3244-2010-0029 - Project Final Report

Abstract

"Fake news" are fabricated news stories intentionally created to misinform or deceive readers. Detecting fake news essentially boils down to the words and linguistic patterns used. This, together with the current climate of hoaxes frequently found online, has inspired us to work on this project, which includes finding common words used in fake and real news. We also want to identify a Machine Learning approach that performs best in detecting fake news, and conclude whether different text vectorization techniques can help improve our models' performances. Some of our project highlights include investigating the impact of having truncated texts on model performance and the use of F-beta score as a metric to evaluate the performance of our models in our project's context.

Introduction

Motivation

Fake news or disinformation is deliberately made to spread propaganda, and such malicious intent may have a far greater impact on societies as a whole, politically, socially, economically, and so forth, making this issue highly detrimental. Furthermore, with the growth in popularity of social media and in an era where most people get their news off the internet, fake news has become increasingly widespread, making manual fact-checking impractical thus giving rise to the importance of having an efficient machine learning approach to filter out fake news from the vast sea of information.

Problem Statement and Research Questions

Using supervised machine learning coupled with Natural Language Processing (NLP), we aim to identify a classification model that is able to detect fake news best by scoring the models tested using F1 and F-beta scores as our metrics. In addition, some of the research questions that we have based our project on include:

1. What are the most common words found in fake and real news respectively? Are they distinctive?
2. How would different word embedding techniques, specifically the use of frequency-based versus prediction-based word vectorizers, affect our model performances?
3. As content is typically summarised in the first few sentences of articles, would we still yield similar model performances with truncated texts as compared to full texts?

Related Work

Fake news detection is not a new field of research in machine learning. In fact there have been a multitude of research papers that aim to discover the best machine learning models when it comes to fake news detection, including reports from various universities and past CS3244 projects from previous semesters. This further substantiates the importance of having an accurate and efficient automatic way of identifying fake news.

The abundance of past works in this area gave us an understanding of what has been experimented with and we noticed that past projects focused on using traditional machine learning models and some deep learning models. In this project, we shall experiment with other Machine Learning models, including AdaBoost, Random Forests and Extra Trees. Since the structure of news articles is such that the first few sentences summarises the essence of the content, we shall also experiment with truncated texts and investigate whether this would make huge differences to our results.

Other reports we take reference from that covers Fake News Detection more extensively include methods involving deep learning models[1] that can be used as multi-modal feature extractor and an event discriminator which is able to identify event-invariant features of a given news. However, we will not be covering Deep Learning in this report.

Dataset

We make use of the Information Security and Object Technology (ISOT) research lab Fake News Detection Datasets from the University of Victoria[2]. It is obtained from Reuters, a legitimate news site and sites tabbed as unreliable by Politifact.com, a fact-checking organisation in the United States. The datasets consist of a 'Fake' CSV file containing 22,845 fake news articles and a 'True' CSV file containing 21,416 true articles. Both contain feature columns that include title, text, subject and date. As the datasets came from a reputable source, is suitably large for our project and is relatively clean, we deem it reliable and suitable to use for this short-term project.

With this data, we experimented with several traditional Machine Learning models, on which we conducted hyperparameter turning, in order to identify best performing models according to their efficiency and F1 scores, which takes special consideration of the false positive and false negative rates of each model.

Methods

Data Cleaning and Preprocessing

We acknowledge that the length and content of titles and texts of news articles vary, with titles having to attract potential readers' attention while capturing the main essence of the text. Texts on the other hand tend to summarise the content of the article in the first few sentences, followed by elaborations. Thus, we decided to evaluate our model performances on 'title' and 'text' separately.

We removed data instances where the 'text' feature was empty and used the 'NLTK' package in Python to remove stop words and non-alphanumeric characters, thereby removing additional noise. In addition, all words were lemmatized and converted to the lowercase format for standardisation.

Text-Vectorisation

The titles and texts of news articles need to be converted into numerical representations so that they can be used to train machine learning models. This is typically done through text vectorisation, where texts are converted to numerical vectors.

In this project, we will make use of two vectorizers, namely TF-IDF and Word2vec, where the vectorizer will first be initiated. The text will then be fitted to the vectorizer, and then transformed for an actual vector to be created.

