


FAKE NEWS - EXAM PROJECT

 **Thomas Høgenhaug**
qgl656

 **Nikolaj Bruun**
srk916

 **Daniel Flamand**
xnt643

 **Sarah Roos Grabas**
lxw489

March 28, 2025

Link to our Git-Hub repository: [Fake-News-ML-007](#)

Part 1

Data Cleaning and Preprocessing

Our data cleaning process begins with the utilization of the `langdetect` library, that, as the name suggests, detects the language of the articles. This step is crucial because our early observations reveal a significant presence of Russian articles, which are irrelevant for our analysis since our model is designed to process English text, including English-specific stopwords removal. We found that slightly over 1% of the articles are in Russian, while approximately 97% are in English. These non-english articles are exclusively noise, and as discussed in ¹ we need to navigate between noise and signal and remove them. After filtering out all non-English articles, we followed the cleaning methodology we learned in Exercise 1 which includes `clean` imported from `cleantext`.

For further text processing, we used the `nlTK` library, importing `word_tokenize`, `stopwords`, and `PorterStemmer`. These tools significantly expedite our text cleaning process. However, we encountered a limitation with the date substitution function. Therefore we also implemented a custom solution using regular expressions with the `re` library.

While experimenting with the large news corpus of 995,000 rows, we had to make adjustments, because of the slow run time for such a large volume of data. Additionally, handling missing values (NaN's) in the content column required further adjustments. When executing our finished version of the cleaning function, it lead to a data reduction of approximately 61%. And after cleaning and stemming that reduction was increased to 68% of the original content column.

We use the `'matplotlib.pyplot'` module to create plots, which gives an overview of relevant information in the articles such as most used tags.

Data and the Representation

We chose to represent the `FakeNewsCorpus` using a `'pandas'` `DataFrame`. `DataFrames` are particularly suited for handling simple datasets where columns are precisely defined. They provide extensive functionality for data manipulation, including numerous built-in functions for cleaning data. The structure of a `DataFrame` offers a clear display of the dataset and allowed us to browse through the articles to identify interesting patterns.

We also used the `'pandas'` module to simplify the categorization of the articles. The `DataFrame` allows for efficient access to different articles categories, which is especially helpful when dividing the types into Fake- and Reliable news.

While we could have opted for `Polars`, a faster library, `pandas` was chosen due to its established use and familiarity within this course context.

¹Skiena, 2017, ch 9.2.1

Inherent Data Issues

Upon working with the dataset, several issues became apparent that could complicate the data analysis. Notably, some columns like 'Keywords' and 'Summary' are entirely empty, providing no information at all. Additionally, other columns such as 'Type', 'Author', 'Meta-Descriptions', and 'Tags' are only partially filled. This absence of data points could pose a challenge for extracting information. Analyzing trends and drawing reliable conclusions becomes problematic when large amounts of the data points are missing.

Observations

We made the following observations: First we discovered that there was a slight correspondence between missing author and being unreliable, fake etc. To further examine the relationship between the two variables, we conducted a correlation analysis using the `.corr()` method from the pandas library. This method defaults to the Spearman correlation, which is robust to non-linear relationships and outliers.² Our analysis focused on the variable representing the presence of an author, which was converted into a binary format (indicating whether an author was present or not), and the broad category variable. The analysis revealed a Spearman correlation coefficient of approximately 0.171³, suggesting a weak yet positive relationship between the two variables. This indicates that while there is some association, it is not strong, and further analysis might be needed to understand the dynamics fully. Another important observation we discovered was the length of the content column. Here we discovered that articles labeled 'rumor', 'bias', 'unreliable', 'satire' and fake were all significantly shorter than 'political', 'reliable'. Unfortunately the longest on average was 'junksci', 'hate', 'clickbait'.⁴

To explore the relationship between article length and the likelihood of being classified as fake news, we conducted a detailed analysis using a randomly selected subset, comprising 80 percent of the total dataset. We began by sorting the articles based on the length of their content. Subsequently, we divided the content into four normalized categories, representing quartiles of article length.

Using the matplotlib library, we visualized the proportion of articles classified as fake news within each length category. These classifications were obtained from the 'broad_category' column, which labels each article as either 'Fake News' or 'Reliable News.' The resulting graph illustrates an observable trend:⁵ the proportion of articles labeled as fake news decreases as article length increases. This suggests that shorter articles may be more likely to contain misleading or false information.

