Detecting Fake News With Machine Learning

Jeffrey Huang
Amador Valley High School, 94566, USA
jeffrey.huang3.14@gmail.com

Abstract: Fake news is increasingly prevalent in our modern digital age. It ranges from misleading writing and disguised opinion pieces to pieces of satire. With the advent of social media and the growth of the internet in the 21st century, the creation, access, and spread of false information are rapidly increasing in volume. Widespread fake news can foster many problems, including misinformed public opinions and dangerous levels of political division. Current methods of tackling fake news revolve around manual reviews, fact-checking organizations, and black-listing unreliable sources. However, many of these tactics are easily exploited or ineffective. This paper applied machine learning on a Kaggle dataset to predict whether an article of news was real or fake. We applied three different classifiers, all yielding promising results.

1. INTRODUCTION
A broad definition of fake news is a collection of false, inaccurate information, fabricated to mimic the form of regular news media. Most of the time, fake news classifies as disinformation, or false information purposely created and spread to deceive people. The nature of social media and the internet makes the spread of information extremely rapid: often referred to as "going viral." These stories can take a life of their own, changing shape and sending the wrong message to millions. Situations like these can escalate to the point where it becomes difficult to discern between fact and fiction.

An extreme example of the effects of fake news is the Pizzagate shooting [12]. A fake news story, and the comments attached to it, claimed that the Comet Ping Pong pizzeria was home to a pedophilia trafficking ring led by Hillary Clinton and her presidential campaign. A man believed this story so firmly that he opened fire with an AR-15 in a local branch of the restaurant in an attempt to dismantle the non-existent trafficking ring. After discovering no sign of child trafficking, he stopped shooting and surrendered. While no one was hurt, this incident still serves as an example of the dangerous effects of fake news.

Search engines are one of the leading forces in tackling fake news. For example, Google has implemented various methods to stifle the spread of fake stories [1]. To appear on Google News, a source must first meet Google's guidelines and apply for inclusion. The decision is then made based on a manual review. Manual review lacks the efficiency to check massive amounts of articles for credibility, thereby ignoring the fact that even a credible publisher could release biased or misleading news. Another method used by Google is banning known fake news sources from receiving ad revenue, which discourages the production of fake news by removing the financial incentive. However, this method does not stop fake stories from showing up in search results, nor does it stop politically-motivated fake stories. Other methods involve funding fact-checking organizations and flagging sites as having been fact-checked to raise awareness. Fact-checking
organizations tie to another, broader problem: defining the Truth. Virtually all of these current methods depend on comparing articles to some form of objective reality. The issue with this is that it is difficult to agree on what this objective truth is, as well as who gets to define it. For example, Google must continue relying on nonpartisan fact-checking organizations because getting involved in defining the truth would draw heavy scrutiny and criticism.

Though it sounds counterintuitive, detecting fake news by directly checking the facts in an article may not be the best method. A machine learning solution would bypass the need to define the truth and would instead rely on an analysis of language patterns to determine whether a piece of news is real or not. Beyond just false information, fake news stories lure readers in, mislead them, and incite strong emotions. These motives produce specific language patterns and writing styles that may be unique to fake and misleading stories. A machine learning approach that can learn to recognize these differences in language, rather than consider the facts themselves, could be more efficient at detecting fake news. The dataset used for this study is from Kaggle.com, containing a set of fake and real news. After cleaning the data and vectorizing it, we test the effectiveness of machine learning in classifying the stories.

2. BACKGROUND
The machine learning approach implemented in this study was a simple binary classification architecture. By analyzing the language of thousands of articles in the dataset, the model theoretically should pick up on the text's nuances, which it can then use to classify a piece of news as real or fake. A problem solved in this fashion is spam email detection. The two issues are remarkably similar: spam emails, like fake news articles, are designed to lure and mislead people while appearing in obnoxious quantities and filling up a user's feed. Sharaff et al. compared the performance of four different types of classifiers on labeling emails in a binary fashion: spam or legitimate [8]. Their best performing model achieved an accuracy exceeding 0.93, which shows promise for a binary architecture in predicting the legitimacy of news pieces. Other similar areas of study include a different method of categorizing fake news called stance detection. Thota et al. focused on implementing a neural network architecture that determines the stance of an article in regards to a given headline [10]. The article could agree, disagree, be neutral, or be completely unrelated. By comparing the article's stance and the mainstream stance on the headline, their model would predict whether the article was fake or real. Their models achieved accuracies of over 0.94 when tested on the Fake News Challenge [4] data set. Other previous studies factor auxiliary information besides the text itself into their predictions, such as sentiments [13] and the social engagement and profiles of viewers [14]. In this specific study, we will be focusing on a binary classification architecture, considering just the text provided in the dataset.

3. APPROACH
The dataset used is available on Kaggle, and contains one set of real news and another set of fake news. All the models here were created through Jupyter Notebook and its available software libraries, including Tensorflow [15], Keras [11], SciKit-Learn [9], Numpy [16], Pandas [17], and more.