TF-IDF, a frequency-based vectorizer, reflects the relevance of a word for a document in a collection of documents. The higher the metric, the more relevant the term is to a particular document, which means that the term is less common across all the documents. In this way, documents with similar words will have similar vectors, which is what we hope will help us differentiate true and fake news articles. To compute tf-idf[3], we will need two terms. Firstly, the term frequency, that is the number of times (tf_{ij}) the word j appears in the document i.

$$tf_{ij} = \#(W_j \mid W_j \epsilon D_i)$$

Secondly, we have idf_j, which is the log of the total number of documents over the number of documents that contains the word j.

$$idf_j = log(\frac{\#D}{\#(D_i \mid W_j \epsilon D_i})$$

The value $tfidf$ is then calculated by

$$tf - idf_{ij} = tf_{ij} * idf_j$$

Word2vec on the other hand is a prediction-based word embedding technique, which involves Neural Networks (NN). This means being Connectionists, and incorporating back-propagation in our model. Unlike TF-IDF, it is able to capture semantic and syntactic information of words. Word2vec utilizes either of 2 model architectures to map words into numerical vectors: Continuous Bag of Words (CBOW) or Skip-Gram. In this project, we chose to utilize the Skip-Gram model, which predicts embeddings for the surrounding context words in the specific window, given a current word. The input layer contains the current word and the output layer contains the context words. The hidden layer contains the number of dimensions in which we want to represent the current word present at the input layer. The hidden layer is computed as $\boldsymbol{h} = \boldsymbol{W^T x}$, where $\boldsymbol{W^{V \times N}}$ is the weight matrix we want to optimise over. The output layer values are computed as $\boldsymbol{Y} = \boldsymbol{W'^T h}$. For the Skip-Gram model, the loss can be computed as

$$E = -\sum_{c=1}^{C} \blacksquare u_{j_c^*} + C \cdot \sum_{j'=1}^{V} \blacksquare exp(u_{j'}) \blacksquare,$$

where j_c^* is the index of the cth output context word and $u_{j_c^*}$ is the score of the j-th word in the vocabulary for the c-th context word.

Models & Optimisation

In our project, we will be mainly focusing on traditional machine learning models. There are 4 main categories of models that we will be focusing on, namely linear models, non-linear models, tree-based algorithms and some ensemble methods.

Linear models	Naive Bayes, Linear Support Vector Classifier, and Logistic Regression
Non-Linear models	k-Nearest Neighbours
Tree-based models	Decision Tree
Ensemble methods	Random Forest, Extra Trees, and Adaboost

We will use the pre-processed data and run it on these 8 different models, before evaluating and comparing their performance using suitable metrics. For models that require hyperparameter tuning, we used GridSearchCV to perform optimisation.

Evaluation

Evaluation Metric

In order to evaluate our models, we have used multiple evaluation metrics. The metrics are mainly, recall, precision and F1 score. The formula below shows us how precision is calculated.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

However, we also need to know how recall is being calculated - which the formula below shows how it is being calculated.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Then, we combine both precision and recall to form our F1 score metrics.

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$False\ Positive\ Rate = \frac{False\ Positive}{False\ Positive + True\ Negative}$$

As we want to avoid detecting fake news as true since we wish to prevent fake news from spreading, we have considered False Positive Rate (FPR) as well to further understand how our model is doing in terms of misclassifying fake news as true.

From these metrics, we will be able to form confusion matrices. Taking into account what our focus is, we have decided to use F1 scores to compare our models' performances. We concluded that false negatives and false positives are more crucial than their true counterparts in the context of fake news detection, where the rates of predicting fake news as true and predicting true news as false should be low.

Preliminary Results

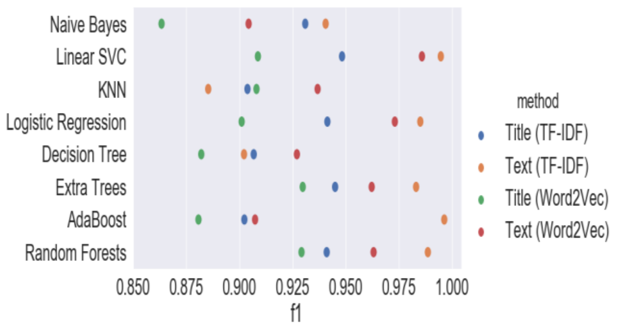

Figure 1: A comparison of F1-scores between different models

As seen from the results presented above, our models achieved high F1-scores of over 0.8 running on different vectoriser techniques and on both text and titles.

