Detecting Fake News with Supervised Learning

Benjamin R. Perucco

December 30, 2020

1 Definition

1.1 Project Overview

Our world is highly interconnected and it is of paramount importance that citizens are informed objectively about issues that influence and shape our world (like issues on geopolitics or climate change). The internet lead to a rise of news media (e.g. social media, news portals, etc.) to report on these stories. The vast amount of (online) articles available yield a new phenomenon called fake news. Fake news is false or misleading information presented as news and can reduce the impact of real news [1].

1.2 Problem Statement

This works' intention is about answering the question whether machine learning can be applied to classify a news article as truthful or fake. There are several articles available where Ahmed et al. classified news articles [2] or hotel guest reviews [3] to be truthful or fake. Ahmed et al. used natural language processing (NLP) models to transform text into a structured, mathematical representation and achieved accuracies of 92% [2].

Based on a set of features $\{f_1, f_2, f_3, ...\}$ extracted from news articles, a machine learning algorithm needs to be trained to find the relationship $\hat{\theta}$ between the a priori known classifiction of the news articles y (= 0 for truthful and = 1 for fake) and the set of extracted features $\{f_1, f_2, f_3, ...\}$. The relation can be written as

$$\hat{y} = \hat{\theta}(X) \tag{1}$$

where $\hat{y}$ is the predicted class label by the trained function $\hat{\theta}$ applied to a feature matrix X. $\hat{\theta}$ must be trained on available data, such that

$$\sum_{i=1}^{n} (y - \hat{y})^2 \tag{2}$$

is minimal. Here it is assumed that we have n news articles where it is already known whether they are truthful or not. A first step to be worked on in this report is to answer how to extract a feature matrix X from a set of texts (or documents). How reliable is the prediction of class labels on unseen news articles once the function $\hat{\theta}$ has been found.

1.3 Metrics

Equation 2 is rather used for regression problems where the predicted output is a continuous variable. For classification problems like news article classification into a truthful (= 0) and a fake class (= 1), the accuracy is a better metric. This can be nicely demonstrated by the confusion matrix

$$\begin{bmatrix} & 1 & 0 \\ 1 & \text{TP} & \text{FP} \\ 0 & \text{FN} & \text{TN} \end{bmatrix}. \tag{3}$$

The entries in the row indicate predicted classes and the entries in the columns represent the real classes. Furthermore, TP stands for true positives, TN for true negatives, FP for false positives and FN for false negatives. To asses the performance of $\hat{\theta}$, the accuracy a is considered, which can be calculated as

$$a = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}. \tag{4}$$

2 Analysis

2.1 Algorithms and Techniques

News articles must be converted into a structured, mathematical representation in order it can be used by machine learning techniques. The following chapters disucss how we can convert text into numerical features.

2.1.1 n-gram Model

The n-gram is usually defined a contiguous sequence of words with length n. For example, if $n = 1$, we speak of a unigram that contains only single word tokens. Or if $n = 2$, we denote this as a bigram which is built on two adjacent word tokens.

Consider the following text: "Sometimes we eat green apples, and sometimes, the apples we eat are red." Based on a unigram (1-gram), we obtain a set of tokens: {'sometimes', 'we', 'eat', 'apples', 'green', 'and', 'the', 'are', 'red'}. We can derive a frequency array of tokens in the text: [2, 2, 2, 2, 1, 1, 1, 1, 1]. For the bigram (2-gram), another set of tokens is obtained: {'sometimes we', 'we eat', 'eat green', 'green apples', 'apples and', 'and sometimes', 'sometimes the', 'the apples', 'apples we', 'eat are', 'are red'}. The corresponding frequency array of tokens in the text is: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]. Please note that punctuation is not considered when converting text into tokens.

In order to build frequency arrays for a set of texts (or documents), a common vocabulary needs to be built of which the n-gram model is underlying principle.