The third observation we made was during our analysis of the 'tags' column. We observed the 25 most common words in 'tags', excluding articles that had no tags. The result was not too surprising, since we already had analyzed the most common words in the 'content' column. However, it was still interesting to compare the two, as 'tags' serve as buzzwords for the articles. We found that the most frequent words in both 'tags' and 'contents' include "Trump", "Russia" and other controversial topics. This suggests that the most common words in the content are also common in the tags. This could be because their role is to attract the readers attention, and therefore are more likely to be controversial words that grabs attention.⁶

Furthermore we also displayed the 10,000 most frequent words, we can see the clear zipf-distribution⁷

Data split

We split our dataset with the `train_test_split` function from sklearn. We did a fully random split, although this was a hot topic in our group. In class (and in the textbook) it was made very clear that we should train our model on the oldest articles and test on the newest. The reason for this was to avoid leakage, preventing our model to know about events that already had occurred. However, after splitting the corpus chronologically, we made the discovery that approximately three quarters of the total amount of reliable articles were in the training set (80%). Therefore we opted for a random split, where we achieved an almost equal distribution. Our split was the suggested 80%, 10%, 10% .

²Skiena, 2017, ch 2.3.1

³See author correlation code in ReadMe

⁴See Figure 3 in the Appendix.

⁵See Figure 1 in the Appendix.

⁶See Figure 2 in the Appendix.

⁷10.000 most frequent words shown in figure 6

Part 2: Simple Logistic Regression Model

Classification of Types

We categorized each label into two groups as specified in the table below.

Labels	Fake News / Reliable News	Reason
Fake News	Fake News	Completely false or misleading content.
Rumour	Fake News	Not confirmed information.
Satire	Fake News	Intended as humor, but still spreads misinformation.
Extreme Bias	Fake News	Relies on propaganda and presents opinion as facts.
Conspiracy Theory	Fake News	Based on unfactual information claims.
State News	Fake News	News from repressive States, often propaganda.
Junk Science	Fake News	Scientifically dubious claims.
Hate News	Reliable News	Spreads hate but is not entirely false or misleading.
Clickbait	Reliable News	Generally factual but uses questionable headlines for attention.
Proceed With Caution	Reliable News	Generally credible, however may require further verification.
Political	Reliable News	Covers political topics with verifiable sources.
Credible	Reliable News	Uses traditional and ethical practices of journalism.

Table 1: Classification of News Labels

This part was almost as difficult as creating our advanced model. The conflict that arose was that news and media content exist on a much broader spectrum than simply "Fake News" or "Not Fake News". For instance, satire is rarely intended as direct news but rather as entertainment or commentary on current events. Meanwhile, categories like "Proceed with Caution" or "Clickbait" may be more likely to spread partial misinformation, as they can disguise misleading content as intended news. This categorization doesn't allow for a more nuanced view of news, either it is fake or not. This is of course not true in the real world, where there are levels to misinformation. Therefore, the lack of accuracy our models might create is solely based on this gross mislabeling ;)

Furthermore, this method is prone to error because it is done by a person, maybe as crowd sourcing, and mentioned in a lecture, this comes with big risks of false identification.

Training Process and Performance of the Logistic Regression Model

Since we had the labels to all the articles we could make supervised-learning. We designed our logistic regression using the 'content' column as x-value and the 'broad category'(fake news / reliable news) as y-value. We used 'CountVectorizer(max_features=10000)' from scikit-learn to convert our text to numerical data, using the bag of words model⁸. The max_features limits our vocabulary to the 10,000 most used words. We fitted the model on our training data and tested it on our validation set. After a period of trial and error tuning the hyperparameter, we found the model worked best with a c-value of 0.01 and a max_iter 10000. We also tried our luck with a grid search, however this exceeded the regularization time consumption of 5 minutes, and therefore we ended up not using it.

We trained the model on the training set and after trial and error on the c-value we ended up with this score for the validation set.

Performance:

Table 2: Validation set performance

	Precision	Recall	F1-score	Support
Fake News	0.84	0.88	0.86	45,501
Reliable News	0.87	0.83	0.85	44,867
Accuracy			0.86	90,368
Macro avg	0.86	0.86	0.86	90,368
Weighted avg	0.86	0.86	0.86	90,368

⁸CountVectorizer

Meta-Data Feature Consideration

Meta-data feature that might be relevant to include:

- **Author** - Many fake news outlets use anonymous authors.
- **Article type** - Relevant categorization of articles.
- **Title** - Quick overview of the article
- **Domain** - Indicates where the article was published.

The meta-data mentioned above gives an important insight into the articles. These data-points can help categorize the articles into groups, for a better overview for analysis. For instance, when we in part 1 made the observation that an article with no author were more likely to be a fake news article.

Though meta-data can be useful, there are also some columns included in the data-set that does not add any real value or insight. These include all the columns that are either partly or completely empty (Keywords, Summary, Tags), since they ramp up the run time, when they only give information about some of the articles, and not all. In general the more unnecessary data that can be excluded the better. This means that columns like Scraped-, Inserted- and Updated-at that we do not touch for reasons explained during our train, validation, test split, all can be left out.

Including the BBC Articles in the Training of the Logistic Regression Model

We applied our BBC articles to the training data, although this seemed silly as we scraped 700-something articles and added it to our 775,000 articles. This felt like urinating in the ocean by Columbia and seeing whether the water taste different in Denmark, or mathematically corresponds to 0.7‰.