We observed several insights from the results in Figure 1:

1. Running models on news text performed better than titles (Orange dots score higher and are more towards the right than blue dots for TF-IDF vectorization, Red dots score higher and are more towards the right than green dots for Word2Vec vectorization)
2. Generally, running models on TF-IDF yield better performance as compared to Word2Vec (Blue dots score higher than green dots) . This is probably due to the small window size used in training the Word2Vec model, hence the context might not be fully captured in the Word2Vec embeddings.

As our models were split into 4 main categories, we will look into the performance of some of these models in greater detail.

Linear Support Vector Classifier (SVC)

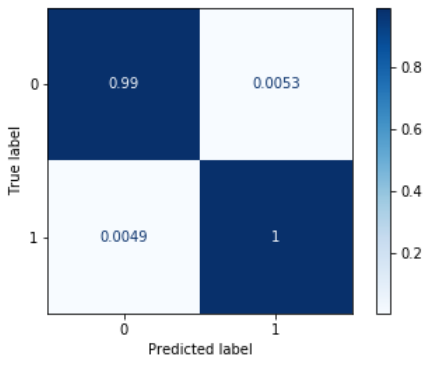

Figure 2: Confusion Matrix for Linear SVC on text using TF-IDF vectorizer

Linear SVC gave us a high F1-Score of 0.995, and a low FPR close to 0.005 on article texts with TF-IDF after tuning. The performance of SVMs, commonly regarded as one of the best text classification algorithms, can be attributed to the fake news detection problem being largely linearly separable. The classification hyperplane separates the vector representations of fake news from those of true news. Here, Linear SVC supports a linear kernel. It is faster and scales better. As these vectors are maximally separated, the model has great generalisation. It is also effective in high dimensional spaces, which is helpful with news texts because there is a large amount of words (features) in the text and title.

k-Nearest Neighbours (kNN)

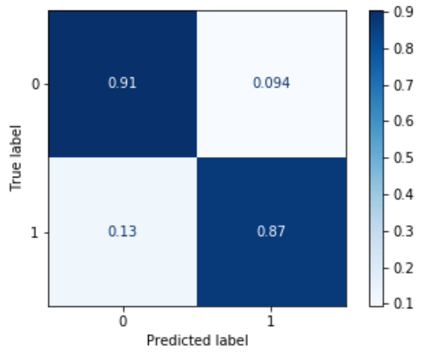

Figure 3: Confusion Matrix for kNN on text using TF-IDF vectorizer

With the feature vector of a particular news, the distance of a feature vector from all the other feature vectors will be calculated. K nearest vectors are selected and the most common class shall be the predicted label for the news. The kNN model did not perform relatively well, compared to prior models, but still yielded a F1-Score of 0.885 and FPR of 0.094 on article texts with TF-IDF after tuning. The downside of kNN is that it does not work well for high-dimensional data like the case in this news data, where the number of features is large. With larger dimensions, it becomes hard for the model to calculate feature similarity and find the nearest neighbours. Calculating distances for every news we want to classify is thus very costly, rendering this model not so useful in real-time prediction.

Decision Tree

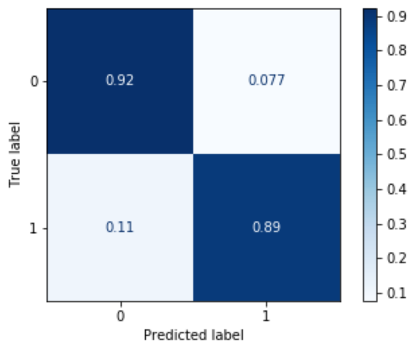

Figure 4: Confusion Matrix for Decision Tree on text using TF-IDF vectorizer

The Decision Tree model takes the values of the words (features) in the text and title, and splits them into groups by highest information gain, helping to discard irrelevant words. This is repeated until each group consists of a single classification - fake or true. The main issue is that it does not generalize well and has high variance with results varying from one training set to another. It is also relatively slower and is unable to capture interactions between words, leading to poorer performance due to the lack of contextualisation. We can see this from Figure 1, where its highest F1-score of 0.925, accomplished when trained and tested under Word2Vec for text, is worse performing than many other models trained and tested under different scenarios.

