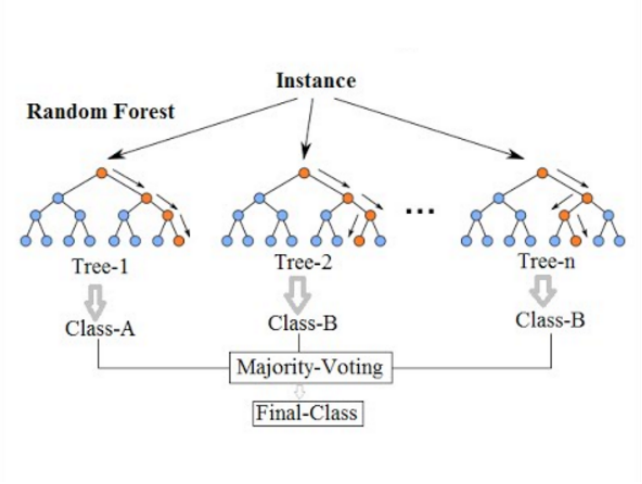

Figure 2: Random Forest Classifier Model (Source: datacamp.com)

3 EVALUATION METRICS

In this section we describe the evaluation metrics used for this study. Different evaluation metrics allowed us to have a more broad perspective on the models performance as well as compare them with each other.

- **Receiver-Operating Characteristics:** ROC curve is a metric to evaluate the classification accuracy under various conditions. True curve is generated by plotting the true positive rate against the false positive rate [4].
  - *True Positive:* Predicted fake news as fake.
  - *True Negative:* Predicted true news as true.
  - *False Positive:* Predicted fake news as true.
  - *False Negative:* Predicted true news as fake.
- **Classification Accuracy:** This shows the amount of correct predictions out of total number of predictions.
- **Precision, Recall and F1 value:**
  - *Precision:* Depicts the relative instance ratio tho the retrieved instances.

$$precision = \frac{TP}{TP + FP} \tag{3}$$

  - *Recall:* Depicts the relevant instances proportion being retrieved from the total amount of relevant instances.

$$recall = \frac{TP}{TP + FN} \tag{4}$$

  - *F1:* Offers additional insight into the performance through precision and recall.

4 RESULT AND ANALYSIS

The final result of the research shows a decent variety between some of the algorithms. The results are shown in table 2. In this section, several results that appear to stand out from the rest, will be highlighted and compared. As well as the result of low precision and recall in several cases will be explained. As shown in table 1 earlier, the amount of false claims in the Snopes dataset significantly outnumbers those that are true. To be more precise, out of 4341 only 26.8% are true, the rest is false. Hence, the precision of True Positives is expected to be low.

Analyzing the result of all models applied on Politifact Dataset, we can state that according to all evaluation metrics, such as per class accuracy, precision and F1 score, it appears that the performance of *Random Forrest* classifier was the best. However, there was no significant difference compared to other models with different configurations. Although it is worth mentioning that *Naive Bayes* classifier with *Bag-Of-Words* feature generation was the worst performing model. This may be due to the fact that *Bag-Of-Words* feature generation does not reflect the importance of a term in respect to the whole corpus. In order to have a visual understanding of the performance results on the PolitiFact dataset they are displayed in *[Figure 3]* .

Furthermore, we observed the following results from training the same set of models on the Snopes dataset. All classifiers, except *Random Forrest*, trained on this dataset seem to struggle with classifying the true labels. At first, we assumed that the reason why it happens could be inequality of true and false labels in the train dataset, but after equaling out the weight of each label it appeared that this does not have a great influence on the trained models.

Based on an equally distributed per-class accuracy we state that *Random Forrest* performed better than other models. But taking into account the *F1 Weighted Average* we state that *Support Vector Machine* with different configurations showed the best performance. Similarly, to the results obtained on PolitiFact dataset, there was no significant difference in performance observed between different models and configurations. The results are visualized in a ROC curve. See figure 4.