Not surprisingly, we discovered this did not influence the f1-score, as we still achieved 85%. We therefore decided to exclude it for the rest of the project.

For the remainder of the project, we will limit ourselves to main-text data only (i.e. no meta-data). This makes it easier to do the cross-domain experiment in Part 4 (which does not have the same set of meta-data fields) .

Part 3: Fake News Predictor

For our advanced model we trained a Support Vector Machine. Our first model was trained using the Linear Support Vector Classification, LinearSVC.⁹ The LinearSVC model is similar to normal SVC, with a linear kernel parameter. The LinearSVC model should perform better with larger datasets, like the large corpus. This is because it is implemented with 'liblinear' instead of 'libsvm' which are two different libraries used to implement Support Vector Machines. 'liblinear' is much faster than 'libsvm' when working with large datasets and linear problems.

Before training our model, we chose to under-sample fake-news, meaning we deleted 3537 random fake-news articles. We did this knowing we had over 350.000 of each category, and with such a large dataset, we could afford to remove a small portion without losing the overall representativeness of the data.

While training our model, we wanted to do a grid search to find the best hyperparameter values for our model. However, the grid search was again very slow, and therefore we started searching for faster models. During our search for a faster model, we stumbled upon the SGDClassifier model while reading the documentation for the module scikit learn.¹⁰ The SGDClassifier is a linear model. In contrast to (batch) gradient descent, SGD approximates the true gradient of $E(w, b)$ by considering a single training example at a time.¹¹ This makes the training faster, because it makes multiple small updates instead of fewer large updates.

The SGDClassifier allows us to work with large text-data-sets like sparse matrix as TF-IDF. When using the module 'TfidfVectorizer', we can convert every text to a row of weighted numbers based on TF-IDF (Term Frequency-Inverse Document Frequency).

- **Term Frequency (TF):** How often a word appear in a given text.

⁹LinearSVC

¹⁰SVC

¹¹1.5.8.1 SGD

- Inverse Document Frequency (IDF): How rarely a word appears compared to other texts.

When we use the 'TfidfVectorizer' module to vectorize the text, we use different parameters.

- max_features=10000. This Parameter means that only the 10.000 most important words, based on their TF-IDF value, is included. This makes the model faster and more precise, since it is trained on fewer and more concise words.
- min_df=2. This parameter makes sure that the model excludes words that does not occur in at least 2 texts.¹²
- ngram_range=(1,2). This parameter makes sure that both unigrams(single words) and bigrams(pairs of words) are used to train the model.¹³ Unigrams consider individual words, and trains the model on word frequency, while bigrams train the model on pairs of words, and therefore also the context the word occurs in. For example, unigrams could misintepret "good" as positive, while it actually appears as "not good" as a bigram. So when including both, we reduce the risk of our model misinterpreting words.

To choose the best 'alpha' parameter, we did a grid search. After trial and error where we narrowed the interval down, we went with sample floats in the range from $1e-7$: $1e-6$.

The grid search was done with the hyperparameter set to:

- scoring = 'f1_weighted', helps handle class imbalance in the dataset. This is done by calculating the f1-score for each class and then weighing each f1-score by the distribution of each class.
- verbose=2, this parameter does not influence the training of the model, however it helps us gain an overview of the progress. It prints messages for each domination of hyperparameters.
- n_jobs=-1. This makes the training much faster, since it parallelizes the calculations on all cores of the working CPU.
- cv =5. This parameter is Cross validation. It splits the data-set into 5 parts. The model is then trained on 4/5 parts and tested on the last part. This process is repeated 5 times, using a different part for testing each time, and taking the average performance of the 5 tests. This average is then compared for each alpha value in our grid search. A real advantage of cross validation is that it yields a standard deviation of performance, not only a mean.¹⁴

Hyperparameters for the SGDCClassifier:

- loss = 'hinge', this gives us a linear SVM, which is what we want as basis for our model.¹⁵
- alpha = 1e-6. 'alpha' is a regularization parameter that controls the strength of the penalty parameter. A lower alpha makes the learning rate larger and allows faster updates.¹⁶ We chose this float based on our grid search result.
- max_iter = 10.000. Max_iter is the maximum number of passes over the training data.¹⁷ We chose 10.000 to make the training faster.
- n_jobs = -1, means that we use all CPUs.
- penalty = 'l1'. When the penalty parameter is set to 'l1' it resets some weights to 0, so it eliminates the least important features. This makes the model more simple. It is relevant for us to choose the 'l1' as penalty since we are training on a set with many features, and 'l1' can eliminate the least relevant ones, this is not achievable with 'l2'

We tested the model on our validation set during the process of training the model. Therefore, we did some fine tuning, which is based on how our model performed on the validation set. We ended up with the hyper parameters above.