Random Forest

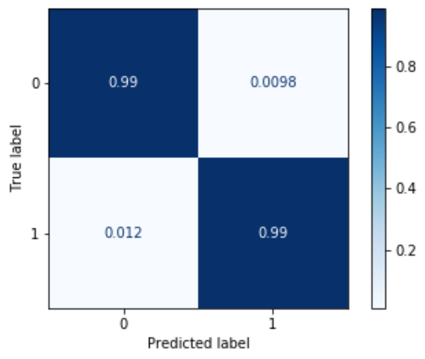

Figure 5: Confusion Matrix for Random Forest on text using TF-IDF vectorizer

The Random Forest model generally gave very high F1-Scores and low FPRs. As seen from Figure 1, all four runs for Random Forest attained higher F1-scores as compared to those of Decision Tree, with Random Forest being amongst the top three models with the highest F1-scores for text using the TF-IDF vectorizer. Random Forest uses a collection of Decision Trees. Rather than training the tree on all data, each tree gets a random subset of the data to be trained on. It lessens overfitting in the trees and introduces randomness with feature projection due to the variation across the trees thus it is more generalisable. With the most common classification as the final output, we attain a higher F1-Score of 0.988 for Random Forest compared to that of Decision Tree of 0.925. However, while Random Forests is more accurate, Decision Tree is more interpretable.

Discussion

Insights from WordClouds

With our preprocessed dataset, we create WordClouds using the 'wordcloud' package in Python to generate visual representations of words used in the dataset proportionate to the amount of their usage in fake and true news respectively. Words from news labelled as fake are in red while words from news labelled as true are in green.

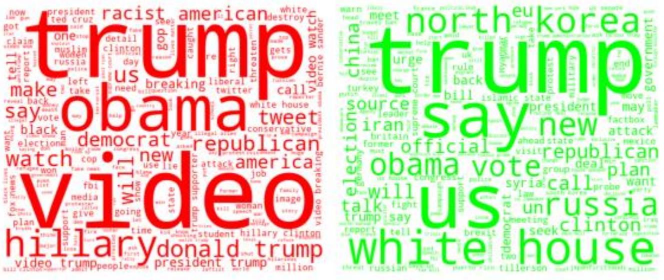

Figure 6: Common Words found in Fake (left) and True (right) news titles

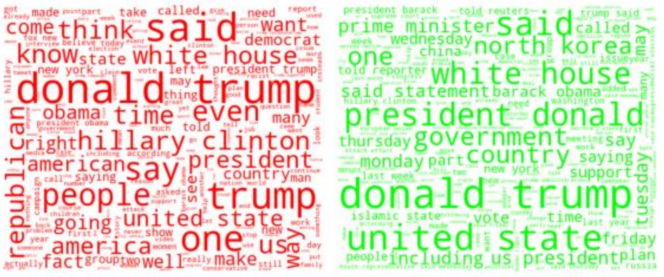

Figure 7: Common Words Found in Fake (left) and True (right) news texts

The dataset largely covers United States (US) politics, as seen from the use of words such as 'trump', 'obama', 'hillary clinton' which are prominent US politician names. In addition, the dataset consisted of news from 2015 to 2018, which includes the timeline of which the presidential campaign of Donald Trump and Hillary Clinton were formally launched and the presidency of Donald Trump.

While the WordClouds for news texts in Figure 7 do not have any notable difference, we observe the vast use of the words 'trump' and 'obama' in both fake and real news titles as seen in Figure 6, though 'obama' is used more frequently in fake news and less so in real news. A stark difference between the two would be the word 'video' found in fake news titles that seems to be as commonly used as the word 'trump'. As titles are designed to capture potential readers' attention, use of the word 'video' in the title could have been used to prompt readers to click or read the fake news article as many people prefer being able to get evidence of an article in the form of a video that is easy and faster to comprehend than reading an entire article. The use of the word 'video' could also make readers believe that it is proof of what the title says. Hence making the fake news more trustworthy and increasing its chances of being circulated amongst users.

WordClouds may be helpful in providing insights on the content of texts, but does little in helping to classify texts as 'real' or 'fake' because words describing similar topics tend to be used in both. We shall proceed to investigate the classification of texts with our Machine Learning models.