2.1.2 Vocabulary

Consider a corpus D which contains a set of documents $\{d_1, d_2, d_3, ..., d_n\}$. Then a vocubulary F is a set of tokens $\{f_1, f_2, f_3, ..., f_m\}$ extracted from the corpus D. Please remind yourself that a token is created based on the n-gram model. Usually for a set of tokens, only the m mostly occuring tokens in a corpus D are considered. In the following, the tokens are denoted as features as these construct the features (or independent variables) of a machine learning model.

2.1.3 Definitions

Let $\sigma(d_i, f_j)$ denote the number of occurances of feature f_j in document d_i. Then a feature matrix S can be built, where

$$S = \begin{bmatrix} \sigma(d_1, f_1) & \sigma(d_1, f_2) & \sigma(d_1, f_3) & ... & \sigma(d_1, f_m) \\ \sigma(d_2, f_1) & \sigma(d_2, f_2) & \sigma(d_2, f_3) & ... & \sigma(d_2, f_m) \\ \sigma(d_3, f_1) & \sigma(d_3, f_2) & \sigma(d_3, f_3) & ... & \sigma(d_3, f_m) \\ ... & ... & ... & ... & ... \\ \sigma(d_n, f_1) & \sigma(d_n, f_2) & \sigma(d_n, f_3) & ... & \sigma(d_n, f_m) \end{bmatrix}. \tag{5}$$

An element $\sigma(d_i, f_j)$ in matrix S (representing the document d_i and feature f_j) is abbreviated using the notation σ_{ij} for simplicity.

2.1.4 Term Frequency Model

The matrix S could be already used for machine learning. Features usually need to be normalized in machine learning to increase performance. Therefore, the term frequency (TF) model normalizes matrix S leading to matrix $\hat{S}$. An element $\hat{\sigma}_{ij}$ of matrix $\hat{S}$ is written as

$$\hat{\sigma}_{ij} = \frac{\sigma_{ij}}{\sum_{j=1}^{m} \sigma_{ij}}. \tag{6}$$

Or spoken in plain language: the number of occurances of a token f_j in a document d_i is divided by the total number of occurances of all tokens $\{f_1, f_2, f_3, ..., f_m\}$ in the same document d_i. So we end up with a representation where the importance of each feature can be compared to other features in the same document.

2.1.5 Inverse Document Frequency Model

The inverse document frequency (IDF) model is used to define the feature importance not just in a document d_i but also compare its importance within a corpus D. A matrix E is introduced, where an element ϵ_{ij} of matrix E is 1 if $\hat{\sigma}_{ij} > 0$ (or $\sigma_{ij} > 0$). An inverse normalization is applied to the matrix E resulting in a vector $\hat{e}$. An element $\hat{\epsilon}_j$ of vector $\hat{e}$ is calculated as

$$\hat{\epsilon}_j = 1 + \log \left[\frac{|D|}{\sum_{i=1}^{n} \epsilon_{ij}} \right]. \tag{7}$$

Or spoken in plain language: in a corpus D which comprises of a set of documents $\{d_1, d_2, d_3, ..., d_n\}$, it is counted in how many documents the feature f_j appears. This number is used to divide the number of documents $|D|$ in a corpus D. Consider two examples: if a feature f_j occurs in each document $\{d_1, d_2, d_3, ..., d_n\}$, we divide the number of documents $|D|$ in a corpus D by the same number. So equation 7 results in 1, weighting feature f_j as 1. On the other hand, if a feature f_j occurs only in one document, equation 7 produces a much larger number, thus increasing the weight of feature f_j in the corpus D.

Finally, a matrix $\hat{P}$ is obtained as the element-wise product of the term frequency matrix $\hat{S}$ and the inverse document frequency vector $\hat{e}$, written as $\hat{P} = \hat{S} \odot \hat{e}$. For an element $\hat{\pi}_{ij}$ of matrix $\hat{P}$, this is written as

$$\hat{\pi}_{ij} = \hat{\sigma}_{ij} \cdot \hat{\epsilon}_j, \quad \text{element-wise for } j = 1, 2, 3, ..., m. \tag{8}$$

This term is denoted as the term frequency inverse document frequency model (TF-IDF).