¹²TfidfVectorizer

¹³TfidfVectorizer

¹⁴Skiena, 2017, page. 227

¹⁵SGDCClassifier

¹⁶SGDCClassifier

¹⁷SGDCClassifier

Part 4: Evaluation of Simple and Advanced Models

It is important to stress that up until this point, we have not touched the test-set (other than cleaning). When testing the model on the test-set, on the first logistic regression, we found that the weighted average f1-score was the same (only difference in insignificant decimals) as the validation set - 85%.

Afterwards we evaluated the test-set on the advanced model. Here, we discovered a very similar, although slightly worse f1-score: 88%. This could be due to overfitting as we played around changing hyperparameters to get the highest f1-score on the validation-set. Even though we tried to avoid this problem by creating a 5-fold cross validation. We were eager to achieve a f1-score above 90%, however this was not possible for us.

Before we were able to test on the liar dataset, we first had to convert the file to a CSV. We also created headers with fitting names for each column, e.g. "Statement". Once this was done, we could easily implement the liar_train_set as the test-set for our model (this set has the most data). With our testing, we achieved a weighted average of 55%, and a very interesting confusion matrix.¹⁸

We could now plug the train_set from liar, and evaluate it on our previously trained logistic regression (we used train_set as this was the file with most lines). We ended up with a weighted average of 55% but what was more interesting was the confusion matrix.¹⁹ The matrix shows that this model predicts way more statements from the LIAR data-set to be Fake rather than to be Reliable, also why it has a larger percentage of the 'fake' articles correct.

When testing the liar dataset on the more advanced model, we scored an overall worse weighted average f1-score of 51%, but the overall performance, as seen in the confusion matrix, is more equally distributed.

We have compared the results in the table below.

	Fake News Corpus	Liar Dataset
Logistic Regression	85%	55%
Support Vector Machine	88%	51%

Table 3: F1-score weighted average

Part 5: Conclusion

We scored significantly worse on the liar dataset in comparison to the test set from the full news corpus. Upon greater observation and investigation, we found that the liar dataset was mostly interviews, contrary to the Fake News Corpus, which is from actual news articles.

The difference in written language versus spoken language is most likely going to be substantially different. Therefore when our model is trained on written-language it is no surprise that the model performed worse on spoken-language data-sets.

To explore the discrepancy between the performance on our large test set and on the LIAR set, we further explored the semantics and statistics of both datasets.

A `simple.describe()` analysis showed the datasets composition where completely different. The mean length of each statement (liar-set) was 10 words, where the mean length was 261 words on the full test set.

Next step in our investigation consisted of finding the most common bigrams. Here we found more of a similarity, where the datasets yielded some of the same 'most common' bigrams, for instance ('unit', 'state') (this was done after cleaning and stemming). Though similar, there were still differences. the LIAR set had a lot of bigrams consisting of names of famous politicians and places, while the train-set had a lot of non-text bigrams, for instance ('contin', 'read') and ('advertis', 'contin'). These differences exaggerate the discrepancy between interviews and news articles.

As mention above, we trained out SVM using both unigrams and bigrams, and seeing this difference in bigrams is comforting for the fact our model performed poorly in part 4.

When evaluating our SVM on the LIAR dataset, our model showed a tendency to label more statements as reliable news than as fake news. It predicted 5,778 articles as reliable and only 4,462 as fake. It's important to note that only

¹⁸Confusion matrix for test set 5

¹⁹Confusion matrix for logistic regression - Liar6

3,638 out of the 10,240 statements in the LIAR dataset were actually labeled as reliable.

It's difficult to determine the exact reason why spoken-style statements are more likely to be labeled as reliable by our model. One likely explanation lies in the well-known saying, "garbage in, garbage out." Since our model was trained exclusively on written articles, applying it to spoken-language content introduces a mismatch that naturally leads to misclassifications.

We also found the labeling interesting enough to evaluate further. We examined which types of articles it was bad at classifying, and found that hate was wrongly classified a whole 27%²⁰ of the time. This makes sense as we labeled 'hate' as 'Reliable News', although it is clearly somewhere inbetween. Similarly, the second ("clickbait" - reliable), third ("satire" - as fake) and fourth ("political" - as reliable) worst performing labels all can be put somewhere between fake and reliable. Again, this binary division of news articles hugely simplifies the spectrum that news content actually exists on. Furthermore the types we found easy to classify, our model also found easy to classify, for instance only 5% of the reliable news was classified wrong.

The large news corpus were only scraped on written news articles. This introduces a clear sampling bias, as our model is trained solely on written language. One way to improve the model would be to improve the dataset itself. Ideally, the dataset should include a large and balanced amount of both written and spoken language. Training on both would likely help improve the models performance across different types of news content.

Moreover is the question about labeling still relevant, who labeled them, and why did they choose the label they did. The reference-link in the read-me is no longer active, hence not a lot of information is provided about the dataset. How are we going to label 'half-true' as this is exactly between true and false. We could have chosen a different labeling, which would probably have yielded another result.

Our model could be optimized for spoken language if it actually was trained on it, although a more advanced model would probably be needed for this. In this project we followed Occam Razors philosophy of simplicity.