3.1 CLEANING THE DATA
We started by extracting 15,000 fake articles and 15,000 real articles and combining them into one dataset, shuffling for randomness. Then, in preparation for applying the models, we began to clean the data. First, we removed all NaN values (empty entries). Second, using the Regular Expression Operation "sub," all characters besides the letters "A" through "Z" were removed [3]. After this, we imported Stopwords and PorterStemmer from the Natural Language Toolkit [7]. The stop-words dataset contains the most common words in the English language that have little to no impact on meaning, such as "the," "a," "an," or "in." PorterStemmer is an algorithm that normalizes text by simplifying variations of a word into a single common root; for example, it simplifies "jump,"
"jumper," "jumping," and "jumped" into "jump." After removing all stop-words and applying the PorterStemmer to the remaining text, we were left with a clean, normalized dataset.

3.2 VECTORIZATION
The next step was to choose how to represent the text to feed into a model. We used three main options: a CountVectorizer (Bag of Words), a Term Frequency-Inverse Document Frequency Vectorizer (TF-IDF), and One-Hot representation into an embedding layer [9]. The bag of words vectorizer collects unique words from the text and the numerical instances of each word. The TF-IDF vectorizer operates similarly, except words that are common throughout all of the data have their weights, or relative importance, decreased. This algorithm produces an arguably more accurate representation of the words in the text and their relative importance. We set both of these vectorizers to extract the top 5,000 most frequent words. The one-hot encoding into an embedding layer essentially creates, for each article, a multi-dimensional vector representing a sequence of words. This method of encoding takes into account the sequential nature of writing: in the one-hot representation, the sentences "live to eat" and "eat to live" are distinct, whereas, in the previous two methods, they are considered equivalent. The one-hot encoding is also implemented with a vocabulary size of 5,000. Each one-hot vector's dimension is also limited to 100, which means each vector represents 100 words from its corresponding article. This action helps to shorten the compile time for the model. The Keras embedding layer then creates a text-feature matrix that captures similarities and connections between articles. Each of these vectorization techniques collects, to some degree, a level of semantic information and meaning from the text. For this study, the embedding layer extracted 100 features to create the matrix.

3.3 APPLYING CLASSIFIERS
After converting our text into vector form, the data was ready to be used. We tested a Multinomial Naive Bayes [6] and a Passive-Aggressive [2] classifier for both the bag of words and TF-IDF representations. We applied a Long-Short Term Memory model [5] to the one-hot into embedding layer representation.

The Multinomial Naive Bayes model uses Bayes theorem and conditional probabilities to produce a probability for the article belonging in each class. It is a fast, simple, model, and called "naive" because it assumes all features are independent. The Passive-Aggressive classifier, rather than using the dataset as a whole, takes in one piece at a time, adjusting the weights of its model based on each entry's results. If the prediction is correct, the model is not changed: it is "passive." If the prediction is incorrect, it adjusts the weights of the model until the prediction becomes correct: hence it is "aggressive." PA algorithms are ideal for classifying massive streams of data. Both the MultinomialNB and PA algorithms do not consider word order in their calculations, so we apply them with the bag of words and TF-IDF representations, both of which do not represent word order as well.

The LSTM model simulates human interpretation, in the sense that it considers an article as a sequence of words, and learns to extract relevant information from it. In this way, the model takes into account the sequential nature of written work. The one-hot encoding represents each article as a sequence of words and preserves the order of the text, so it is ideal for applying the LSTM model. As for hyperparameters, the LSTM had two dropout layers with a value of 0.3: one after the embedding layer, and one after the LSTM layer. This layer was simply a measure to avoid over-fitting, a phenomenon where a model trains so well that it ends up memorizing the training set instead of learning for it. The model used the binary cross-entropy loss function, a sigmoid activation function, and the "adam" optimizer [11]. All the details are depicted below in Table 1.
Table 1. LSTM Model, Embedding and Classifier Details

| Name                  | Value   |
|-----------------------|---------|
| Vocabulary Size       | 5,000   |
| Max One-hot Vector Length | 100    |
| Extracted Features   | 100     |
| Dropout Value         | 0.3     |
| Batch Size            | 512     |
| Epochs                | 10      |
| Loss Function         | Binary Cross-Entropy |
| Optimizer             | “adam”  |
| Activation            | Sigmoid |

4. RESULTS

We randomly selected 33% of the data for testing, while the remainder was used for training purposes. Because selection was random, both the training and test sets had virtually the same amount of real and fake news articles. In total, five models were fit and tested. The confusion matrices for each are shown below.

**Figure 1.** Confusion matrix for CountVectorizer and MultinomialNB

**Figure 2.** Confusion matrix for CountVectorizer and PA classifier

**Figure 3.** Confusion matrix for TF-IDF Vectorizer and MultinomialNB

**Figure 4.** Confusion matrix for TF-IDF Vectorizer and PA classifier

**Figure 5.** Confusion matrix for one-hot encoding and LSTM
The two tables in blue were generated from applying the Multinomial Naive Bayes classifier (Fig. 1) and the Passive Aggressive classifier (Fig. 2) to the bag of words representation of the data. Both models are respectable, achieving accuracies of around 0.91 to 0.92, with the rate of false-positive and false-negative being about the same.

