Fake News Detection: Building a System to Identify Unreliable News Articles

Machine Learning Project | Final Report

Team: Jackie Glasheen, Harsh Vardhan Pachisia, Ridhi Purohit, Shwetha Srinivasan

May 2023

Overview

“If we are not serious about facts and what’s true and what’s not, if we can’t discriminate between serious arguments and propaganda, then we have problems”. Former President Barack Obama stated this over 6 years ago, and with the percolation of AI and the trend of social media amplifying unreliable information, the issue of fake news affecting public discourse will only get worse. It is critical that action is taken to build systems that can identify and verify false or misleading information that is disseminated through various media channels, including social media, websites, and news outlets. In this project, we attempt to build a system to identify news articles that might be unreliable. The motivation behind this project is to provide a tool that helps stem the flow of misinformation. Skewed facts can influence public opinion, trigger social unrest and sway political outcomes as we have seen worldwide over the past decade.

For this task, we utilize a dataset of 20,166 news articles hosted on Kaggle from different authors and sources. We had access to the text, title, and author of each article with 16,608 forming a training set and had been labeled as reliable/unreliable with the rest forming a test set and remaining unlabelled. We first cleaned the dataset (text, author, title) and then conducted pre-processing on the data such as removing stop words, stemming, and lemmatizing words. Based on our exploratory data analysis, we tuned various supervised and unsupervised machine learning models to ascertain which ones provide us with the best overall performance given the equal importance of false positives and negatives in fake news prediction. All our code, processed data, and final report can be found here (need a UChicago account to access).

In general, we find that supervised models tended to outperform the unsupervised models providing greater accuracy as well as easier interpretability for users. While we got consistently high test accuracy (>90%) across most of our models, when we dug deeper into our features, we find concerns about inbuild biases in our training data, which give us pause as to whether fake news prediction can be a task simply done by machines due to inherent question of defining what the ‘truth’ is. What may seem reliable to one person may be unreliable to another due to their worldview. Mike Devito, a researcher at Northwestern puts it best, “These are not solely technical problems; these are human problems that technology has simply helped scale yet we keep attempting purely technological solutions. We can’t simply machine-learn our way out of this disaster, which is a perfect storm of poor civics knowledge and poor information literacy.” Hence, while we do get good results from our models, we do believe that the way our training data has been labeled will play a major role in how our models would work in the real world, potentially skewing their results. Ultimately, this leads us to conclude that the task of fake news prediction cannot be solely done via machine learning, and needs a more robust, holistic, human-centric approach to understanding what the ‘truth’ itself is to determine reliability.

Data and Task Setup

Data Investigation

We explored the data and prepared summary statistics and charts that highlight key features of the data. One, the distribution of reliable/ unreliable article labels. We find 10413 are marked unreliable, and 10387 are marked reliable (balanced dataset). Second, we find the average number of utterances in an article is approximately 35. We find differences in average utterances for unreliable and reliable articles: unreliable news has fewer utterances on average and a larger range of utterances. We find that the average article length is 768 words and the average title length is 12. Third, we found the distribution of words per article is different for reliable and unreliable articles: unreliable articles more often have very few words per article relative to reliable articles. Reliable articles have a relatively balanced distribution of words per article, while unreliable articles have a more skewed distribution. Fourth, the number of articles written per author, and trends around authors and their tendency to either write unreliable or reliable articles. We find many authors have written many articles, and authors tend to write either exclusively reliable or exclusively unreliable articles. Fifth, trends around missing data (null values). We find articles with null authors are largely marked unreliable articles. Finally, in investigating the data, we noticed various data anomalies including non-English language usage, missing text for articles, noise in the text (i.e., titles within the article text column), and noise in the author and title columns (i.e., titles appearing in the author column).

Data Preprocessing

Understanding the data provided for detecting fake news presented challenges in terms of the varied ways people write and present their thoughts. We had to make certain assumptions while parsing textual content and had to rely on libraries to do the heavy lifting in terms of text normalization. For data cleaning, we performed the following. First, we removed rows where data was present in a language other than English because our model will be trained based on the English language so those would be irrelevant data points. Second, we removed rows where the column ‘text’ had null values or no content because the major data point to be fed into the NLP model would rely on the text of the article to predict misinformation. In this scenario, having information on just the author or title or both without information on the article will not be a value add to the model’s prediction ability. For ‘author’ and ‘title’, if there were null values present but there was content in the ‘text’ column, then the nulls were replaced with the word ‘unknown’. This was done to protect the information from the articles we would majorly rely on while predicting reliability. Third, it was observed that in some rows, the content in ‘title’ column was also present verbatim in the ‘text’ column, so the repeated text in ‘text’ column was removed resulting in less redundancy and reduced ambiguity. Finally, we removed noise from the data such as urls, punctuation marks, white space and converted digits to words.

For text normalization: first, we tokenized the words so that it would be easier to process for normalization. Second, we removed common stop words in the English language so that we only retain the necessary information. Third, we used stemming to reduce the words to their root forms by removing prefixes and suffixes. This would be helpful for text matching at a later stage while also reducing the number of unique words present in the data. Fourth, we performed lemmatization to further remove redundancies in the text by reducing the words to their dictionary form. This will help improve the accuracy of the NLP algorithms and reduce the dimensionality of the textual data. Finally, we check the impact of data cleaning and text normalization by figuring out the total and unique word counts before and after preprocessing.