In conclusion, while our model performed well in written news detection, it had limits when applied on spoken content. This underline the importance of diverse training data, and a more nuanced labeling.

.

²⁰Obtained in SVM.ipynb

Appendix

Additional Figures

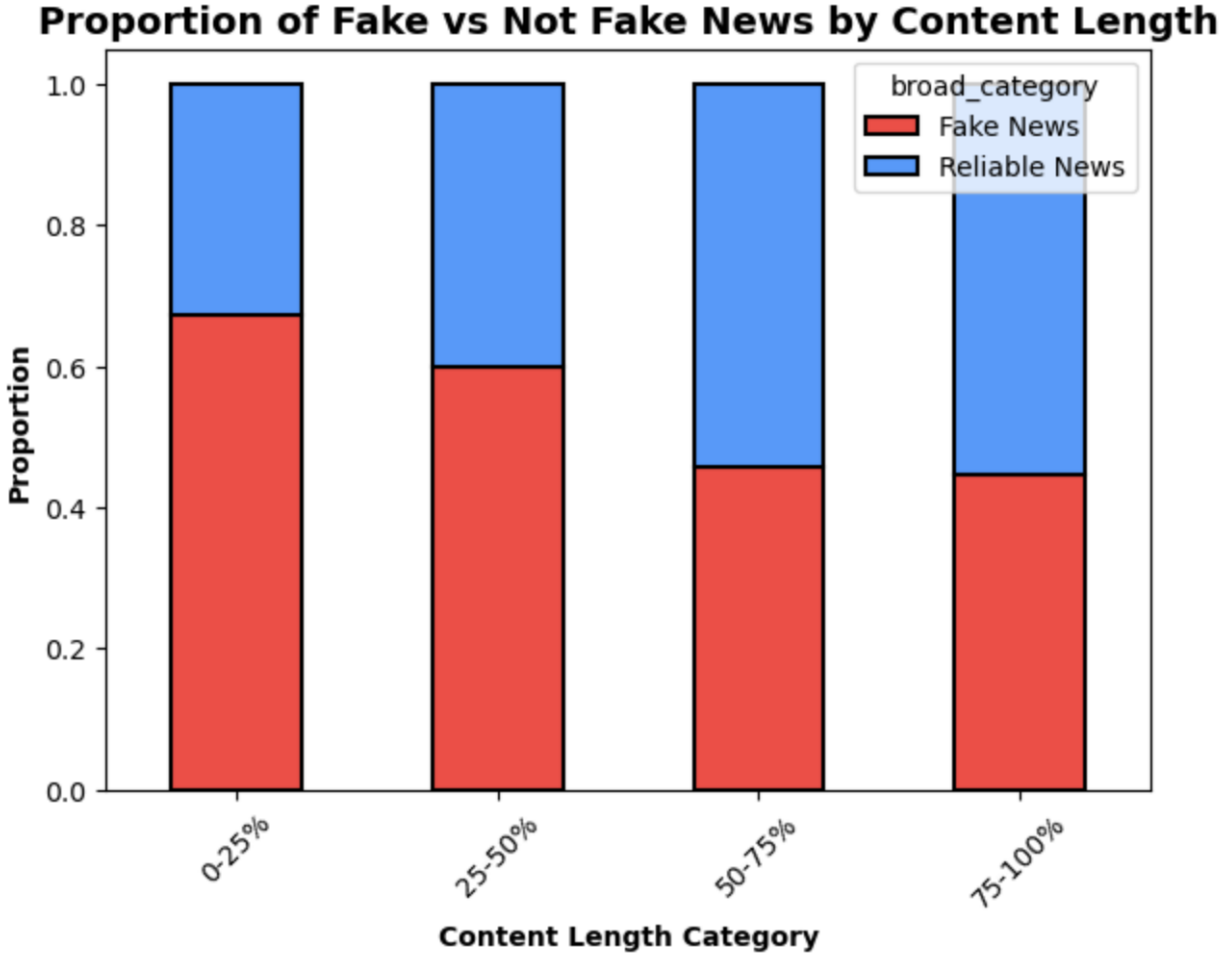


Figure 1: This chart illustrates the correlation between the length of articles and their classification under the broad category label. The data has been meticulously organized by article length and segmented into four equal quartiles, as indicated by the respective percentage labels. This visualization represents 10 % of the larger article corpus.

	Predicted: Fake	Predicted: Reliable	Total (Actual)
Actual: Fake	2989 (TN)	3613 (FP)	6602
Actual: Reliable	1473 (FN)	2165 (TP)	3638

Table 4: Confusion matrix liarset on SVM showing classification results of fake vs. reliable news.

	Predicted: Fake	Predicted: Reliable	Total (Actual)
Actual: Fake	40050 (TN)	5615 (FP)	45665
Actual: Reliable	5372 (FN)	39331 (TP)	44703

Table 5: Confusion matrix test-set on SVM showing classification results of fake vs. reliable news.