These next two tables in red are again the Multinomial Naive Bayes classifier (Fig. 3) and the Passive Aggressive classifier (Fig. 4), this time applied to the TF-IDF representation of the articles. Because the TF-IDF classifier takes into account the redundancy of overly common words, the prediction accuracy was expected to be higher. That turned out to be the case, with the Multinomial Naive Bayes model returning a test accuracy of 0.954, and the Passive Aggressive classifier climbing to an impressive accuracy of 0.992.

The last table in green (Fig. 5) is the testing results from the LSTM model, applied to the one-hot representation of the dataset. The accuracy was again decent, with 96.8% of cases being labeled correctly. Even though it takes into account the sequential nature of the text, the LSTM model still does not perform as well as the simpler PA classifier. However, the results were still satisfactory.

5. CONCLUSION
Our results indicated that all of the models performed well in identifying fake news. As expected, switching from the count vectorizer to the TF-IDF vectorizer improved the accuracy of both the Naive Bayes and the Passive Aggressive classifier. The more realistic representation of the relative importance of words helped the algorithms make better predictions. The LSTM model also performed very well, beating out the Naive Bayes classifier.

We found that a wide variety of machine learning algorithms all demonstrate a strong ability to detect fake news. It appears that comparing the facts themselves is not necessary when trying to detect disinformation. Instead, the algorithms were able to detect nuances and patterns in the articles effectively, and they were able to use this information to label the test data to a very high degree of accuracy. These results show promise towards implementing machine learning as a standard method of detecting misleading and incendiary news in the future, which would be far more efficient than relying only on manual review and fact-checking organizations.

The logical next step would be to test additional machine learning models in order to find an optimal one. Another aspect to expand upon is the models themselves, by introducing more complicated network structures such as a Convolutional Neural Network. Other future areas of study could apply this framework to different platforms: for example, rather than detecting just fake news articles, the same architecture could be applied towards detecting fake headlines and stories on Twitter, Facebook, and other social media platforms. The future end goal is a universal, efficient algorithm for detecting fake and misleading stories across all avenues of the internet. Testing models on sets of data from different platforms would be an ideal next step towards expanding in this direction.

REFERENCES
[1] Bowden, I. (2016, December 1). How Google is tackling fake news, and why it should not do it alone. Search Engine Land. https://searchengineland.com/google-tackling-fake-news-not-alone-264058
[2] Crammer, K. C., Dekel, O. D., Keshet, J. K., Shalev-Shwartz, S. S., & Singer, Y. S. (2006). Online Passive-Aggressive Algorithms. Journal of Machine Learning Research, 0. http://www.jmlr.org/papers/v7/crammer06a
[3] https://docs.python.org/2/library/re.html
[4] http://www.fakenewschallenge.org/
[5] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
[6] Kibriya A.M., Frank E., Pfahringer B., Holmes G. (2004) Multinomial Naive Bayes for Text Categorization Revisited. In: Webb G.I., Yu X. (eds) AI 2004: Advances in Artificial
Intelligence. AI 2004. Lecture Notes in Computer Science, vol 3339. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30549-1_43

[7] https://www.nltk.org/

[8] Sharaff A., Nagwani N.K., Dhadse A. (2016) Comparative Study of Classification Algorithms for Spam Email Detection. In: Shetty N., Prasad N., Nalini N. (eds) Emerging Research in Computing, Information, Communication and Applications. Springer, New Delhi. https://doi.org/10.1007/978-81-322-2553-9_23

[9] https://scikit-learn.org/

[10] Thota, Aswini; Tilak, Priyanka; Ahluwalia, Simrat; and Lohia, Nibrat (2018) "Fake News Detection: A Deep Learning Approach," SMU Data Science Review: Vol. 1 : No. 3 , Article 10. Available at: https://scholar.smu.edu/datasciencereview/vol1/iss3/10

[11] https://keras.io/

[12] Haag, M. (2020, June 27). Gunman in ‘Pizzagate’ Shooting Is Sentenced to 4 Years in Prison. https://www.nytimes.com/#publisher. https://www.nytimes.com/2017/06/22/us/pizzagate-attack-sentence.html

[13] O. Ajao, D. Bhownik and S. Zargari, "Sentiment Aware Fake News Detection on Online Social Networks," ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, United Kingdom, 2019, pp. 2507-2511, doi: 10.1109/ICASSP.2019.8683170.

[14] K. Shu, S. Wang and H. Liu, "Understanding User Profiles on Social Media for Fake News Detection," 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Miami, FL, 2018, pp. 430-435, doi: 10.1109/MIPR.2018.00092.

[15] https://www.tensorflow.org/

[16] https://numpy.org/

[17] https://pandas.pydata.org/