FAKE NEWS PROJECT (DATA SCIENCE)

TECHNICAL REPORT

Aske Lundsgaard PWG445
Henrik Christensen DJL587
Linnea Andersen WZP568
Mads Vestergaard Jensen NDH409

https://github.com/henrikdchristensen/FakeNewsDetection

March 31, 2023

1 Data preprocessing

The FakeNewsCorpus (FNC) is a large (>27 GB, $\sim$8.5M news articles) data set containing labeled articles (Szpakowski, 2020). It consists of 17 columns of which columns like keywords and tags are missing values for many of the articles. Since the data set is so large, it is difficult to train with the entire data set due to memory constraints. Thus, the aim is to use about $\sim$10% of the data set. We begin by shuffling the large data set to avoid the data being sorted in a particular way (e.g. by date, domain name, etc.) and then select 10% of the shuffled data set. To achieve this, we convert the CSV file to H5 (a performant binary tabular format) using the library `pandas` and `h5py`. After shuffling the rows in the H5 file, we convert it back to CSV, see diagram in Appendix B.1.

Next, we remove unwanted columns and rows such as empty content or label or rows not of interest such as rows with the label "rumor", which is not described in the corpus' description. After the removal of these rows we are left with $\sim$7,3M rows, see Appendix B.2.

1.1 Data cleaning

Next we clean, tokenize, remove stop words and stem the content of the news articles. For cleaning, we used regular expressions to replace dates, numbers, urls, special characters and most punctuation using the `re` library. For tokenization, we found the built-in `split()` function sufficient. For stop words and stemming we have used the `NLTK` framework. Not only is this framework widely used, but it also offers robust support for the English language.

The size of our vocabulary is 1,262,429 words. Considering that this includes all forms of words, names and misspellings, this seems reasonable compared to the size of the English language vocabulary. Removing stop words and stemming reduce the size of the vocabulary by 0.01% and 17.24%, respectively. Since we remove 179 stopwords, it logically does not reduce the vocabulary by much, while stemming reduces it a lot since many words have the same word stem.

1.2 Balancing, splitting and representation

Before splitting the data into a training, validation and test set, we made it possible to balance the data set in regards to labels. We do this in two ways: By original label and by binary label, see Section 2.1. Thus, we can train the model using both the unbalanced and balanced training data. Through all training, we have used a 80/10/10 split.

We represent the data set using a `pandas` data frame, which is a tabular format. The `pandas` library are chosen for various reasons: It is easy to visualize, easy to work with, CSV files are easily processed, powerful for analyzing data, and efficient if the data fits into memory (Team, 2019).

1.3 Data exploration

First, we explore the data to discover patterns, which can aid us in distinguishing between fake and reliable articles. When exploring the contributions of the different domains to each type of label, see Figure C.3, we observed that for most of the labels, the main contributions stem from a single or two domains: The domain *beforeitsnews* contribute with more than 85% of the fake articles, while the *nytimes* contribute around 80% of the reliable articles. This can make it more difficult to generalize our model for other data sets that contain articles from different sources.

Next, we investigated the number of words of the raw, uncleaned articles. Figure 1 reveals a multitude of outliers for both labels with the most comprehensive articles coming in at 33,936 words and 18,941 words for the reliable and fake article set respectively. The mean for the reliable articles is 512 words, while for the fake articles, the mean is 425 words, see Appendix C.1. Thus, there is not a huge difference in the content length between the reliable and the fake articles. We considered removing outliers e.g. articles longer than ∼2,500 words, however we ended up not doing this as we decided to preserve the whole data set.

After cleaning, we have looked at the 50 most frequent words for each label. In Figure 2, we depict these words using a Venn diagram to demonstrate which words intersects and which are unique to each label. While the word 'trump' and 'iran' are two of the most frequent for the fake articles, words like 'world', 'govern' and 'news' are most frequent for the reliable articles. Common words include 'american', 'presid' and dates, numbers and urls. This gives us an insight into what topics the two article classes concern, while also suggesting that there may not be a big difference between how many numbers, dates and urls each article includes. Contrary to our expectations, the reliable articles have '!' as one of the most frequent words. Refer to Appendix C.2 for a list of the 25 most frequent words for each label.

Furthermore, we investigated punctuation, <num> count, sentence length (Appendix C.6), negations and pronouns (before removing these as stop words). This exploration did not convince us that they could be used for our classifier as they were all quite similar.

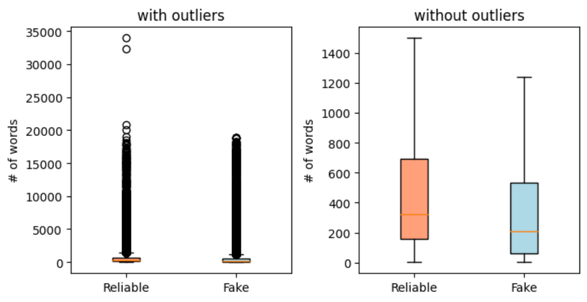


Figure 1: Number of words in reliable and fake articles

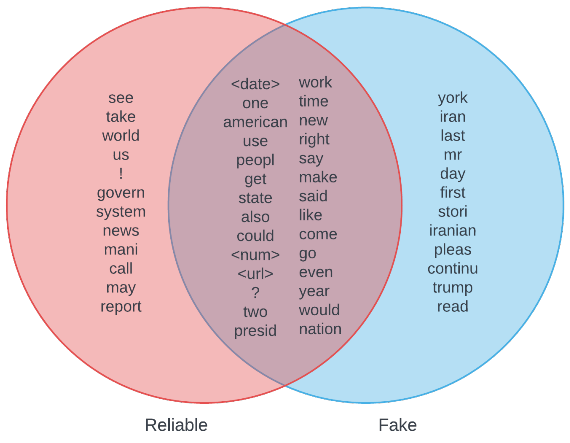


Figure 2: Top 50 most frequent words for each label, intersected and unique

2 Simple model

2.1 Grouping the labels

The FNC has 11 different labels, which we reduce to two labels in order to train a binary classification model. We group the labels into the following based on the description given in the FNC repository (Szpakowski, 2020):

`Reliable`: *reliable, clickbait, political*

`Fake`: *fake, conspiracy, junksci, hate, unreliable, bias, satire*

Besides the obvious categorizations, we have chosen to categorize *clickbait* and *political* as reliable. Even though *clickbait* articles use exaggerated headlines to promote themselves and *political* articles support certain, political standpoints, both types of articles generally provide credible information. *Bias* articles similarly support certain views, but may on the contrary be based on propaganda. Arguably, *unreliable* articles may be credible, but as they require further verification, we have chosen to categorize them as fake. This uncertainty of whether the content is credible will most likely act as a limitation on our model when using it on other data sets, since our model will have learned to label credible-but-unverified content as fake.