As part of the EDA process, we conducted sentiment analysis. Sentiment analysis can be a useful tool for fake news detection, as fake news articles often contain language that is emotionally charged or misleading in order to manipulate readers' emotions. We considered two measures of sentiment analysis: polarity and subjectivity as we thought these would be most relevant for fake news detection tasks (Appendix 3). Based on an initial analysis, polarity scores don't seem to correlate with news being reliable or unreliable. Subjectivity scores, however, are more spread out for unreliable news, indicating that more unreliable articles are subjective and unreliable news is more subjective than reliable news.

Model Findings

Supervised Models

Predicting Unreliable News with only Author

In investigating the data, we discovered that there are many prolific authors (i.e., they write many articles), and authors tend to consistently write reliable articles, or consistently write unreliable articles. In the chart, each point represents an author, and we plotted the number of articles written by a particular author by the proportion to articles written that are unreliable. We find that the vast majority of authors either write exclusively reliable or unreliable articles.

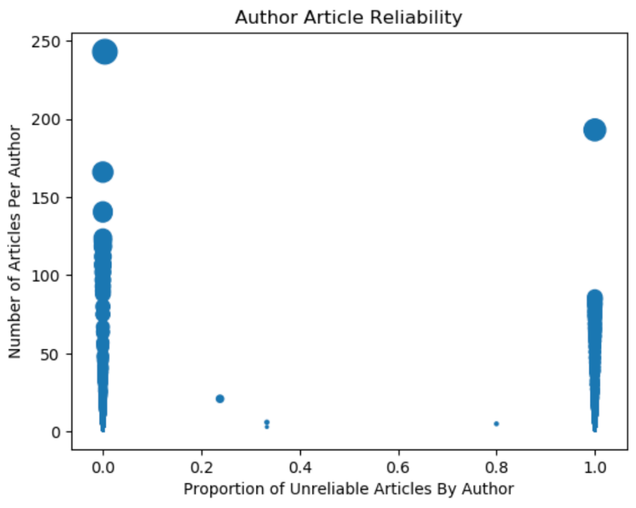


Figure 1: Author Article Reliability

Based on this finding, we investigate the power of authorship alone in predicting fake news. We first investigated the power of authors alone in predicting fake news using a simple author "lookup table." Using the training data, we determine the percent of each author's articles that are unreliable and accordingly make a reliable/unreliable prediction. Then we match each author in the testing data up to the relevant author prediction.[1] This simple model produces ~92% testing accuracy. Next, we implemented a simple logistic regression model using dummy variables for each author. We chose to run a logistic regression model so that we could predict the probability of each text being reliable or unreliable given the set of dummy variables indicating author. This model produces ~ 93% testing accuracy. In both cases, the testing accuracy errors were largely due to authors being present only in the testing data. The high accuracy of the author-only models highlight the fact that authorship is an extremely powerful predictor

[1] For any author who writes >= 50% unreliable articles, we predict they will always write unreliable articles. When the author only existed in the testing data, we predict they will write unreliable articles.

of fake news. However, we are aware that this type of model only works well if the authors in the test data exist in the training data, and thus may not be all that useful in the "real world" where training data may be more limited in authorship than the test data.

Feature Creation

After preprocessing the data (stemming and lemmatized), we decided to use both the title and text of an article together to provide a more comprehensive representation of the content and context of the article. Combining them provides enhanced information as well as contextual clues with the title conveying the main idea with the text giving more detailed information. Moreover, the title and text should contain complementary information that helps the model make more accurate predictions.

Based on the data selected, we first created certain text-based features. These include word count, sentence count, and average word length to predict unreliable news. The length of an article can provide some indication of its depth and level of detail. Longer articles may be perceived as more comprehensive and well-researched, potentially indicating higher reliability. In a similar vein, the number of sentences could reflect the complexity and structure of the article. Articles with more sentences indicate more elaborate arguments or thorough explanations, suggesting a higher level of reliability. The average length of words can provide insights into the complexity of the language used in the article. Articles with longer words may be more technical or academic in nature, which can imply a higher degree of reliability. Next, we also performed count vectorization and tf-idf vectorization of the data to serve as features. Count vectorization captures the importance of a word in a specific document. It calculates the product of the term frequency (count) of a word and its inverse document frequency (idf). Tf-idf normalizes the term frequency by the document length to account for variations in document sizes. Tf-idf assigns higher weights to words that are both frequent in a document and rare in the entire corpus, capturing their relative importance. It helps in distinguishing words that are highly specific to a document from those that are commonly used across the corpus. By considering the overall corpus statistics, tf-idf can help identify words or phrases that carry discriminatory power for classifying reliable or unreliable news.

Logistic regression

We decided to use logistic regression since it is commonly used for binary classification tasks, where the goal is to predict the probability of an instance belonging to a certain class (in this case, reliable or unreliable news). First, we ran a logistic regression model using only text-based features, achieving a 68.8% test accuracy, with a high level of false positives and false negatives. This is a problem since for fake news, we need to address both scenarios and aim for a balanced approach that minimizes both false positives (classifying true news as fake) and false negatives (classifying fake news as true). If true news is misclassified as fake news, it can lead to a lack of trust in reputable news sources and the spread of misinformation. If fake news is misclassified as true news, it can lead to the spread of harmful and dangerous information that can have serious consequences for individuals and society as a whole.