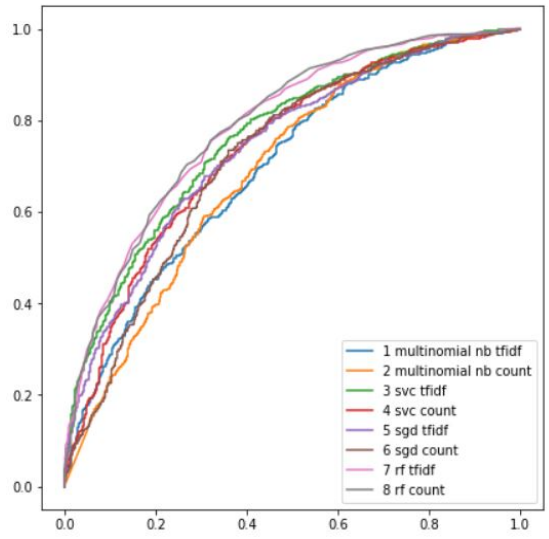

Figure 3: Performance of all algorithms and configurations. PolitiFact dataset

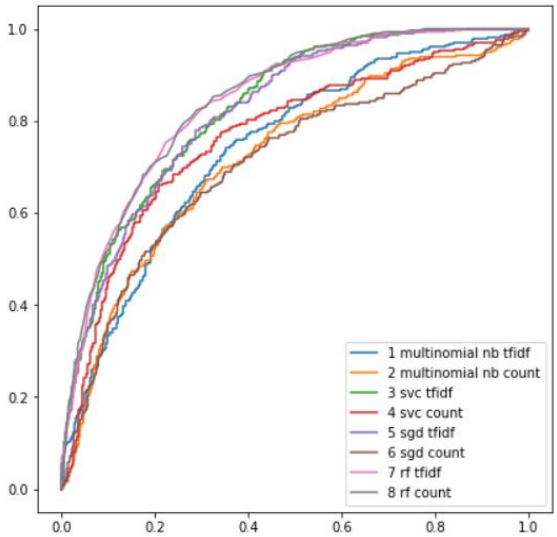

Figure 4: Performance of all algorithms and configurations. Snopes dataset

Rather disappointing was the result of all classifiers in combination with *Doc2Vec* feature generation technique. Compared to other feature generation techniques it showed in general a 10% decrease in per-class accuracy and F1 average score.

Dataset	Configuration	True Claims Accuracy	False Claims Accuracy	F1-Score Average
Politifact	Naive Bayes (count)	0.76	0.71	0.72
	SVC (count)	0.67	0.6	0.64
	SGDC (count)	0.61	0.76	0.57
	Random Forrest (count)	0.71	0.71	0.71
	Naive Bayes (tf-idf)	0.72	0.68	0.7
	SVC (tf-idf)	0.71	0.65	0.68
	SGDC (tf-idf)	0.67	0.67	0.66
	Random Forrest (tf-idf)	0.7	0.66	0.68
	Naive Bayes (Doc2Vec)	0.56	0.59	0.57
	SVC (Doc2Vec)	0.6	0.69	0.62
	SGDC (Doc2Vec)	0.54	0.57	0.54
	Random Forrest (Doc2Vec)	0.57	0.56	0.56
Snopes	Naive Bayes (count)	0.76	0.77	0.71
	SVC (count)	0.55	0.8	0.74
	SGDC (count)	0.74	0.62	0.63
	Random Forrest (count)	0.78	0.78	0.74
	Naive Bayes (tf-idf)	0.66	0.82	0.78
	SVC (tf-idf)	0.5	0.82	0.74
	SGDC (tf-idf)	0.64	0.83	0.78
	Random Forrest (tf-idf)	0.55	0.86	0.77
	Naive Bayes (Doc2Vec)	0.34	0.73	0.63
	SVC (Doc2Vec)	0.43	0.73	0.64
	SGDC (Doc2Vec)	0.36	0.72	0.61
	Random Forrest (Doc2Vec)	0.47	0.75	0.68

Table 2: Comparison of various approaches for credibility classification on Snopes and PolitiFact datasets.

5 CONCLUSION

Main purpose of this research was to prove that fake news detection based on the content can be an example of binary test classification. In the result section it was shown that *tf-idf* in combination with *Random Forrest* or *Support Vector Machine* outperform other configurations of classifiers and feature generation techniques. The best performing models were able to detect about 8 out of 10 fake claims correctly, which confirms that they are well suited for this task. Almost certainly, these algorithms showed a good performance because they were able to better cope with a large amount of features, and were able to weight some of them more than the others.

The study was presented clear and transparent. All intermediate steps were reasoned and described in great details. The code is available on my *Github* [6] page. Following the work flow of this research, the experiment can be reconstructed with minor or none variation in result.

REFERENCES

[1] Leon Bottou. 2018. Large-Scale Machine Learning with Stochastic Gradient Descent. (2018), 2–8.

[2] Fact checking platform. (accessed April 4, 2019). https://www.politifact.com/

[3] Fact checking platform. (accessed April 4, 2019). https://www.snopes.com/

[4] Michael D. Franzen. (accessed April 17, 2019). Receiver-Operating Characteristics. https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-79948-3_1240

[5] Quoc Le and Tomas Mikolov. 2014. Doc2Vec: Distributed Representations of Sentences and Documents. https://cs.stanford.edu/~quocle/paragraph_vector.pdf

[6] Vladyslav Maksyk. 2019. Fake News Detection With Machine Learning. https://github.com/vladmaksyk/Fake-News-Detection

[7] *John Michael Kovachi.* (accessed April 18, 2019). Implementing a Multinomial Naive Bayes Classifier. https://medium.com/@johnm.kovachi/implementing-a-multinomial-naive-bayes-classifier-from-scratch-with-python-e70de6a3b92e

[8] *Rohith Gandhi.* (accessed April 19, 2019). Support Vector Machine. Introduction to Machine Learning. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[9] *scikit-learn: machine learning in Python scikit-learn 0.20.3 documentation.* (accessed April 18, 2019). https://scikit-learn.org/stable/