It is important to note that since we are training a fake news detector, we are looking to find fake news and thus, we label fake articles as the positive class (`True`), while reliable articles are labeled as negative class (`False`) - this is e.g. important when calculating precision and recall. In this report, however we will continue using 'reliable' and 'fake' as terms.

2.2 Models based on content

For simplicity and efficiency reasons, we chose to train our simple models using a Bag-of-Words (BoW) representation. In total, we trained seven simple baseline models. Their respective performances are displayed in Table 1. We decided to include these models to attain a diverse selection of simple models that could be easily and quickly trained. In order to prevent leakage, we created the BoW vocabulary only from the training data. Additionally, we took strict measures to limit the training of models solely to the training set.

The untuned LogisticRegression (LR) model was effective in distinguishing between reliable and fake news articles. This is not surprising given its special design for binary classification problems as it provides a probabilistic interpretation of the input features which are based on the content BoW.

Model name	Train acc. [%]	Val. acc. [%]	Precision [%]	Recall [%]	F1 [%]	Time [s]
(BM) LogisticRegression, tuned	89.8	87.4	89.2	86.1	87.6	366
LogisticRegression, untuned	90.0	87.2	89.1	85.9	87.5	369
PassiveAggressiveClassifier	89.8	80.6	79.2	80.5	80.0	50
DecisionTreeClassifier	65.2	65.3	59.1	89.7	71.2	104
AdaBoostClassifier	65.2	65.3	59.1	89.7	71.0	283
RandomForestClassifier	68.6	68.5	75.7	50.1	60.4	24
KNeighborsClassifier	98.1	54.6	100	10.6	19.2	750

Table 1: Simple models and their performance on the training data and validation data, sorted by F1-score. The tuned logistic regression model was chosen as the final baseline model (BM).

In the tuned LogisticRegression, we optimized the c-parameter of the logistic model using the `GridSearchCV` library, which resulted in only a slight improvement (final c was 0.1). This model will be referred to as the 'baseline model'.

Regarding the F1-score, the untuned LogisticRegression model exhibits the strongest performance, whereas the K-Neighbors classifier has the weakest performance. Despite attempts to improve the model's performance by balancing the training data according to the types, binary labels and only considering articles labeled as 'fake' and 'reliable', no significant improvement in the F1-score was achieved. Results from trying this can be found in Appendix F.

2.3 Models including metadata

We enhanced the performance of the Logistic Regression model by including metadata: The title, author and domain. The results can be seen in Table 2.

The FNC articles are classified based on their domain, not by actually fact-checking their content (Szpakowski, 2020). Thus, the domain and label are fully correlated, and it would be a huge advantage to include the domain name as a feature. From the results, we see that this expectation holds as our baseline model achieves a validation accuracy close to perfect.

Including the author, we expect it will similarly help the model. We see in the results, that this also holds. Giving the model the title as well will likely give the model a better idea of the subject of the article, which also improves the performance - as can be seen in the results.

Meta fields included	Train acc. [%]	Val. acc. [%]	Precision [%]	Recall [%]	F1 [%]	Time [s]
Domain, authors, title	99.9	99.6	99.7	99.5	99.6	332
Domain, content	99.9	99.3	99.4	99.3	99.4	335
Author, content	94.7	92.3	93.8	91.2	92.5	334
Titles, content	91.1	88.2	89.4	87.8	88.6	383
Baseline model (only content)	89.8	87.4	89.2	86.1	87.6	357

Table 2: Simple logistic models including metadata and their performance on the training data and validation data sorted by F1-score

Although including this metadata, both separately and combined, clearly improves our model, we want to train a classifier that can also classify cross-domain articles. Thus, including metadata from the FNC will generalize poorly and we continue the development of our model without using metadata.

3 Advanced models

3.1 Logistic Regression models (LR)

Our baseline model is trained using input features represented as a Bag-of-Words (BoW) of the content data. However, it is often more effective to incorporate term frequency-inverse document frequency (TF-IDF) weights into the model, as this accounts for the frequency of each word, resulting in a more accurate representation of the significance of rare words. The performance improvement resulting from the utilization of TF-IDF weights is illustrated in the top of Table 3.

Furthermore, we also experimented with representing the input using bigrams and trigrams. However, we found that memory usage and the training time significantly increased with these representations. Thus, it was only possible to train the 'LR, TF-IDF, trigrams'-model on 100k rows, while the others were trained on 800k rows. This likely explains why this model performed the worst.

3.2 Neural Networks (NN)

In addition to optimizing our logistic model, we also trained different neural networks. We wanted to see if they could benefit from the relatively vast amount of training data available as deep learning has yielded state-of-the-art performance in other fake news detectors (Hu et al., 2022). We constructed 5 different models, A-D, the performance of which can be seen in the bottom of Table 3. To prevent data leakage, we only train and build vocabularies using the training data. We will now explain the reasoning behind the chosen models.

Model name	Train acc. [%]	Val. acc [%]	Precision [%]	Recall [%]	F1 [%]	Time [s]
(AM) LR, TF-IDF, bigram	94.5	90.4	90.2	91.6	90.9	1936
LR, TF-IDF, unigram	89.4	87.9	88.2	88.6	88.4	198
LR, TF-IDF, trigram (100K)	97.8	82.5	80.5	88.0	84.1	470
Embedding, LSTM, dropout (C)	90.4	88.3	86.4	91.9	89.1	-
Embedding, LSTM (B)	89.1	87.8	87.7	89.1	88.4	-
Embedding, CNN (E)	88.5	87.4	90.3	84.9	87.5	-
Embedding, pooling (A)	89.3	86.4	85.1	89.4	87.2	-
GloVe embedding (D)	74.2	71.1	73.2	85.2	78.7	-

Table 3: Logistic TF-IDF models and neural network models and their performance on training and validation data. Note that it was only possible to train the LR, TF-IDF, trigram on 100k rows, while the others were trained on $\sim$800k rows

In Model D, we apply a pretrained Global Vector (GloVe) embedding matrix in the embedding layer of the network. One advantage of using pre-trained embeddings is that they provide a good starting point for the Embedding layer, which can accelerate the training process and improve the accuracy of the model. However, we have a quite large data set available, and for our models, training the embedding layer from scratch led to better results as it allowed the model to adapt to the specific characteristics of our dataset, resulting in improved accuracy.