As a result, we moved on to using the other two features, count and tf-idf vectorization for our logistic regression model. We wanted to see what the differences would be between having unigrams vs. bigrams. We choose to ngram range of (1,2) since choosing unigrams and bigrams for tasks such as text classification was recommended in the literature we read since choosing a higher gram size would cause the number of possible n-grams to grow exponentially, which can lead to sparsity issues and overfitting. Hence, we ran the models using both of them for logistic regression to tune our parameters. When we run a logistic regression model with either 1gram or bigram, both of them seem to have similar high accuracy, between 95-96% test accuracy. To better understand such high accuracy, we decided to dig deeper into which features were the most relevant to the prediction. Between unigram and bigrams, the predictors do completely change, with a less clear pattern in unigram as compared to bigrams. For bigrams, the weights of unreliable news seem to be more far-right leaning (with high weights for 'queer f**k', 'obamacar quit', 'huge trump') (Figure 2) with the ones for reliable news not having much of a pattern (eg: 'irregular nune' has the highest weight).

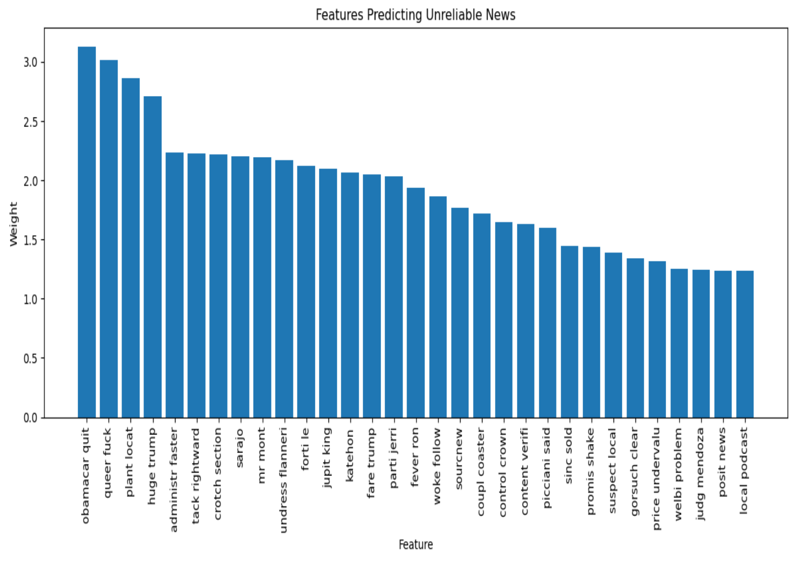


Figure 2: Top 100 features predicting unreliable news using a count vectorizer with bigrams

We were also worried that there could be duplicates, i.e. the same articles both in the training and test data causing such a high accuracy score. While we checked this and found no duplicates, we did find 274 articles that had an over 90% cosine similarity between training and test articles. However, we decided to keep them since while the words in similar articles were the same, the order in which they were presented was different (i.e., were unique articles). We also ran k-fold cross-validation 10 times, getting a mean accuracy score of 94.7% using the count vectorizer. The scores were also consistent throughout each fold, leading us to believe that the model is working well across the folds and that our accuracy is not based simply on a specific train-test split. Next, to prevent further overfitting, we ran regularized logistic regression models adding a penalty term to the model's training process and encouraging it to use only the most relevant features, reducing the chances of overfitting, and reducing our accuracy to around 94%. In general, the logistic regression models with different regularization strengths consistently show high accuracy, precision, recall, and F1 scores across different data representations (count-lemmatized and tf-idf-lemmatized). These models are reliable and perform well in distinguishing between real and fake news. However, we are concerned with potential overfitting, where the model may have learned specific details of the training data but struggles to generalize well to unseen data.

Passive Aggressive Classifier

We used a passive-aggressive classifier next since it is good at adapting quickly and is efficient. Its purpose is to make updates that correct the loss, causing very little change in the norm of the weight vector. The updates are designed to minimize the loss while causing very little change in the norm of the weight vector, which can help prevent overfitting and improve the generalization performance of the model. It is useful when the number of examples is large, as is the case in this dataset when predicting fake news.

Similar to logistic regression, we used the processed title and text for our input data and used the count vectorizer and tf-idf vectorizer as features for our model. We tuned the hyperparameter for the model, max iterations, by trying out different values of it and plotted the corresponding performance metrics of each iteration in Appendix 1. We can see that the optimum maximum iteration is around 7 where it attains the highest accuracy on test data and has similar values on precision, recall, and F1. The PassiveAggressiveClassifier max_iter=7 achieves reasonably good performance, although slightly lower compared to logistic regression. We decided upon our choice of max_iter after experimenting with different values. These models achieve high accuracy on the training data and decent accuracy on the test data. The precision, recall, and F1 scores are generally high, indicating reliable performance in detecting fake news.

Naive Bayes Classifier

Naive Bayes is a popular machine learning algorithm for detecting fake news because of its simplicity, efficiency, and effectiveness. The algorithm works well in situations where the data has a large number of features and the features are independent of each other, as is the case in natural language processing tasks like fake news detection. The algorithm can effectively model the many different factors that might be indicative of fake news, such as the language used in the article, the sources cited, and the overall sentiment. As Naive Bayes is a probabilistic algorithm, it assigns probabilities to different outcomes. This allows the algorithm to not only detect whether an article is fake, but also to quantify how likely it is to be fake. We have used multinomial Naive Bayes as it performs well on text classification tasks, can handle missing data well and is robust to irrelevant features.