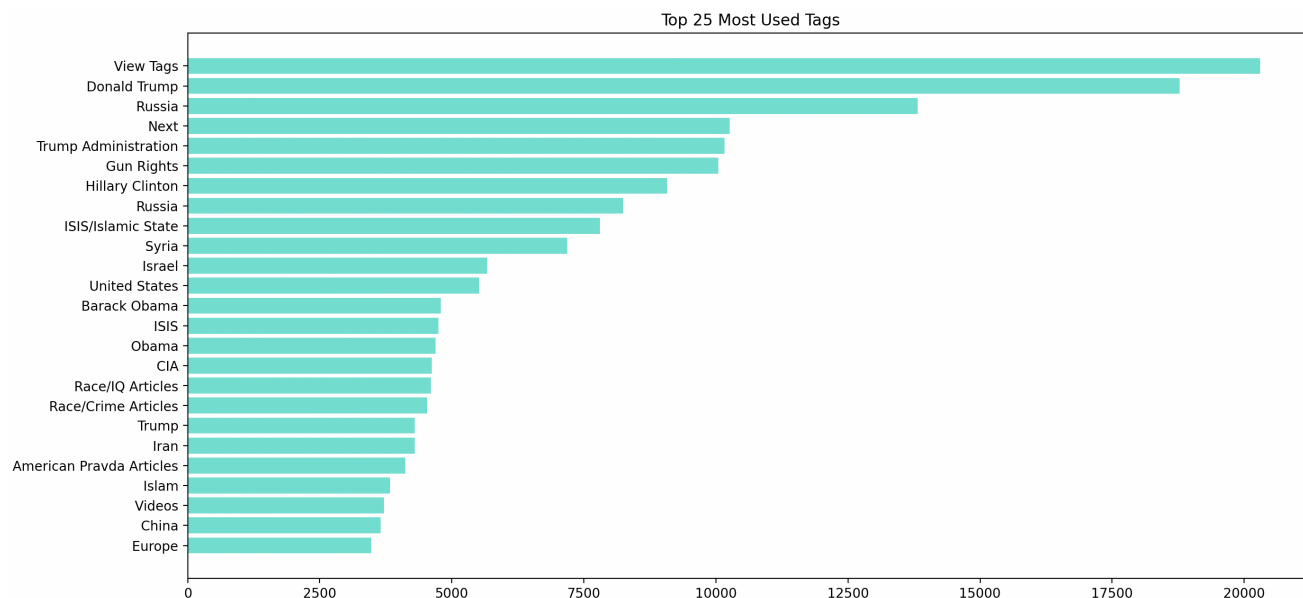


Figure 2: Top 25 most used 'Tags'

	Predicted: Fake	Predicted: Reliable	Total (Actual)
Actual: Fake	5429 (TN)	1173 (FP)	6602
Actual: Reliable	2946 (FN)	692 (TP)	3638

Table 6: Confusion matrix liar-set on logistic regression showing classification results of fake vs. reliable news.