2.2 Data Exploration

A set of truthful and fake news articles is available on kaggle.com [4]. The data was collected from real world sources. Truthful articles were obtained from Reuters and fake news articles were gathered from unreliable websites that were flagged by Politifact which is a fact-checking organization. These datasets contain different types of articles on different topics, the majority of articles focus on political and world news topics [3].

There are 44,898 articles in total in the corpus. 21,417 of the articles are classified as truthful and 23,481 are classified as fake, so both class labels are roughly balanced. Approximately 6,169 of the articles are not unique. The duplicates are being removed after text processing discussed in chapter 3.1.

2.2.1 Visual Analysis of Top Features

A first glimpse of the top 20 unigrams per class is given in figure 1. Please note that the extracted unigrams are based on the processed text. It can be observed from the unigrams that probably the most discussed topic in the set of news articles is the presidential election in the U.S. during the year 2016. Despite some unigrams that are used frequently in both classes of articles (like said, trump, presid, republican, etc.), we also see distinct unigrams only used in one class of articles.

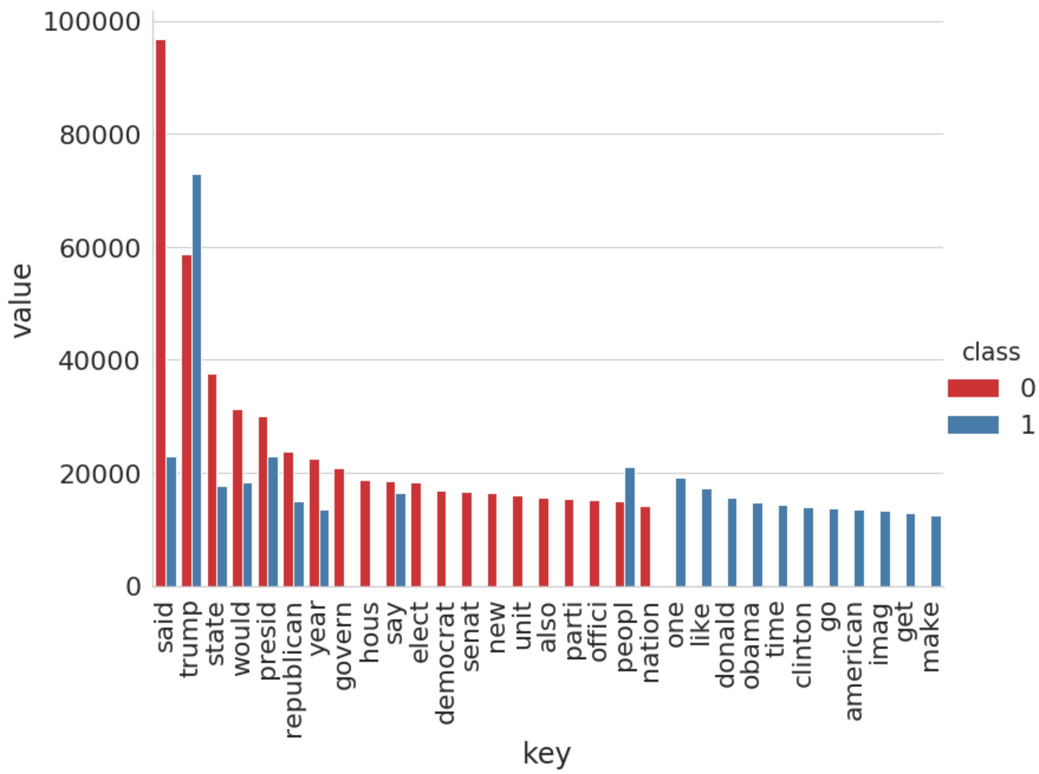

Figure 1: Frequency of the top 20 unigrams per class extracted from the corpus.

3 Methodology

3.1 Data Preprocessing

Raw text has to be brought into a clean state in order for feature extraction. Several text cleansing steps are performed like removal of introductory statements, dates, hyperlinks, numbers and stopwords. Finally, so called stemming is performed to end up with the stem of a word (for example cat should identify such strings as cats, catlike, and catty [5]).

3.2 Implementation

3.2.1 Feature Extraction Method

Then the randomized corpus is split into three different datasets as displayed in the table below.

Name	Purpose	Size	TF	IDF	TF-IDF
Training	Model training	60%	$\hat{S}_{\text{train}}$	$\hat{E}_{\text{train}}$	$\hat{S}_{\text{train}} \odot \hat{E}_{\text{train}}$
Test	Hyperparameter tuning	20%	$\hat{S}_{\text{test}}$	$\hat{E}_{\text{train}}$	$\hat{S}_{\text{test}} \odot \hat{E}_{\text{train}}$
Validation	Model validation	20%	$\hat{S}_{\text{valid}}$	$\hat{E}_{\text{train}}$	$\hat{S}_{\text{valid}} \odot \hat{E}_{\text{train}}$

It is good practise not to use the same data for model training and validation. For hyperparameter tuning a test dataset is used to avoid overfitting.

Term frequency (TF) feature matrices are calculated for each set according to equation 6. In comparison, the inverse document frequency (IDF) feature matrix is only calculated for the training set according to equation 7. The reason is that the validation and test set should simulate the performance behaviour in case of new unseen data. Therefore, the IDF feature matrix is estimated on a hypothetical training set and the resulting matrix $\hat{E}_{\text{train}}$ is used for the transformation of all sets into the TF-IDF feature space according to equation 8. Contrary, the TF feature matrix describes the relative importance of features in a single document. This is independent of other documents in the corpus and therefore, we can derive three matrices $\hat{S}_{\text{train}}$, $\hat{S}_{\text{test}}$ and $\hat{S}_{\text{valid}}$ for the three sets.

3.2.2 Modeling

SageMaker [6] is used as the training and deployment platform for the machine learning models. The evaluated machine learning models are based on the sci-kit learn framework [7] because of its simplicity to try out different machine learning models.

The following models k nearest neighbor (knn), support vector machine (svm), logistic regression (log), gradient boosting (gbc) and multilayer perceptron (mlp) are used. Furthermore, the TF and TF-IDF feature extraction methods are applied. The feature size is set to 500, 1000 and 5000 features and unigram and bigram are used for feature extraction.

3.3 Refinement

Hyperparameter tuning is performed on the training set and tested on the test set according to the accuracy (equation 4). A hyperparameter set is search for which maximizes the accuracy. For the hyperparameter search, 20 combinations are evaluted and in total this results in 1200 models to be trained and tested (5 models x 2 feature extraction methods x 3 different feature sizes x 2 differnt n-grams x 20 models for hyperparameter tuning). In the following table 1, the model hyperparameters as well as its varied ranges are shown.

Parameter	Type	Range
	k nearest neighbor [8]	
n neighbors	int	[3, 15]
weight	cat	'uniform', 'distance'
p	int	[1, 8]
	support vector machine [9]	
random state	int	1
kernel	cat	'poly'
C	float	[0.001, 3.0]
degree	int	[2, 3]
	logistic regression [10]	
max iter	int	10,000
C	float	[0.001, 3.0]
	gradient boosting [11]	
random state	int	1
learning rate	float	[0.001, 0.5]
n estimators	int	[100, 1000]
max depth	int	[2, 10]
	multilayer perceptron [12]	
random state	int	1
learning rate	float	[0.0001, 0.1]
max iter	int	[100, 10000]
activation	cat	'identity', 'logistic', 'relu', 'tanh'
hidden layer size	int	[2, 10]
start size[1]	int	[10, 1000]
end size[1]	int	[10, 1000]

Table 1: [1] Note that start and end size are not available as parameters in the multilayer perceptron model in scikit-learn. Based on the hidden layer size and its start and end size, a linear interpoliation is performed for the number of neurons in between in case the hidden layer size is higher than 2.