We implemented a Multinomial Naive Bayes model as literature and research indicated that this model would be best for tasks that use discrete features. As the count vectorizer and tf-idf are discrete features, the multinomial model was implemented. This model had a 92% accuracy and 91% F1 with count vectorizer as a feature and 99% accuracy and 80% F1 score with tf-idf as a feature. The difference in scores between the two can be for a variety of reasons such as the information content or the characteristics of the corpus. For example, TF-IDF can emphasize features with a higher discriminative power and can lead to different classification compared to CountVectorizer. Furthermore, if the dataset contains many repetitive or non-discriminatory words that occur frequently across different classes, CountVectorizer may assign higher weights to these less informative features. In such cases, TF-IDF, by considering the inverse document frequency, could better downweight those common words, thus classifying it differently. We extracted the top features for both reliable and unreliable news for the CountVectorizer model (Appendix 6). We continue to see similar trends where certain words associated with right-leaning ideologies such as ‘jockey ayatollah’ and ‘woke follow’ are important features for predicting unreliable news. At the same time, certain n-grams are repeated as top features in both labels such as ‘mann sir’ and ‘vaccin learn’. This could be because this model assumes independence among features but it is possible and likely that it isn’t the case and that the model can’t capture more complex relationships and correlations between features.

Support Vector Machines

Support Vector Machines (SVMs) work by finding the best hyperplane that separates the training data into different classes, with the goal of maximizing the margin between the two classes. SVMs have been used for fake news detection with some success, especially when the classification problem is highly nonlinear and the data has a large number of features. SVMs can be particularly useful when dealing with imbalanced datasets, where there are many more examples of one class than the other, as SVMs can often achieve good performance even with limited data from the minority class. SVMs can be sensitive to the choice of hyperparameters, such as the regularization parameter and kernel type, which can make them more difficult to tune and optimize than some other classification algorithms. We have implemented a linear SVM as data on fake news prediction is linearly separable and has low computational cost making it a simple, interpretable model for fake news detection.

We trained our SVM model on other kernels such as polynomial and radial basis function to check our assumptions about the linear separability of the dataset. The Linear SVM model was the best performing one in terms of accuracy and F1 Score.

Random Forest Classifier

Random forests are a type of ensemble learning method that combine multiple decision trees to make a final prediction. Each decision tree in the random forest is built independently on a subset of the training data and a subset of the features, and the final prediction is made by combining the predictions of all the individual trees. There are some important features of random forests that make it a suitable model for fake news prediction. Random forests are less prone to overfitting and as a result have better accuracy. They can also provide insights on which features are important for classification. We extracted the most important features for classification for our TF-IDF random forest model (Appendix 2) and the results show there is some overlap between the important features extracted for the logistic regression and naive bayes model. Furthermore, the importance of each single feature is extremely low which is to be expected considering the large corpus of words.

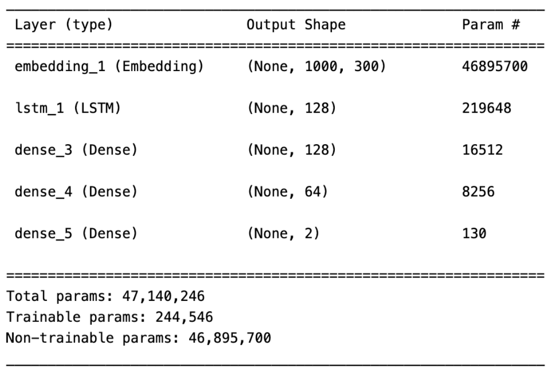
We also fine-tuned the hyperparameter - number of estimators - for our model. We ran the model for a range of estimators from 25 to 200 and found that the model performed best when the number of estimators was 125. As we increase the number of parameters, the performance of our model improves but it takes longer to run our model.

Unsupervised/ Deep Learning Models

Long Short-Term Memory (LSTM)

The motivation to use a Long Short Term Memory model was to evaluate how a deep learning model would perform in terms of predicting fake news without performing too much data wrangling. The expectation was that LSTM will help the model learn from the linguistic context that has been developed progressively because it is capable of storing information for extended time periods. The methodology used was to first get the data in a clean form, then get word embeddings using Google's word2vec and finally train a LSTM neural network to predict fake news. Word embeddings are used so that the model can learn context of the sentences from past order of words in the text as well as the future order. The

LSTM model was tested to contrast with the supervised models that do not take the context of previous inputs into account.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 1000, 300)         46895700