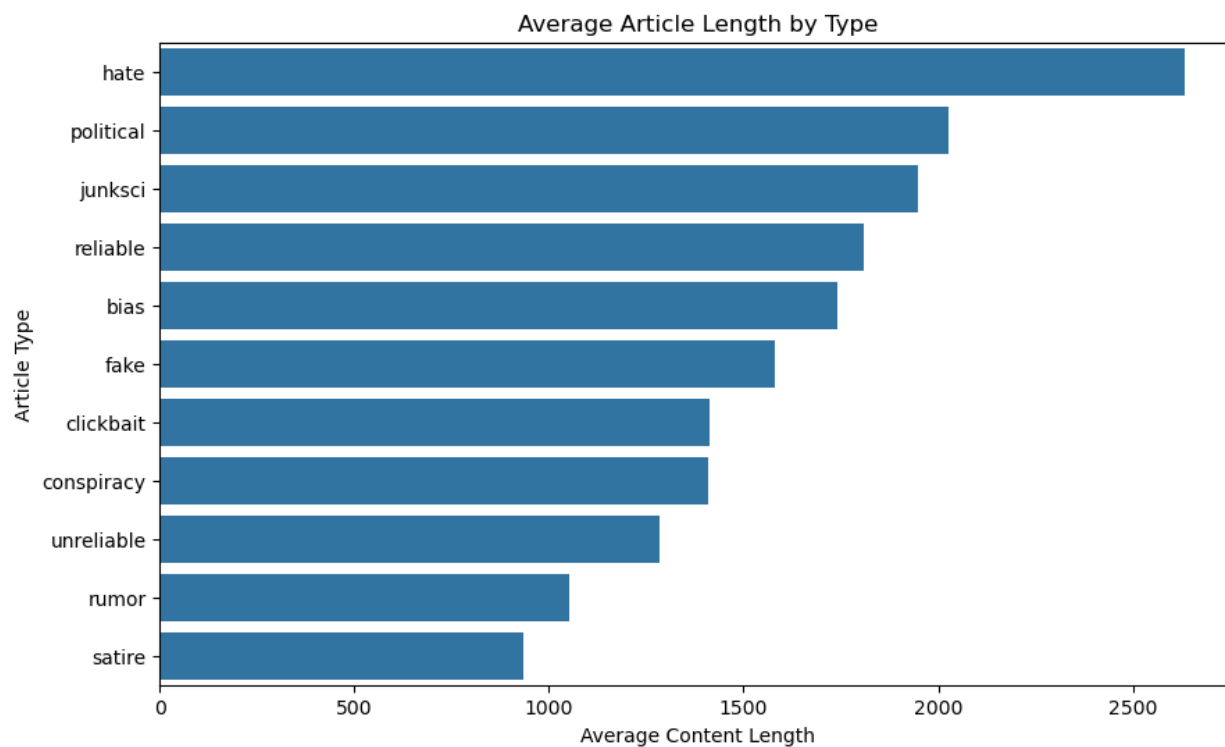


Figure 3: Average string length for every type.