The embedding layer learns an embedding vector with m dimensions for each word and applies this embedding for the first n words in each news article, thus generating a m x n matrix (dimensions x n first words). We selected a 16 dimensional vector and a max sequence length of 60 representing the first 60 words from each article. This was mostly due to computional complexity concerns, so raising these numbers could have yielded better results. We chose not to use a custom pretrained TF-IDF vector as weights for this layer as TF-IDF vectors do not capture the semantic relationships between words in the same way that word embeddings do ("A Guide on Word Embeddings in NLP", n.d.).

Model A is a simple feedforward neural network that uses a Global Average Pooling layer and a Dense layer with 16 units and ReLU activation, which applies a linear transformation to the output of the pooling layer. Then we have a final Dense layer with a single unit and sigmoid activation, which produces a probability score that indicates the likelihood of the input belonging to a particular class (true, false). The Global Average Pooling layer calculates the mean of each feature map across all its spatial locations, resulting in a single value for each feature map. This technique can be useful in reducing overfitting by reducing the number of parameters in the network while still preserving the most important patterns. This model has much better accuracy than model D but still has a lower accuracy, precision and F1 score the other of the models.

In Model B, we incorporate both an embedding layer and a Long-Term Short Memory (LSTM) which is a type of recurrent neural network that preserves long-term dependencies in sentences, enabling the model to classify sentences and longer texts as opposed to just individual words (Olah, 2015). It outperforms Model A in all metrics except for recall. This indicates that the addition of the LSTM layer, which utilizes 100 memory cells, has enhanced the model's ability to predict both positive and negative cases in the dataset.

Model C is a tweaked version of model B and has the highest scores across all metrics. This can be attributed to the use of hidden dropout layers. Dropout is a regularization technique used to prevent overfitting by randomly dropping out (i.e., setting to zero) some of the neurons in a layer during training (we use a dropout rate of 0.3 as this yielded the best results on the validation set). By adding a dropout after each layer, we can force the LSTM and dense layers to learn a more robust representation of the input and avoid overfitting on the training data.

For Model E, we tried a 1d convolutional neural network. The main idea behind a CNN is to detect local patterns or features in the input data by applying a set of filters to the data - we apply 128 filers of size 5. By using multiple filters, the Conv1D layer can hopefully learn to detect different patterns or features at different positions of the input sequence. Next we apply a GlobalMaxPooling1D to select the most outstanding features and to avoid overfitting. We also apply a dense layer with 64 neurons that uses ReLU to add non-linearity to the model enabling it to learn more complex patterns in the input data. This model performed fairly well achieving the highest precision of 90%, but a recall of only 84% and a F1 score of 87,5%.

The best advanced model was the 'LR, TF-IDF, bigram'-model, which achieved an validation accuracy of 88.3% and a F1-score of 90.9%. From now on, this model will be referred to as the 'advanced model'. However, the best NN model was very close (Model C), which achieved an accuracy of 88.3% and a F1-score of 89.1%. For further details about the parameters of our models, refer to Appendix D.

4 Evaluation

4.1 FakeNewsCorpus test data

We evaluate our models on the test data. The performance can be seen in Table 4. The advanced model performs the best with 90.9% F1-score, compared to the baseline's 87.5% F1-score. In addition, the advanced model achieves an accuracy of 90.4% and the baseline achieves an accuracy of 87.1%. Overall the Logistic Regression (baseline model) achieves more or less the same evaluation metrics as the models in Table 3, which also shows that non-deep models can do fairly well and might suffice in many tasks. When it comes to fake news detection, we generally tolerate false positives (predicting a reliable article to be fake) more than false negatives (predicting a fake article to be reliable). Thus, we should focus on getting a high recall as it implies that the classifier has few false negatives. The advanced model has the highest recall of 91.5% for FNC, while the LR, TF-IDF, trigram model has the highest recall of 67.4% on LIAR.

Model name	Test acc. [%]	Precision [%]	Recall [%]	F1 [%]
Advanced	90.4	90.3	91.5	90.9
Baseline	87.1	89.1	86.1	87.5

(a)

Model name	Liar acc. [%]	Precision [%]	Recall [%]	F1 [%]
LR, TF-IDF, trigram	54.9	58.3	67.4	62.5
Advanced	52.2	58.1	51.2	54.4
Baseline	48.1	60.9	19.2	29.3

(b)

Table 4: Evaluation of our models on (a) the FNC test data and (b) the LIAR data set. In (b), we include the LR, TF-IDF, trigram model since this model performs better than our advanced model on LIAR

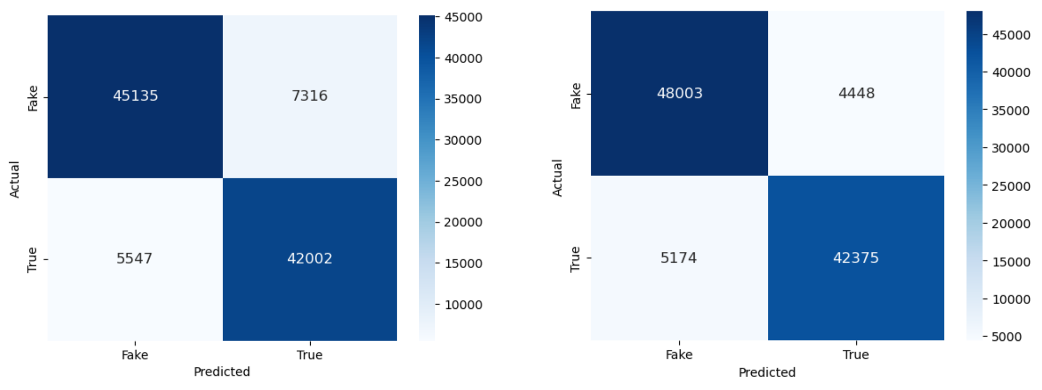


Figure 3: Confusion matrix for baseline and advanced model, respectively

We also take a look at our results using a confusion matrix as shown in Figure 3. From this, it is clear that our advanced model more often predicts correctly. We have the precision of predicting the positive class (fake news) from Table 4a ($P_{pos\_AM} = 90.3\%$), and from the confusion matrix, we can calculate the precision of predicting the negative class (reliable news) as:

$$P_{neg\_AM} = \frac{TN}{TN + FN} = \frac{42375}{42375 + 4448} = 90.5\%$$

Thus, they are nearly the same, which indicates that the model is not biased towards one particular class. For the baseline model, we have a slight screwiness, since the precision is 4% higher for predicting fake articles than reliable ($P_{pos\_BM} = 89.1\%$ and $P_{neg\_BM} = 85.2\%$).

4.2 LIAR data set

We evaluate our models on the LIAR data set by combining the `train.tsv`, `test.tsv` and `valid.tsv` files (Wang, 2017). We convert the data set to a CSV file to be able to push it through our pipeline. The LIAR data set has the following labels, which we translate to the binary labels:

`Reliable`: *half-true, mostly-true, true*

`Fake`: *pants-fire, false, barely-true*

Arguably the half-true label could be either fake or reliable, but we have decided to categorize it as reliable based on the description given by PolitiFact (PolitiFact, 2022).

The performance of our models on the LIAR data set can be seen in Table 4. Again, the advanced model achieves the best F1-score of 54.4% compared to the baseline's 29.3%. In addition, the advanced model achieves an accuracy of 52.2% and the baseline achieves an accuracy of 48.1%. This is a significant reduction compared to the test data. Thus, the models generalize poorly to the LIAR dataset.

We have also tested all other models on the LIAR data, where the best were the Logistic Regression model together with a TF-IDF trigram representation. This achieves a F1-score of 62.5% (Table 4). This representation captures more contextual information than using the baseline's unigram representation, which probably is the main reason for the better performance.

The LIAR dataset has a distribution of 44% fake- and 56% reliable articles (Appendix E.3). This also means that an unintelligent model which only predicts `False` (i.e. reliable articles) would get higher accuracy than our current models. However, since no fake articles would be predicted correctly, this would give a recall and F1-score of 0.

4.3 Improvements

Were we to further improve our model, we would spend more time tuning the hyperparameters of our neural network, such as the number of layers, dropout rate, activation functions, and optimizers, which include learning rate and momentum. As the optimization process is complex, it primarily involved trial-and-error. We attempted to adjust the learning rate, loss function, and optimizer, but did not observe any improvements. We would have liked to do this tuning in a more systematic fashion, e.g. using grid search like we did with the logistic model and plotting the accuracy and loss as functions of epochs.

Due to the time it took to train our neural network, we had to both limit the total vocabulary and take only the 60 first words from each article to train on. With more computational power and time, it could be interesting to increase these numbers. Furthermore, it would have been interesting to conduct additional tests using the advanced GloVe model on the LIAR data set. Since the word embeddings in the GloVe matrix are not tailored specifically for our data set, it is possible that employing this model could have led to improved results on a distinct data set.

We would have liked to also create a Support Vector Machine model since this is a simpler machine learning model that we have seen perform well in existing fake news classifiers (Thompson et al., 2022). Unfortunately, we found that the training time was too long for our time frame. Additionally, we could have utilized different types of deep learning models, such as convolutional neural networks, which is also a popular, well-performing model among existing fake news classifiers (Thompson et al., 2022).

Lastly, we would also remove infrequent words and outliers. Both would further reduce the size of our data set. While removing infrequent words also remove words that are likely insignificant for the classification task and only cloud our models, removing outliers would likely also further improve the accuracy of our model because long articles are more complex to classify and don't necessarily match the general media landscape (Nicholson, 2018).

5 Conclusions

There are some major differences between the FNC and the LIAR data set that could explain the difference in performance. The LIAR data set is build from fact-checked statements from PolitiFact spoken in large by politicians, see Appendix E.4. Since the content consists of statements only, the content length is much smaller than the FNC - on average, the FNC articles are 25 times longer than the LIAR articles, see Figure 4 and Appendix C.1.

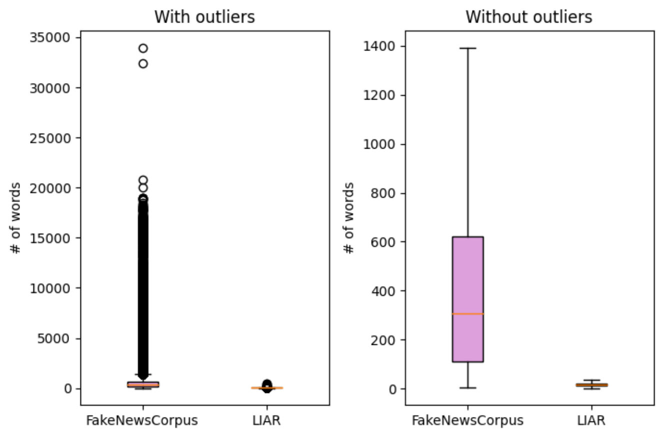


Figure 4: The number of words in the FNC and LIAR data set, respectively

The statements are individually fact-checked by humans, rather than the articles being categorized based on the domain on which it is published as the FNC does. PolitiFact is oriented towards political statements (PolitiFact, 2022), which is clear from inspecting the main subjects: The top 5 subjects in the LIAR dataset are clearly political: Health care, taxes, education, elections and immigration (Appendix E.1). In the FNC data set, from manual inspection and the domain names, it seems like there are a lot of articles about non-current events and non-political subjects. From our word frequency analysis, it is also apparent that some of the not-shared most frequent words for the FNC carry less meaning than for the LIAR set (Appendix E.2). As an example, the words "like", "one" and "time" are most frequent for FNC contrary to "presid(ent)", "job" and "vote" for the LIAR set. This suggests that the LIAR set is built from statements that carry much more condensed messages.

The labels in the LIAR data set are also very different from the labels in the FNC. While the FNC aims to classify different types of articles (clickbait, satire, etc.), the LIAR data set classifies statements on a true-fake scale (Appendix E.3).

Fake news prediction is innately a difficult task as we are dealing with natural language. In our case, the our detector would most benefit from a better or a different model, as mentioned in Section 4.3. In general, we believe that in order to improve fake news classifiers, it is necessary to have better data sets available for training that are well-classified and varied in type, source, length, platform etc. Solely the task of classifying news articles is difficult: It requires in-depth research and (political) objectivity. The quality of labeling is arguably better for the LIAR data set than the FNC as the LIAR statements are individually labeled by humans. However, the LIAR data lacks variation.

Another way to improve performance on fake news prediction is better feature extraction. It may be possible to uncover more implicit patterns in fake articles by looking at linguistic features such as passive voice, reported speech, stative verbs, lexical features such as character count and misspelled word count or behavioral features such as how a news story is shared on social media (number of retweets, shares, tags, etc.)

In our grouping of the labels in both the FNC and the LIAR data set, we found that it was difficult to translate a label as either reliable or fake in more instances. To tone with the media landscape, it may thus be necessary to train models for multi-class classification.

References

A guide on word embeddings in nlp. (n.d.). Retrieved March 31, 2023, from https://www.turing.com/kb/guide-on-word-embeddings-in-nlp#challenges-with-the-bag-of-words-and-tf-idf

Hapke, H., & Nelson, C. (2020). *Building machine learning pipelines*. O'Reilly Media, Inc.

Hu, L., Wei, S., Zhao, Z., & Wu, B. (2022). Deep learning for fake news detection: A comprehensive survey. *AI Open*, *3*, 133–155. https://doi.org/https://doi.org/10.1016/j.aiopen.2022.09.001

Nicholson, B. (2018, June 6). *Here's what you need to know about viral article length*. Retrieved March 31, 2023, from https://www.newswhip.com/2018/06/viral-article-length/

Olah, C. (2015, August 27). *Understanding lstm networks*. Retrieved March 31, 2023, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

PolitiFact. (2022). *The principles of the truth-o-meter: Politifact's methodology for independent fact-checking*. Retrieved March 28, 2023, from https://www.politifact.com/article/2018/feb/12/principles-truth-o-meter-politifacts-methodology-i/

Szpakowski, M. (2020, January 25). *Fakenewscorpus: A dataset of millions of news articles scraped from a curated list of data sources*. Retrieved March 30, 2023, from https://github.com/several27/FakeNewsCorpus

Team, D. (2019, May 1). *6 essential advantages of pandas library 8211; why python pandas are popular?* Retrieved March 30, 2023, from https://data-flair.training/blogs/advantages-of-python-pandas/

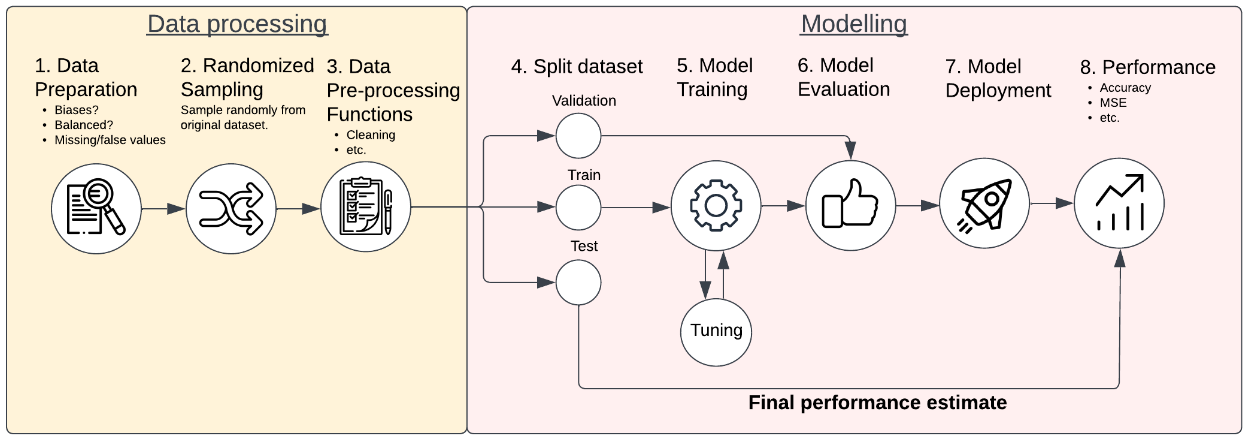
Thompson, R. C., Joseph, S., & Adeliyi, T. T. (2022). A systematic literature review and meta-analysis of studies on online fake news detection. *Information*, *13*(11). https://doi.org/10.3390/info13110527

Wang, W. Y. (2017). "liar, liar pants on fire": A new benchmark dataset for fake news detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 422–426. https://doi.org/10.18653/v1/P17-2067

Appendices

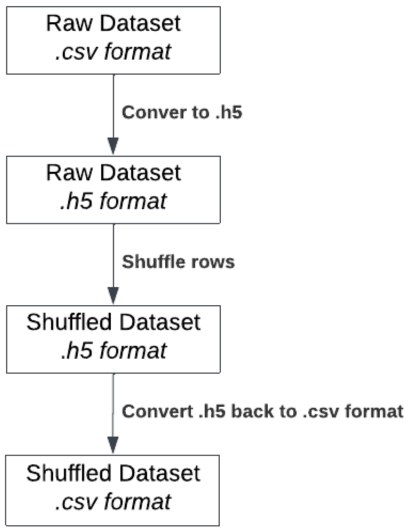
A Pipeline

Through the phases of the project, we have followed the development pipeline shown in the following Figure. The figure is inspired by Hapke and Nelson, 2020.



B Generating randomized sampled dataset

B.1 Illustration of method to generating randomized dataset



B.2 Number of rows removed

Total rows	8,528,956
Retained rows	7,273,069
Removed rows	1,255,887

B.3 Processing time of generating dataset

This is done on a regular laptop.

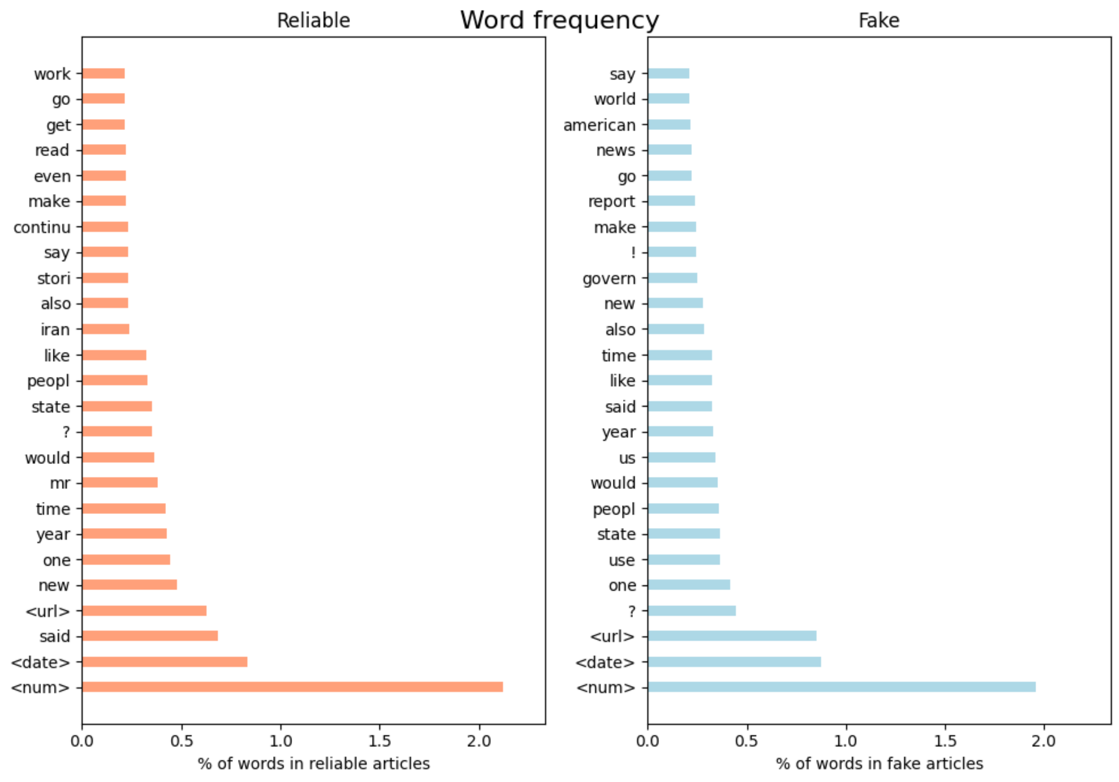
.csv to .h5	14min (all rows)
shuffle .h5	4h 5min (all rows)
.h5 to .csv	25 min (all rows)
remove unwanted rows	33 min (all rows)
clean, tokenize, remove stopwords, etc.	3h (100k rows)

C Data exploration

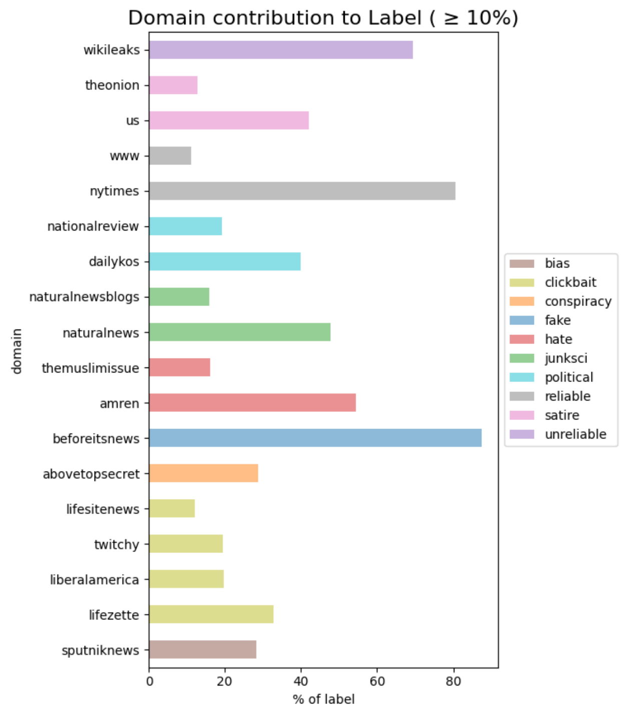
C.1 Word count without cleaning

	FakeNewsCorpus			**LIAR**		
	Total	Reliable	Fake	Total	Reliable	Fake
count	1,000,000	522,475	477,525	12,791	7,134	5,657
mean	470	512	425	18	19	17
std	660	620	699	10	11	9
min	4	4	6	2	2	2
max	33,936	33,936	18,941	467	467	309

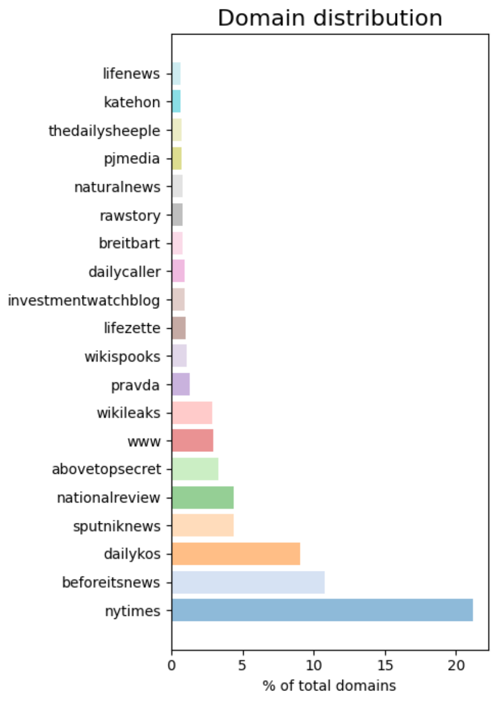
C.2 Word frequency: 25 most frequent words appearing in reliable and fake articles, respectively



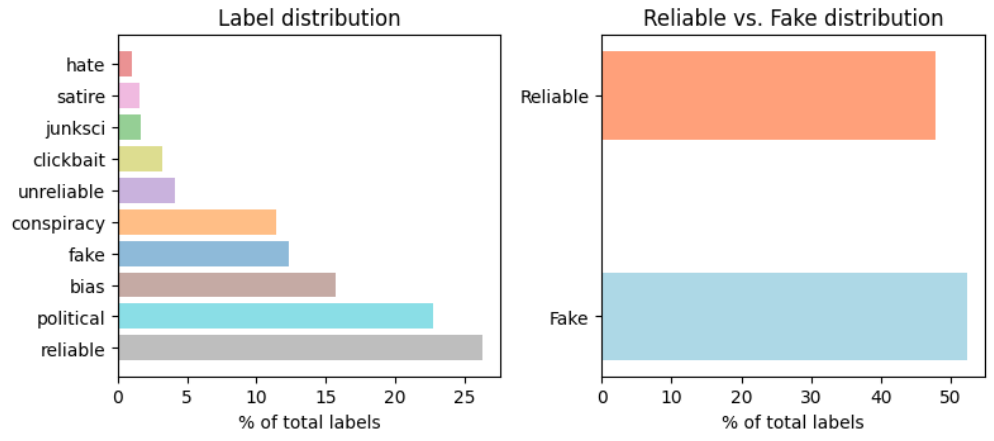
C.3 Contribution to label per domain



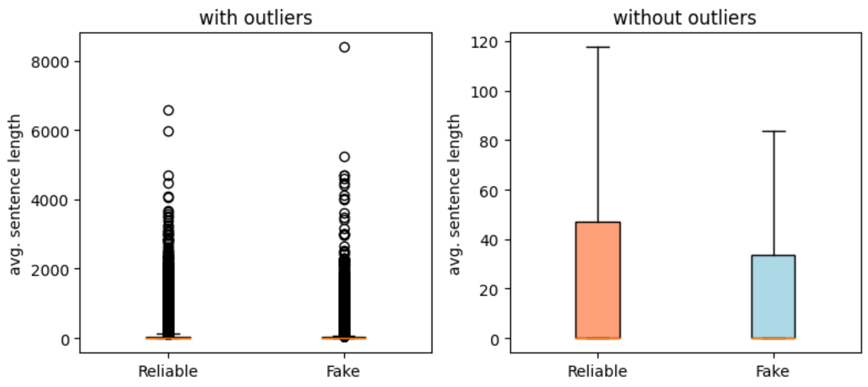
C.4 Domain distribution



C.5 The distribution of labels for all labels and binary labels, respectively



C.6 The average sentence length for reliable and fake articles, respectively



D Reproducibility: Advanced models

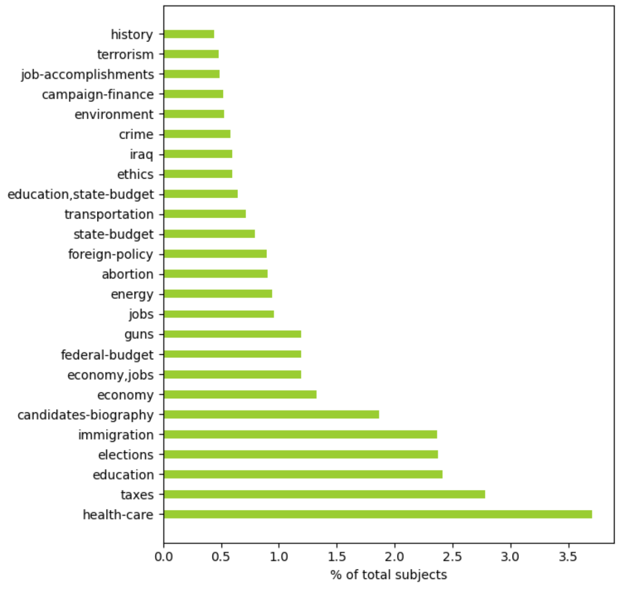
The parameters of models A-E are shown below. All the models use the `adam` optimizer and the `binary_crossentropy` loss function.

```
Model: "A"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 60, 16)            14671328

 global_average_pooling1d_1   (None, 16)               0
 (GlobalAveragePooling1D)

 dense_11 (Dense)            (None, 16)                272

 dense_12 (Dense)            (None, 1)                 17

=================================================================
Total params: 14,671,617
Trainable params: 14,671,617
Non-trainable params: 0
_________________________________________________________________
Model: "B"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 60, 16)            14671328

 lstm_3 (LSTM)               (None, 16)                2112

 dense_13 (Dense)            (None, 1)                 17

=================================================================
Total params: 14,673,457
Trainable params: 14,673,457
Non-trainable params: 0
_________________________________________________________________
Model: "C"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, 60, 16)            14671328

 dropout_6 (Dropout)         (None, 60, 16)            0

 lstm_4 (LSTM)               (None, 100)               46800

 dropout_7 (Dropout)         (None, 100)               0

 dense_14 (Dense)            (None, 64)                6464

 dropout_8 (Dropout)         (None, 64)                0

 dense_15 (Dense)            (None, 1)                 65

=================================================================
Total params: 14,724,657
Trainable params: 14,724,657
```

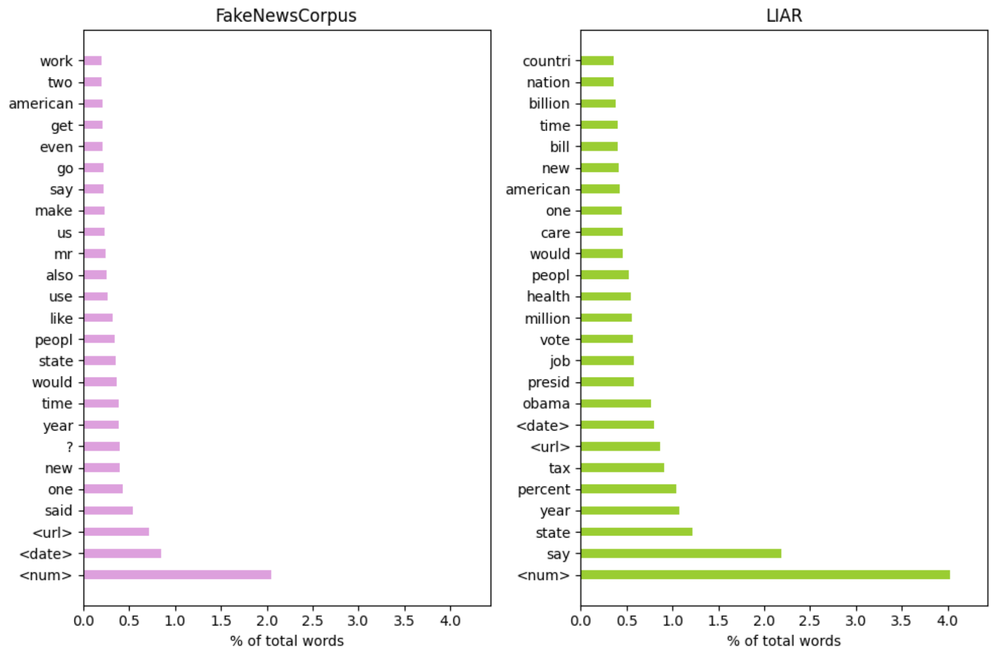
```
Non-trainable params: 0
_________________________________________________________________

Model: "E"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_9 (Embedding)     (None, 60, 16)            14671328

 conv1d_2 (Conv1D)           (None, 56, 128)           10368

 global_max_pooling1d_2 (Glo  (None, 128)              0
 balMaxPooling1D)

 dense_16 (Dense)            (None, 64)                8256

 dense_17 (Dense)            (None, 1)                 65

=================================================================
Total params: 14,690,017
Trainable params: 14,690,017
Non-trainable params: 0
_________________________________________________________________
```

E LIAR dataset

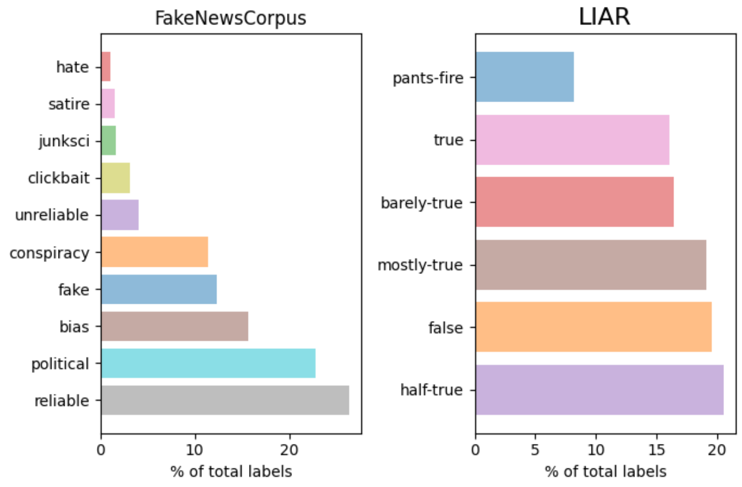
E.1 Subjects: The 25 most appearing subjects in the LIAR data set



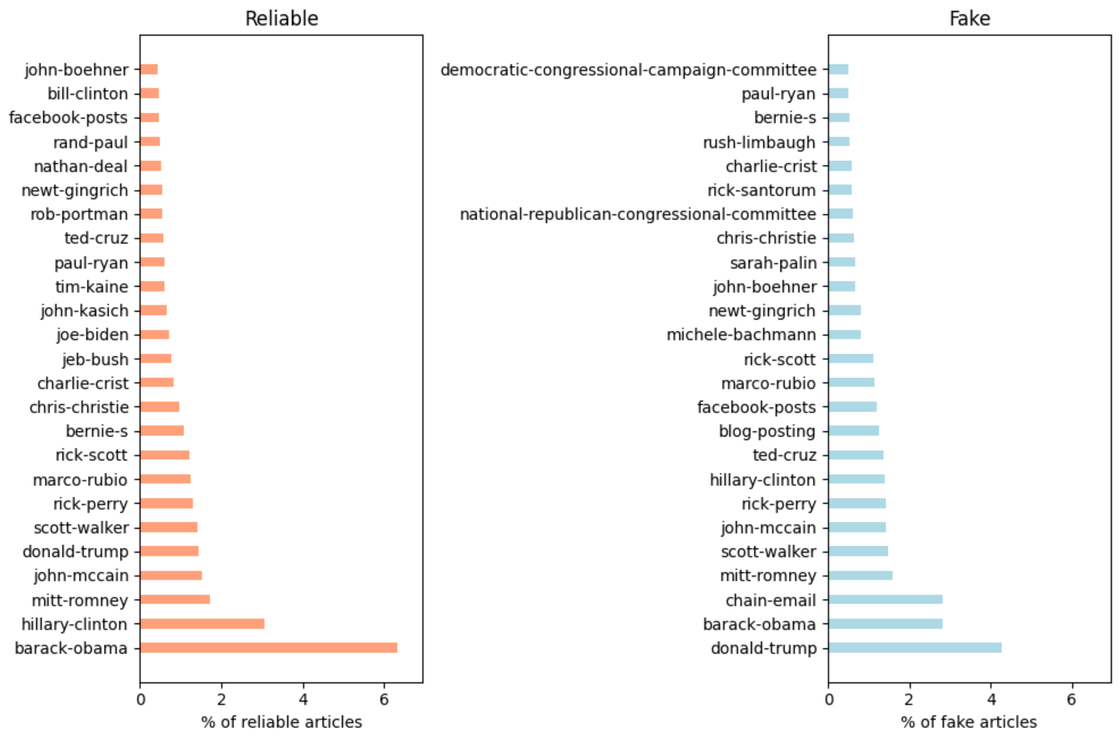
E.2 Word frequency: The 25 most frequent words in the FNC and LIAR data sets, respectively



E.3 The distribution of labels in the FNC vs. the LIAR data set



E.4 Speaker frequency: The 25 most contributing speakers to reliable and fake articles, respectively



F Logistic model all test results for different distributions

Name	Train Acc	Val Acc	precision	recall	f1	time
Logistic regression (balanced binary classes)	0.83125	0.803000	0.802239	0.825336	0.813623	383
Logistic regression (balanced types)	0.82300	0.764000	0.862319	0.666045	0.751579	231
Logistic regression (balanced fake reliable)	0.78625	0.684000	0.753589	0.596591	0.665962	346