Weighted F-beta score as a metric

Initially, as mentioned under 'Evaluation Metric', we calculated F1-scores for all the models. With a beta coefficient of 1, false positives and false negatives have equal weightage in determining the performance of each model. Upon further analysis, we realised that the FPR played a greater importance when it came to comparing the performance of our various models as we do not want fake news to be wrongly identified as true. Using F1 would be detrimental as we only want to prevent the circulation of fake news but not facts. Hence we have decided to tweak our F1 metric by changing the beta coefficient to 0.5 in order to place greater emphasis on minimising the FPRs of each model.

Source removal from text

As previously mentioned, we speculate that our F1-scores has been inflated as the source of the news - the word "Reuters" was included in the majority of our True news. Reuters is an international news organisation which focuses on fair representation while still ensuring relevance of interest. They have a policy to take on an objective and neutral stance when reporting news. This adds to the credibility of Reuters as a source of trustworthy news. However, if we leave the word 'Reuters' in our text, it could have led to a source bias. Hence, we shall test to see if the removal of this source would have an impact on our models' performances. We will compare the F-beta scores obtained for our models before and after the removal of source.

Models	**Full-text: F-beta scores**	**Source removed: F-beta scores**
Naive Bayes	0.929	0.928
Linear SVC	0.995	0.988
KNN	0.835	0.865
Logistic Regression	0.984	0.978
Decision Trees	0.997	0.961
Extra Trees	0.979	0.974
Adaboost	0.997	0.959
Random Forests	0.988	0.979

Table 1: Comparison of F-beta scores obtained on full-text and text with source removed, using TF-IDF. Red text indicates a drop in F-beta scores.

As shown in the results above, across most models, we found that the F-beta scores decreased slightly when the source information was removed. This confirms our hypothesis that our F1-scores in the preliminary results have been inflated by the presence of the source information. The slight decrease in F-beta scores is most probably attributed to the fact that the text corpus still remains largely unchanged. In our removal of source, we only removed a few words from the text corpus which indicates source information such as "Washington - Reuters" and "New York - Reuters" which was commonly found in the beginning of the text while the rest of the text corpus remains untouched.

Nonetheless, this shows that hidden encodings within our dataset could skew our findings and more care should be given to remove these encodings. Upon closer inspection of our dataset, we realised that a lot of the fake news also had hidden encodings such as their source being mentioned at the end of the text, which could have played a role in skewing our results.

Truncated Text

Since most news tend to be well-summarised in the beginning, we wanted to investigate the effect of truncating the text feature, limiting it to the first 150 words and assess its impact on the model's performance. Building on our insights from source removal, we will take that into account when experimenting with truncated text.

Models	Source removed: F-beta scores	Truncated text: F-beta scores
Naive Bayes	0.928	0.925
Linear SVC	0.988	0.976
KNN	0.865	0.835
Logistic Regression	0.978	0.962
Decision Trees	0.961	0.886
Extra Trees	0.974	0.953
Adaboost	0.959	0.883
Random Forests	0.979	0.966

Table 2: Comparison of F-beta scores obtained on source-removed text and truncated text, using TF-IDF. Red text indicates a drop in F-beta scores.

From the results obtained, there is a decrease in performance across all models. We believe this is likely due to the fact that with truncated text, there are less words in the text corpus and since one metric of the TF-IDF is counting how many times a word appears in a text, it is unable to capture the true frequency of the words (especially words that distinguish fake news from true news, like superlatives) due to fewer words being present in the truncated version of the news texts. Hence, it is unable to evaluate how relevant these words are in relation to the news article.

We also noticed that for the decision tree model, the F-beta score for truncated data fell quite significantly as compared to the fall in F-beta score for the other models. This could be attributed to the fact that the decision tree, which is already known to easily overfit training data, had overfitted the truncated text data even more. As such when we tested the model on the entire length of text in the test set, it was unable to generalise well and gave poorer performance as compared to the other models. However it is worth noting that the score is still relatively decent and we will explore this further later on.

However, we also note that from the results obtained, the F-beta scores are still fairly high, indicating that a sufficiently high performing model can still be obtained even with a limited text corpus of 150 words.