lstm_1 (LSTM)                (None, 128)               219648

dense_3 (Dense)              (None, 128)               16512

dense_4 (Dense)              (None, 64)                8256

dense_5 (Dense)              (None, 2)                 130

=================================================================
Total params: 47,140,246
Trainable params: 244,546
Non-trainable params: 46,895,700
_________________________________________________________________
```

Figure 3: Summary of a Neural Network With 128 Memory Units in LSTM Layer

The first model that was tested incorporated a LSTM layer with 128 memory units. The data used for LSTM prediction involved concatenated title and text of the article resulting in more than two hundred thousand parameters to train. This LSTM model achieved the highest accuracy of 0.5373 on validation data and an accuracy of 0.5481 on training data after running for 12 epochs. Running for 6 epochs resulted in training accuracy of 0.5478 and validation accuracy of 0.5376 (refer Appendix 4). This indicates that doubling the epochs virtually had no effect on the accuracy, possibly because of the sheer size of the data, the model is very slow to learn.

Retraining the same LSTM model with data consisting of concatenated author name, title and text of the articles for 18 epochs initially resulted in slightly higher accuracies (reaching 0.7321 training accuracy) until the model stalled to a training accuracy of 0.5399 and validation accuracy of 0.5358 (refer Appendix 5). Including the author name and increasing number of epochs barely impacted accuracy. This seems to indicate that the model complexity is not enough to handle data complexity. Possibly increasing the number of memory units in the LSTM layer will help the model learn more accurately given the huge corpus of data. The accuracy results of the LSTM model were surprising when compared to the results from supervised learning models. While supervised learning achieved near perfect accuracy of ~95%, the deep learning model stalled to an accuracy of ~53%. Even including the author name which is a strong predictor of reliable vs. unreliable news in the supervised learning models while training for a significant number of 18 epochs did not have much of an impact in the predictions of the LSTM model. This suggests that deep learning is not always the right solution.

Predicting something so subjective to context like fake news is a challenging task. LSTM as a deep learning algorithm completely relies on textual information to predict. However, what it learns from the text is not really interpretable particularly when we consider the broader context of what gets classified as fake news and by whom. The model relied on word embeddings as vector representations of words that occurred in the corpus of data used for training. Words that have similar meanings have the same representation so that the machine learning model can understand the context of the sequences of words. It creates an approximation of meaning for the LSTM model of what the underlying meaning of data could be. However, what associations are learned by the model is opaque. We did not have visibility into what the model puts more weight on in terms of predicting fake news.

Further, complications arise when specifying the model as it is a matter of trial and error to choose the size of the memory units required to be able to learn properly from a huge corpus of data. This issue came up as the same model with different kinds of data performed quite similarly. Hyperparameter tuning for epochs is also a consideration as spending hours training the model with poor accuracy is not a satisfactory task if it results in only being to classify fake news one out of two times approximately. For such a sensitive task, accuracy of prediction matters but we also need to think about who considers this accurate. The model has no way of learning why something is classified as fake news because it is not a matter of factually checking the veracity of the content of the article. All the model is learning is based on associations and similarity of words that occur sequentially. However, the reliability of predicting the accuracy of facts presented in an article based on word associations is suspect. This is possibly why the model is not performing well, because learning based on individual content of articles is not easy to generalize when the model might see a completely unrelated article. If the model is provided with imperfect or biased classification of fake news, then it will learn to generalize predictions based on poor associations that are not relevant to identifying what is factually correct. Therefore, it can be concluded that deep learning while being a valuable technique is also context dependent. Sometimes simpler and explainable supervised learning models can be preferred and relied on.

BERT Model

The BERT model (or, Bidirectional Encoder Representations from Transformers model) was pre-trained on large amounts of text and can predict missing words/tokens in text, understand the relationship between different sentences, etc. We were interested in investigating the BERT model because of its ability to create bidirectional feature representations of each token based on context (the text before and after the word appears), which can be leveraged to deeply analyze text and make classification predictions considering text context.[2]

In terms of model architecture, our input layer is the “bert layer” with dropout (we use both the attention mask and the bert token ids as inputs). Dropout helps to prevent overfitting in the training process and help the model become more generalizable.[3] Our first dense layer uses a tanh activation with 50 neurons, and the second (output) layer uses the sigmoid activation function. The sigmoid function allows interpreting the model output as a probabilistic prediction for our binary classification of unreliable news. We use “Adam” for stochastic gradient descent optimization, and define the loss function as binary cross entropy.[4]

Overall, this model took substantially longer than the other models to run.[5] Our final model gave a train accuracy of 91% and a test accuracy of 95%. We believe test accuracy is higher because we predict the test accuracy only using the final trained model, while early batches of the training data received prediction on a worse, less-trained model. We believe the accuracy could improve further with more epochs and more fine tuning of the hyperparameters. But, given that other models achieved very good accuracy with much shorter run time, we believe using BERT and the model architecture we laid out above may not be preferable relative to the other options. Given time constraints, we relied on existing BERT implementations to help guide our hyperparameter and architecture choices. If this project were to

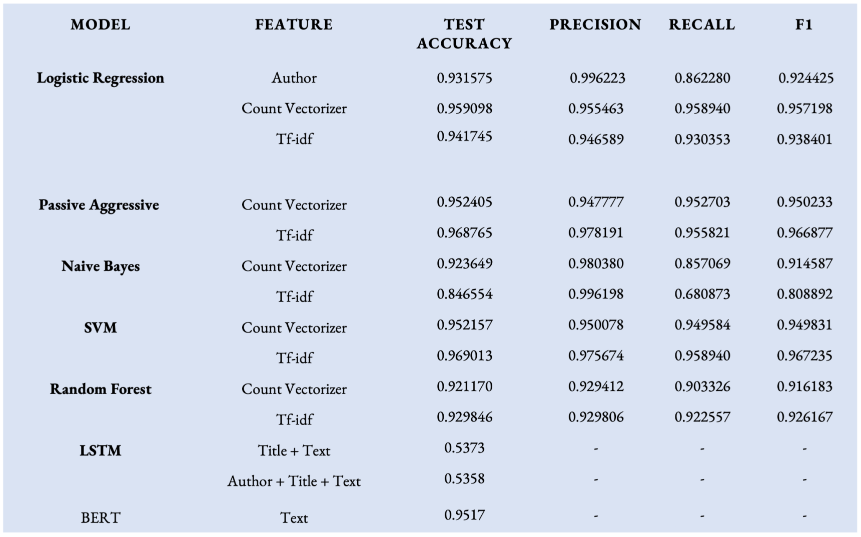
[2] For the BERT model, we used the “clean data” (punctuation removed, some data anomalies removed, etc.) rather than the stemmed and lemmatized version of the data that we used in the supervised models; BERT's pre-training uses text that is not stemmed and lemmatized, so stemming and lemmatization could make it more difficult for the model to understand the text. We chose to use the “bert-base-uncased” model variant. We chose “base” as opposed to “large” because the training data is relatively small and the task is relatively simple. The Bert model has a pre-trained tokenizer, which we used to transform our raw text into tokenized text. Due to run-time and processing concerns for the overall model, we set a max length of 75, and effectively truncate all text with more tokens. The padding and truncation parameters in the tokenizing function ensure all of the tensors are the same shape.
[3] We implemented “dropout” of .2, which temporarily “drops” 20% of the data.
[4] Binary cross entropy is appropriate given that we want to calculate the loss for our neural network after performing a binary classification prediction.
[5] Using batches of 30 and only a single epoch, the model took over 3 hours to run.

continue, we would like to more fully investigate different model architecture and parameter options: we would like to investigate other activation function options (like “relu”), we would like to test the impact of using a higher max length before truncation, we would like to test different optimizer options (ie, will a higher learning rate lead to instability? Is the decay rate optimal?), and we would like to fully investigate the tradeoff between batch size, epochs, and excessive model run times.

Conclusion

Overall Model Performance

Among our supervised models, the logistic regression, SVM and passive aggressive classifier performed the best. These models had a high accuracy rate (96-97%) and a high F1 score (96%). Common among these models was the use of TF-IDF vectorizer that takes into account relative importance and contextual information, making it best suited for our task.



MODEL	FEATURE	TEST ACCURACY	PRECISION	RECALL	F1
Logistic Regression	Author	0.931575	0.996223	0.862280	0.924425
	Count Vectorizer	0.959098	0.955463	0.958940	0.957198
	Tf-idf	0.941745	0.946589	0.930353	0.938401
Passive Aggressive	Count Vectorizer	0.952405	0.947777	0.952703	0.950233
	Tf-idf	0.968765	0.978191	0.955821	0.966877
Naive Bayes	Count Vectorizer	0.923649	0.980380	0.857069	0.914587
	Tf-idf	0.846554	0.996198	0.680873	0.808892
SVM	Count Vectorizer	0.952157	0.950078	0.949584	0.949831
	Tf-idf	0.969013	0.975674	0.958940	0.967235
Random Forest	Count Vectorizer	0.921170	0.929412	0.903326	0.916183
	Tf-idf	0.929846	0.929806	0.922557	0.926167
LSTM	Title + Text	0.5373	-	-	-
	Author + Title + Text	0.5358	-	-	-
BERT	Text	0.9517	-	-	-

Figure 4: Model Performance Metrics

Since the supervised models get high accuracies and generally had fast run-times (as seen above), they are preferable to the LSTM and the BERT models. It is possible that with more architecture investigation and parameter tuning, these models would outperform the supervised models; but in their current specifications, they are not optimal. Of LSTM and BERT, BERT achieved a higher accuracy; we think this is the case in part because BERT’s substantial pre-training and context-awareness makes it ideal for our prediction task given training and test text; on the other hand, the semantic-similarity focus of LSTM may not be ideal for this prediction task given our limited computing power. Note that we only compute accuracy for LSTM and BERT due to complexity in extracting other measures of performance.

Key Takeaways and Lessons Learnt

1. Most Helpful Features: The TF-IDF vectorizer created with bigrams was our most helpful feature in predicting unreliable news. This could be the case since it captures both the importance of individual words and the contextual information provided by word combinations, enabling the model to identify key phrases and patterns indicative of unreliable content. Further, the inclusion of bigrams enhances the representation of the text data, allowing for a more nuanced understanding of the relationships between words and improving the model's ability to distinguish between reliable and unreliable news based on specific word combinations and their significance. While TF-IDF vectorizer features were useful, they are limited since they treat each word as independent and disregard the order and structure of the text, potentially missing important contextual information. Second, these features rely solely on the frequency or presence/absence of words, overlooking the semantic meaning and subtleties conveyed by the text, which could affect the accuracy of predictions, especially in cases of sarcastic or figurative language. Finally, using these features has a high computational demand due to the sheer number of features used (each word can technically be a feature). We also found author indicator variables were also powerful features for predicting unreliable news. However, as noted in the report, this might have limited real-word usefulness as the features success depends on the training data having comprehensive authorship that is inclusive of all the authors in the real-world data we wish to predict. Also, using exclusively authors as features doesn't account for the fact that new authors may start writing articles after model training.

2. Reliability of the Kaggle dataset: We need to understand the origins of this dataset better for being confident that our underlying data is not biased. This requires us to know who are the people that built and labeled the training articles as reliable or unreliable, their worldview, and political leanings since that could have a major impact on how the models generalize on unseen data. Depending on what the definition of fake news is for the person performing the classification, the labels could change, as we cannot confirm whether fake news was assessed on the basis of factual accuracy of the reported news.

3. Model Generalization and Bias: While the provided Kaggle dataset was well suited to predicting fake news using machine learning models, we cannot verify how our models will perform in predicting fake news in the real world. There is substantial prior work using machine learning to perform binary classification tasks given training text, and we add to this body of work by implementing models that were fairly successful in predicting unreliable news. This exercise had clear classifications of reliable versus unreliable news, however, that is not the case in the real world where the models will have to make various generalizations. We find an inherent bias in the training data with it being skewed towards predicting articles that are more far-right leaning as being unreliable. This raises concerns about the motivations and interest of the person(s) who is performing the fake news classification. Here, the biases exhibited by human beings are very well being translated into biases that the model exhibits based on its biased learning.

4. Interpretability: A major concern is that of interpretability of the machine learning models. We find that supervised learning models perform better than unsupervised models in predicting news and are more interpretable. However, it is worth noting that the unsupervised learning methods we implemented have the potential to improve in accuracy with additional fine-tuning. Though it is very hard to glean what exactly are the associations being learned by the unsupervised models as we lack visibility into their inner workings.

Limitations and Scope for Future Work

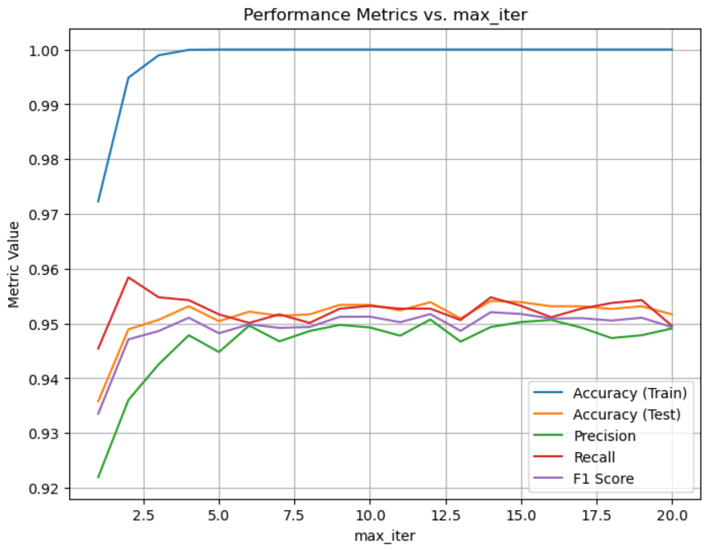
Any fake news prediction task comes with inbuilt biases due to the fact that the training data has to be labeled by humans as being either reliable or unreliable. Given that this is a judgment call by people based on their worldview, it skews our meaning of the truth. To tackle this, it is critical that any training dataset that is created is labeled by a diverse group of people with differing opinions and points of view coming to a consensus before marking an article as reliable/unreliable. Likewise, we train our model on a dataset from a particular moment in time, while written language expression trends may change over time. For example, in training our models that used vectorized text as features, we found that many of the features that were important for the prediction of unreliable news contained extreme worlds, curse worlds, etc. It is entirely possible that in the future, fake news will be written, on average, in a way that *seems* more professional and objective. With our model trained already, it may perform worse in the future as writing styles change. To combat this, the model training would have to be re-done at regular increments to keep up with the changing way fake news is presented, although there is still a concern that fake news will be harder to predict if unreliable news and reliable news writing styles converge.

If we had more time, we would have liked to make greater use of neural networks and stacked models to combine both textual and non-textual features. Moreover, we would have liked to adjust our models to better interpret different dialects. Performance issues stemming from limited neural network size, time constraints and technical constraints could have been mitigated. For example, with LSTM including more memory units on a cloud based system could have definitely improved model performance and execution timings. Likewise, for the BERT model we would have liked to be able to test out more model architecture and hyperparameters.

Ultimately, we can see that even though in this limited task our machine learning models fairly accurately predicted fake news, in general we should be cautious about using machine learning alone to predict whether an article is reliable or not. It is critical that fake news predictors are made more robust and holistic. As stated by Tom Rosentiel (Senior Fellow at Brookings), “Misinformation is like a plumbing problem you fix. It is a social condition, like crime, that you need to constantly monitor and adjust to”.

Appendix

Appendix 1: Hyperparamter Tuning for Passive Aggressive Classifier

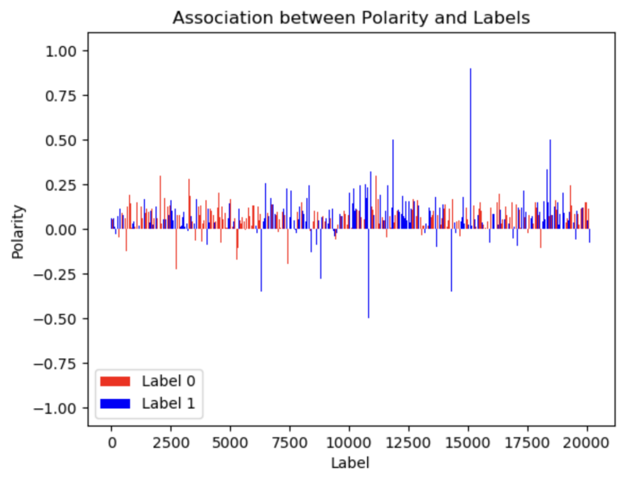


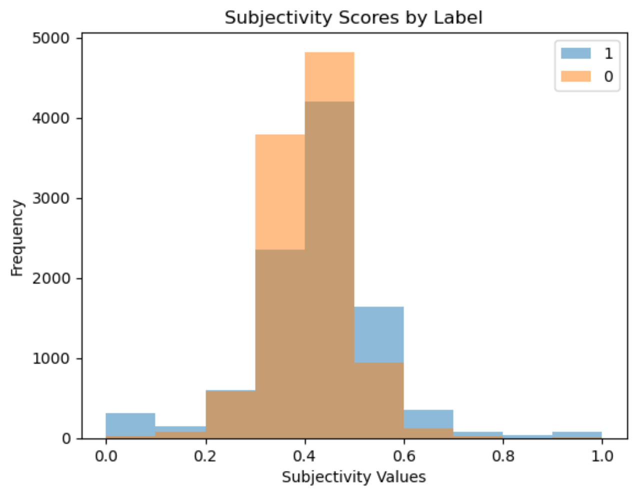
Appendix 2: Top Features for Random Forest

Top Features for Random Forest

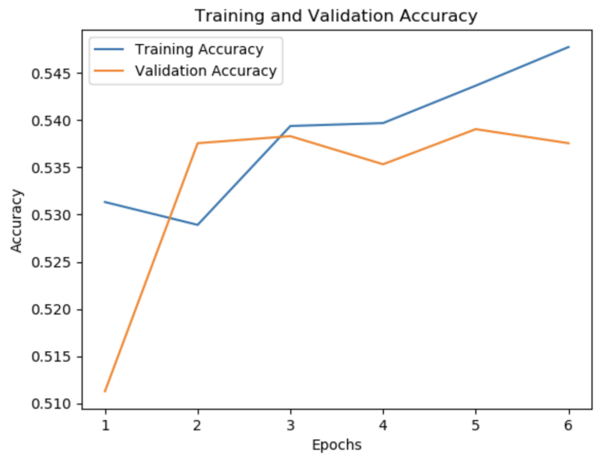
Features	Importance
leagu winchest	0.009578554861204888
syria hand	0.007711597354449654
abdolrasoul	0.006936382583147403
everywher si	0.005849749878172425
irregular nune	0.0034024296950169197
snowden launch	0.0029342689040036465
mr mont	0.0028977690487274623
jockey ayatollah	0.002755146636810198
control crown	0.002714681009471531
rico mani	0.002384475563542574

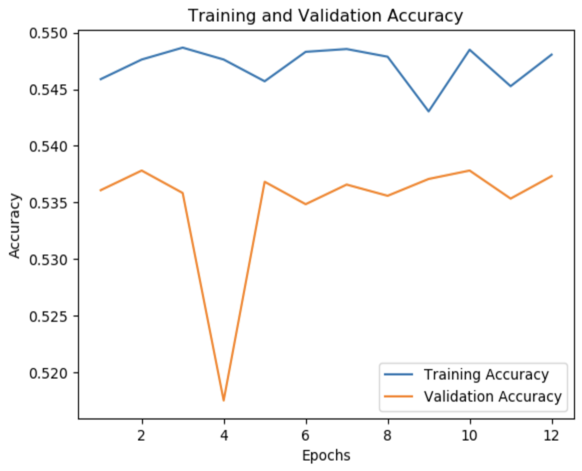
Appendix 3: Sentiment Polarity Scores



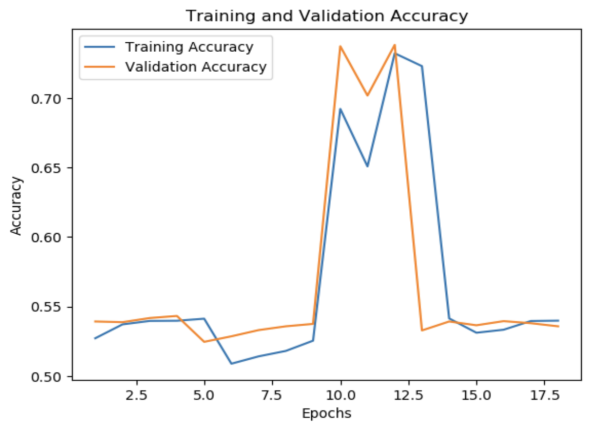


Appendix 4: LSTM Accuracy for Fake News Prediction Using Title and Text of Article (6 Epochs vs. 12 Epochs





Appendix 5: LSTM Accuracy for Fake News Prediction Using Author, Title and Text of Article (18 Epochs)



Appendix 6: Top Features for Reliable and Unreliable News for Multinomial Naive Bayes Model

Top Features for Reliable News		**Top Features for Unreliable News**	
Feature	Log Probability	Feature	Log Probability
everywher si	-9.02	vitamin best	-9.76
leagu winchest	-9.36	mann sir	-9.87
mann sir	-9.5	messag actress	-9.97
eighti georg	-9.94	mr mont	-10.09
messag actress	-10.18	claim alway	-10.35
acceler econoday	-10.24	sentenc appeal	-10.3
technolog fight	-10.24	control crown	-10.55
vaccin learn	-10.26	vaccin learn	-10.549
sentenc appeal	-10.26	quarter hous	-10.553
bednov went	-10.44	jockey ayatollah	-10.59

References

1. Barack Obama on fake news: 'We have problems' if we can't tell the difference | Social media | The Guardian
2. Misinformation is eroding the public's confidence in democracy
3. The Future of Truth and Misinformation Online | Pew Research Center
4. https://towardsdatascience.com/detecting-fake-political-news-online-a571745f73
5. Language Understanding with BERT. The most useful deep learning model | by Cameron R. Wolfe | Towards Data Science.
6. Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing.
7. Summary of the tokenizers — transformers 4.2.0 documentation
8. Summary of BERT tokenizer -- BERT documentation
9. Leveraging N-grams to Extract Context From Text | by Aashish Nair | Towards Data Science
10. Working with Text data — Applied Machine Learning in Python
11. Padding and truncation
12. Detecting Fake News — with a BERT Model | by Skillcate AI | Medium, BERT for "Everyone" (Tutorial + Implementation) | Kaggle, Fake News Detection using BERT Model Python – Towards AI, Fake News Detection Using BERT | Kaggle ,
13. Language Understanding with BERT. The most useful deep learning model | by Cameron R. Wolfe | Towards Data Science
14. The Future of Truth and Misinformation Online | Pew Research Center
15. https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c.
16. Dropout layer
17. Understanding binary cross-entropy / log loss: a visual explanation | by Daniel Godoy | Towards Data Science
18. https://keras.io/api/optimizers/adam/
19. Fake News Detection using Bi-directional LSTM-Recurrent Neural Network
20. Fake News Classifier (using LSTMs)
21. Word Embeddings and Their Challenges
22. A Guide on Word Embeddings in NLP
23. Fake News Detection Using Machine Learning Approaches