4 Results

4.1 Model Evaluation and Validation

4.1.1 Top 5 Accuracy Performance

4.1.2 Influence of Parameter Variation

In total there are 60 parameter variations studied (5 models x 2 feature extraction methods x 3 different feature sizes x 2 differnt n-grams). Per parameter combination, the model is taken which leads in the highest accuracy after hyperparameter tuning. The influence of the different parameters is demonstrated below. In figure 2, the model performance can be seen for the five machine learning models evaluated.

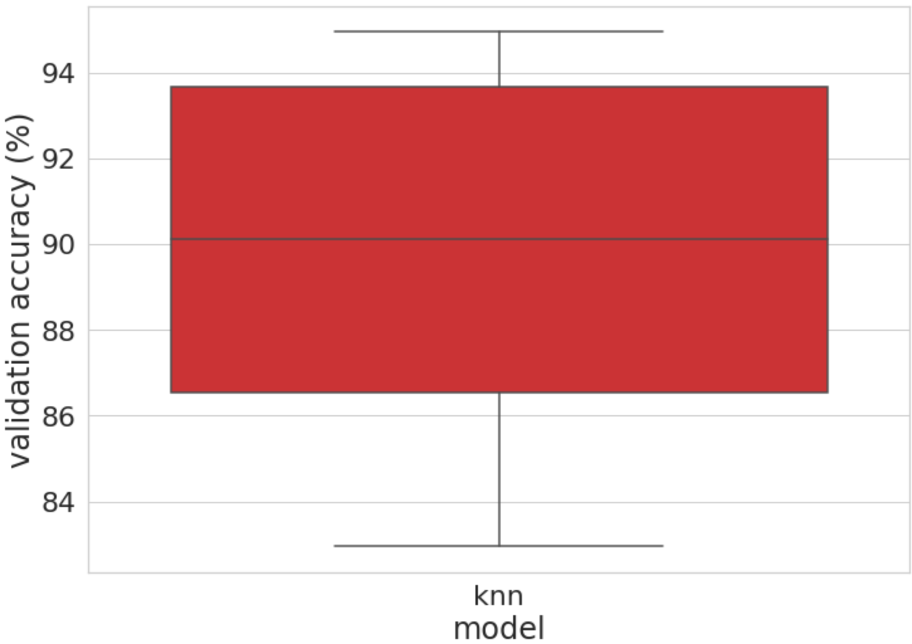

Figure 2: Model performance of five evaluated machine learning models.

4.2 Justification

5 Conclusion

5.1 Reflection

5.2 Improvement

References

[1] Fake news, December 30, 2020, wikipedia.org,
https://en.wikipedia.org/wiki/Fake_news

[2] 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. https://doi.org/10.1007/978-3-319-69155-8_9

[3] Ahmed, H, Traore, I, Saad, S. Detecting opinion spams and fake news using text classification, Security and Privacy, 2018, https://doi.org/10.1001/spy2.9

[4] Fake and real news dataset, December 30, 2020, kaggle.com,
https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

[5] Stemming, December 30, 2020, wikipedia.org,
https://en.wikipedia.org/wiki/Stemming

[6] Amazon SageMaker, December 30, 2020, amazon.com,
https://aws.amazon.com/sagemaker

[7] scikit-learn - machine learning in Python, December 30, 2020, scikit-learn.org, https://scikit-learn.org/stable

[8] scikit-learn - classifier implementing the k-nearest neighbors vote, December 30, 2020, scikit-learn.org, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[9] scikit-learn - c-support vector classification, December 30, 2020, scikit-learn.org, https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[10] scikit-learn - logistic regression classifier, December 30, 2020, scikit-learn.org, https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[11] scikit-learn - gradient boosting for classification, December 30, 2020, scikit-learn.org, https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[12] scikit-learn - multilayer perceptron classifier, December 30, 2020, scikit-learn.org, https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html