Models	Source removed: F-beta scores	Truncated text: F-beta scores
Naive Bayes	0.701	**0.862**
Linear SVC	0.988	0.936
KNN	0.974	0.934
Logistic Regression	0.950	0.950
Decision Trees	0.757	**0.765**
Extra Trees	0.916	**0.930**
Adaboost	0.773	**0.778**
Random Forests	0.900	**0.932**

*Table 3: Comparison of F-beta scores obtained on source-removed text and truncated text, using Word2Vec. **Green text** indicates an increase in F-beta scores*

As shown in the results above, running the models using Word2Vec yields widely different results. We believe this is due to the fact that we are utilising Word2Vec's skip-gram model with a small window size, which works well even with little data as it is used to predict surrounding context words, given a particular word.

This could also be an indication that looking only at truncated text, which is likely to be a summary of the news article, allows Word2Vec to gain a sufficient understanding of the context of news articles to accurately come up with word embeddings such that the models can better distinguish between fake and real news.

A full summary of the results obtained is available in the Appendix section at the end of this report.

Conclusion

In addition to current related work on fake news detection, we have experimented with other less common machine learning models. We also investigated the impact of using truncated texts and F-beta scores as a metric that places more weight on false positives.

It is essential to choose a model that helps to identify the smallest misclassification rate on fake news in order to prevent these fake news from spreading while ensuring the real news does not get censored. By looking at our FPRs, we will be able to pick out the model that has the lowest rate, which can help us censor fake news most accurately out of all the models we have tested with.

We believe that having a machine learning approach to detect fake news is worth the effort as there is probably a clear underlying pattern that distinguishes fake from true news, especially with the use of superlatives and exaggerations, which are more commonly seen in fake news. We can see this by observing results from using decision trees. The results for this in tables 1 and 2 are from decision trees built with the default settings in sklearn, which implies a fully grown or unpruned tree. As seen from the results, its performance is still good after evaluation on the test set whether or not there is a source present or truncation has taken place. From what we have learned in class, this should not be the case, as decision trees are models that overfit easily and thus should give poor performance on the test set. This could mean that there is some similarity between the train and test set. While this could be due to the fact that the data is primarily from Reuters, it could also mean that there is some way to clearly divide fake from true news in general. Having shown this with our project pertaining to fake news detection, we think that this application of machine learning is one that is worth improving on in later iterations.

Shortcomings

For this project, we trained and tested our models using the ISOT dataset. However the ISOT dataset primarily contains news from the political sector, which is not representative of all sources of fake news in general. This constitutes a form of selection bias, which we can mitigate by training our models on more datasets or a more diverse dataset that covers more topics. We think the usage of just this dataset is a weakness of our project as our models will not be able to distinguish fake from true news, especially when we input news articles outside politics. Our generally high-scoring results stemming from default models serve as a warning that this is likely the case. We may not have gotten an accurate representation of our models' performances with this dataset that largely focuses on a specific topic.

We were also unable to perform ensembling by taking the inputs of other models, and blending or conditionally ensemble them. Doing so may have given us an even more accurate model that could outperform our current models and perform well with other real world datasets.

Further Improvements

This project can be further improved by combining our dataset with other fake news datasets to get a more diverse distribution of fake data for training. We may also experiment with deep learning models such as Recurrent Neural Network (RNN) in the future as its ability to capture sequential data makes it very appropriate for text classification due to the natural sequence found in English sentences.

In this project, we explored two NLP techniques to convert textual information into numerical representations. However, we might not have captured contexts which could be useful in distinguishing between real and fake news, since we only considered individual work tokens. In the future, we could experiment with n-grams tokenization.

Lastly, besides texts, the usage of audio recordings, images and videos as part of news dissemination, real or fake, has become increasingly popular as social media continues to grow. These are things that our models definitely did not account for but we see the importance of having models that take into consideration different forms of expressions so that fake news can still be detected without manual fact-checking, regardless of the form of expression used to misinform and deceive readers. Hence we could combine the use of RNN and CNN together to train both the text and image portion of fake and true news so as to be able to apply our model on more diverse data.

References

[1] Zhou, Xinyi and Zafarani, Reza. (2020) *A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities*

[2] Ahmed H, Traore I, Saad S. "Detecting opinion spams and fake news using text classification", Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January / February 2018.

[3] Ahmed H, Traore I, Saad S. (2017) "Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127- 138).

[4] Lorent, Simon. (2019) *Fake News Detection Using Machine Learning*
https://matheo.uliege.be/bitstream/2268.2/8416/1/s134450_fake_news_detection_using_machine_learning.pdf

[5] Baad, Dipika. (2020) *Sentiment Classification using Word Embeddings (Word2vec)*
https://medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca