# Machine Learning Project Report: Fake News Detection

Philipp Hoffmann, Martin Wagner
Applied ML – Ludwig-Maximilians-Universität München
philipp.hoffmann@campus.lmu.de, wagner.mar@campus.lmu.de

July 14, 2025

## 1 Task Overview

Due to the enormous amount of news published every day, distinguishing fake from real news is becoming an impossible task for humans, making the usage of computer models indispensable. In this project, we tested different models and text preprocessing methods using the "fake-and-real-news-dataset"[1]. This dataset collects title and text of over 40000 news articles classified as fake or true news.

## 2 Methods

We compared the performance of logistic regression and linear svm models (mainly on the collection of titles) using different vectorization techniques from natural language processing. We used `TfidfVectorizer` and `CountVectorizer` from `scikit-learn` for basic text vectorization, which handle tokenization including lower-casing and punctuation removal.

Given a vocabulary of tokens (e.g. words), bag-of-words assigns to a document $d$ for each token $t$, the term-frequency $\mathrm{tf}(t,d)$, defined as the count of term $t$ in $d$ divided by the total number of terms. A bit more evolved is Tf-Idf-vectorization, which also takes into account the frequency of how often a term occurs in a fixed corpus $D = \{d_1, \ldots, d_n\}$ of documents $d_1, \ldots, d_n$. Given a term $t$, the inverse-document frequency of $t$ in $D$ is $\mathrm{idf}(t,D) := 1 + \log\left(\frac{1+n}{1+\mathrm{df}(t)}\right)$, where $\mathrm{df}(t)$ is the number of documents in $D$ containing the term $D$. Then the Tf-Idf-score of a term $t$ with respect to the collection $D$ is defined as

$$\text{tf-idf}(t,d,D) := \mathrm{tf}(t,d)\,\mathrm{idf}(t,D).$$

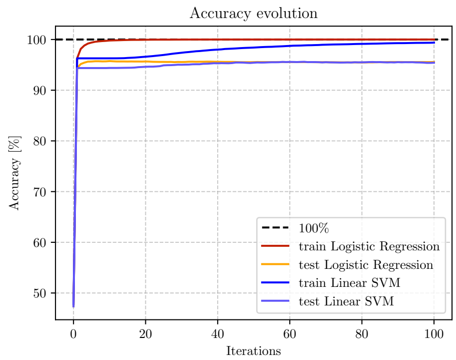
Note that $d$ does not need to be contained in the collection $D$ of documents itself. The tools from `scikit-learn` also apply $L2$-normalization to the obtained vectors, which we retained. Additionally, we applied z-score normalization afterwards, as this gave us better results, in particular for linear SVMs. We adopted models in courselib to support sparse matrix calculation and implemented methods to adopt for feature shifts (due to z-score normalization) without destroying the computational efficiency resulting from the sparsity of the unshifted feature matrices. Moreover we implemented custom tokenizers using `nltk` that support for example stemming or lemmatiztation, i.e. reduce words to their basic form. To test all of these options more compactly, we implemented a multi-column-vectorizer which supports vectorization of text data of one or multiple columns of a dataframe with different vectorization options.

## 3 Experiments and Results

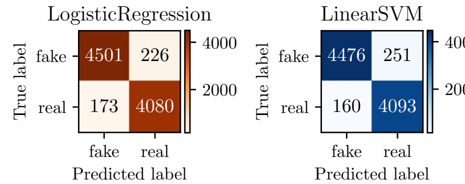
As both bag-of-words and tf-idf yield highly sparse, high-dimensional vectors, one expects the problem to be linearly separable, thus we restrict our experiments to linear models, namely logistic regression and linear SVM. For all our experiments, we used a 80/20 train-test-split.

We compared the models on tf-idf-vectorized text data with good performing learning rates with full batch size. While both models achieved almost perfect train accuracy, logistic regression performed slightly better and faster. Smaller batch sizes could not improve results, but slowed down the training time significantly (see Figure 1). Because of this we only used the logistic regression model for further experimentation.