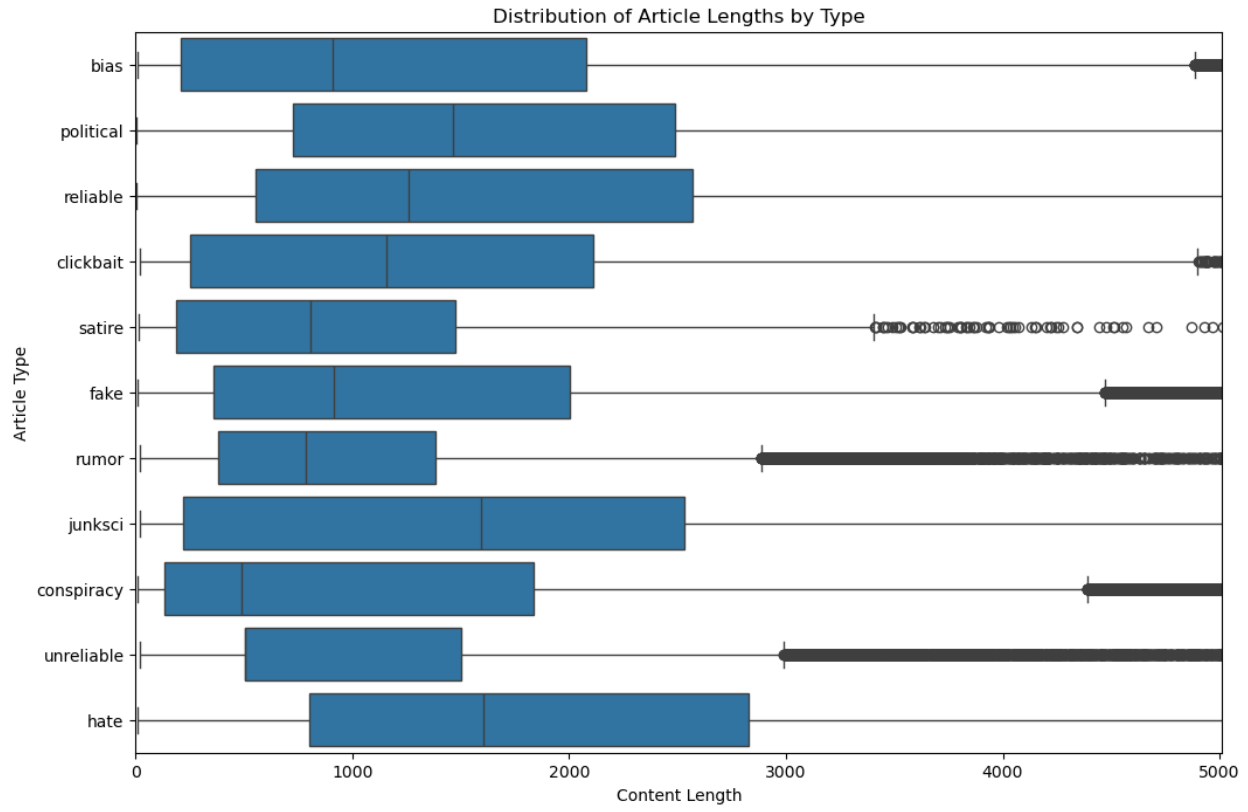


Figure 4: Distribution of Article Lengths by Type

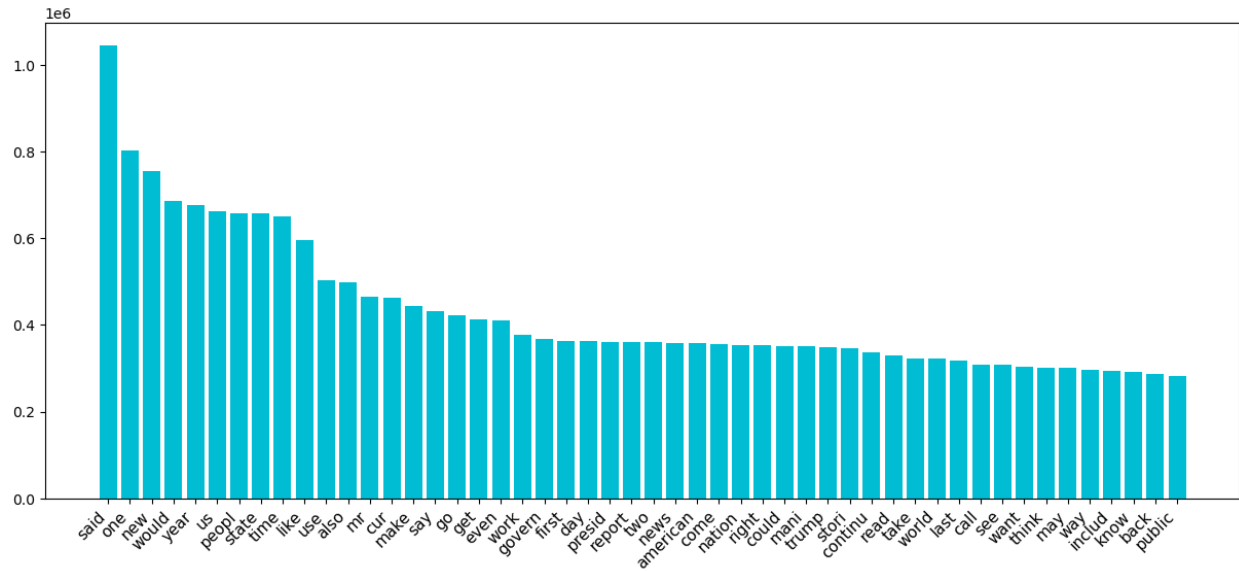


Figure 5: 50 Most common words

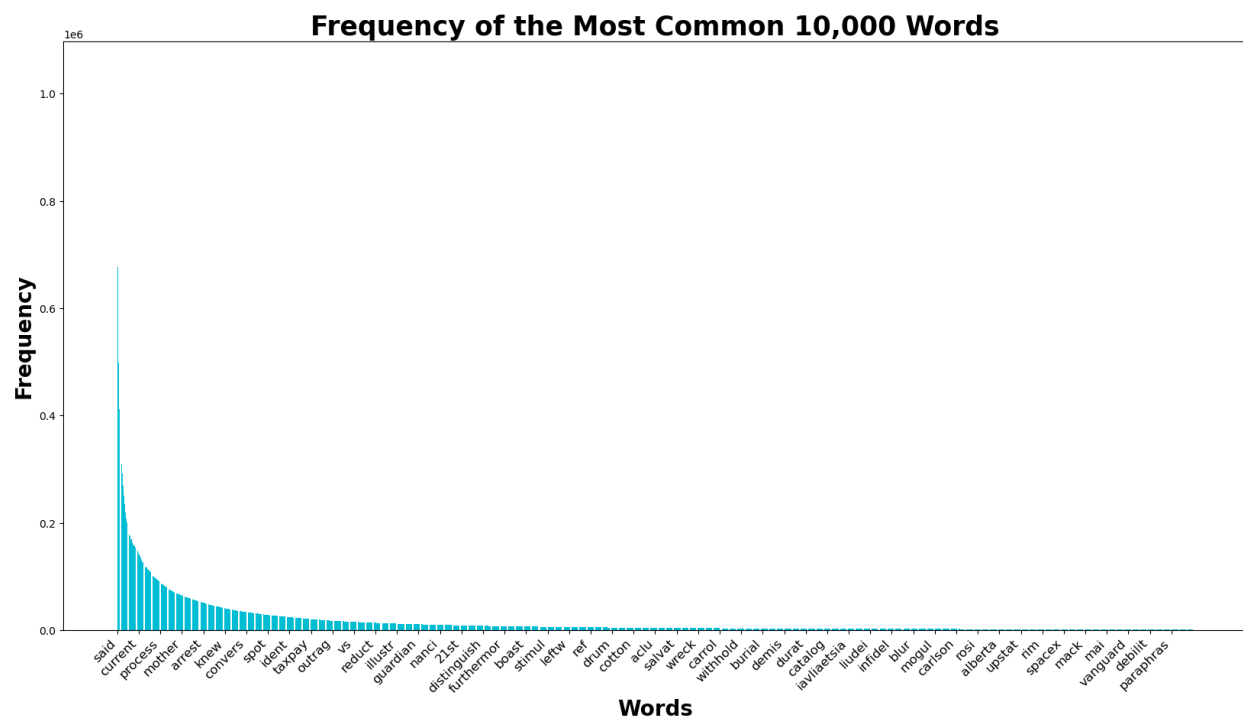


Figure 6: Frequency of the Most Common 10,000 Words