[1] `https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset`




(a)



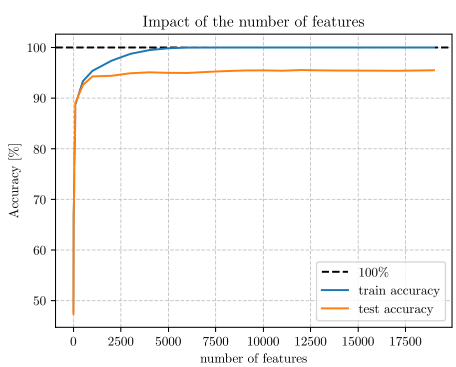

(b)

| Model | batch size | Train Accuracy [%] | Test Accuracy [%] | Training Time [s] | Precision | Recall | F1-Score |
|---|---|---|---|---|---|---|---|
| Logistic Regression | full | 100.0 | 95.56 | 0.68 | 0.95 | 0.96 | 0.95 |
| Linear SVM | full | 99.38 | 95.42 | 0.91 | 0.94 | 0.96 | 0.95 |
| Logistic Regression | 100 | 100.0 | 95.19 | 9.2 | 0.94 | 0.96 | 0.95 |
| Linear SVM | 100 | 96.59 | 92.19 | 23.95 | 0.91 | 0.92 | 0.92 |

(c)

Figure 1: Model comparison between logistic regression and linear SVM. (a): Accuracy evolution on full batch, (b): corresponding confusion matrices, (c): comparison mini batch and full batch

For the second part of our experiment, we tested different vectorization methods. Figure 2(a) shows how feature number impact accuracy: In general more features result in better accuracy, but more than 5000 features yield only minor improvement. This trend is also confirmed by Figure 2(b), when using text or text and titles as data, the number of feature significantly increases while also resulting in even better test accuracy.




(a)

| Data | Features | Train Accuracy [%] | Test Accuracy [%] | Vectorization Time [s] | Training Time [s] |
|---|---|---|---|---|---|
| title | 19562 | 100.0 | 95.56 | 0.56 | 0.52 |
| text | 111364 | 99.98 | 98.44 | 11.71 | 4.14 |
| title+text | 130926 | 100.0 | 98.95 | 12.21 | 4.56 |

(b)

| n-gram range | Features | Train Accuracy [%] | Test Accuracy [%] | Vectorization Time [s] | Training Time [s] |
|---|---|---|---|---|---|
| 1-1 | 19562 | 100.0 | 95.56 | 0.55 | 0.51 |
| 1-2 | 211984 | 100.0 | 95.6 | 1.55 | 0.99 |
| 1-3 | 495860 | 100.0 | 93.92 | 2.71 | 1.56 |
| 2-2 | 192422 | 100.0 | 91.8 | 1.22 | 0.72 |
| 3-3 | 283876 | 100.0 | 71.71 | 1.38 | 0.83 |

(c)

Figure 2: (a): impact of number of features, (b): performance with different data, (c): Influence of n-gram ranges, n-gram range given as minimum length - maximum length

With that in mind we compared tf-idf as well as bag-of-words vectorization using four different tokenization methods, with and without stop word removal. As Section 3 highlights, all combinations achieved similar performance, but stop word removal slightly decreases test accuracy. Moreover our custom tokenizers improved test accuracy slightly but the vectorization time increased heavily with the complexity of the tokenizer.

| Vectorization | Stop Words | Tokenizer | Features | Train Accuracy [%] | Test Accuracy [%] | Precision | Recall | F1-Score | Vectorization Time [s] |
|---|---|---|---|---|---|---|---|---|---|
| tf-idf | None | default | 19562 | 100.0 | 95.56 | 0.95 | 0.96 | 0.95 | 0.58 |
| tf-idf | None | basic | 24897 | 100.0 | 96.65 | 0.96 | 0.97 | 0.96 | 5.04 |
| tf-idf | None | stemming | 17293 | 100.0 | 96.49 | 0.96 | 0.97 | 0.96 | 11.57 |
| tf-idf | None | lemmatization | 20847 | 100.0 | 96.71 | 0.96 | 0.97 | 0.97 | 33.72 |
| tf-idf | english | default | 19288 | 100.0 | 94.4 | 0.94 | 0.95 | 0.94 | 0.57 |
| tf-idf | english | basic | 24625 | 100.0 | 95.9 | 0.95 | 0.96 | 0.96 | 5.02 |
| tf-idf | english | stemming | 17154 | 100.0 | 95.57 | 0.95 | 0.96 | 0.95 | 10.52 |
| tf-idf | english | lemmatization | 20799 | 100.0 | 95.97 | 0.96 | 0.96 | 0.96 | 29.79 |
| bag-of-words | None | default | 19562 | 100.0 | 95.59 | 0.95 | 0.96 | 0.95 | 0.57 |
| bag-of-words | None | basic | 24897 | 100.0 | 96.69 | 0.96 | 0.97 | 0.97 | 5.03 |
| bag-of-words | None | stemming | 17293 | 100.0 | 96.44 | 0.96 | 0.97 | 0.96 | 11.57 |
| bag-of-words | None | lemmatization | 20847 | 100.0 | 96.68 | 0.96 | 0.97 | 0.97 | 33.69 |
| bag-of-words | english | default | 19288 | 100.0 | 94.57 | 0.94 | 0.95 | 0.94 | 0.58 |
| bag-of-words | english | basic | 24625 | 100.0 | 95.94 | 0.95 | 0.96 | 0.96 | 5.03 |
| bag-of-words | english | stemming | 17154 | 100.0 | 95.59 | 0.95 | 0.96 | 0.95 | 10.55 |
| bag-of-words | english | lemmatization | 20799 | 100.0 | 96.01 | 0.96 | 0.96 | 0.96 | 29.85 |

Table 1: Comparison of different vectorization configurations with four different tokenizers. default: default tokenization from `TfIdfVectorizer` without custom tokenizer, basic: `nltk.word_tokenize` combined with punctuation removal and lowercasing, stemming: basic tokenizer combined with `SnowballStemmer` from `nltk`, lemmatization: basic tokenizer combined with `WordNetLemmatizer` from `nltk`.

Lastly we investigated which words or short phrases the model identified as most relevant for distinguishing between real and fake news (Figure 3). For this we trained the model on different combinations of n-grams. Figure 2(c) shows that training only with bi- and especially trigrams decreased test accuracy significantly.

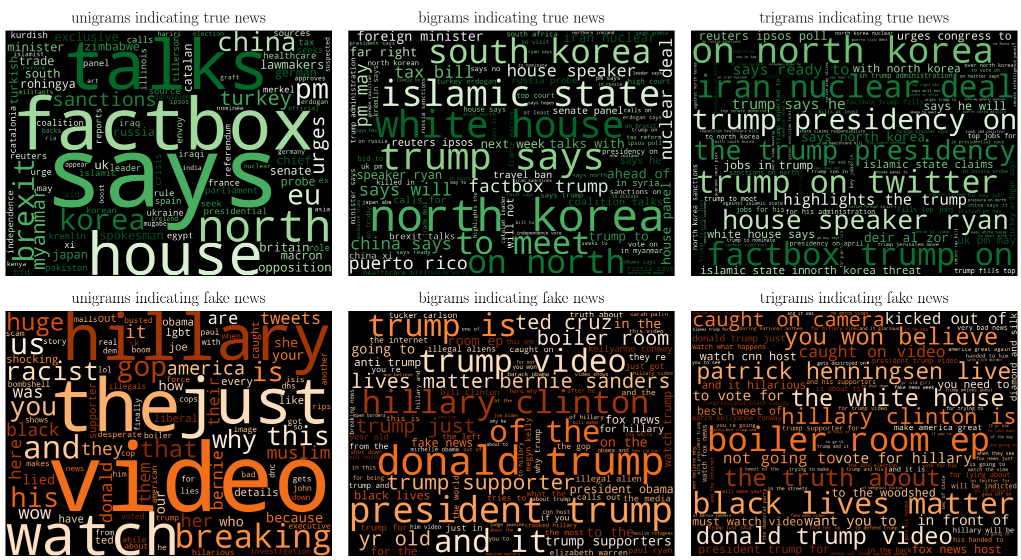



Figure 3: Illustration of most important n-grams identified by the model trained only on uni-, bi-, trigrams resp.

## 4 Discussion

Our experiment shows that linear models are indeed well suited for detecting fake news, even when analyzing only news titles. If one additionally also trains the model on the whole body of text, almost perfect classification is achievable. Removing stop words from the titles seems to worsen the performance, which either could indicate that titles become too short or that they are more common in fake or real news. It seems like machine learning models prefer single words for classification, whereas humans usually need context for understanding text. However our models only give binary classification and cannot detect the exact passages containing fake news, which could be an interesting objective for future research. A starting idea for this could be to divide the text into smaller paragraphs.