**Classifying fake news articles using multiple machine learning algorithms**

Team 14: Zachary Almaraz, Caleb Fowler, Jonathan Pierre, Zachary Vasey

**Abstract**

In this study, we explore the viability of using machine learning algorithms to determine whether a news article is fake or real. Using a dataset of 38,729 news articles, we compare the success of four different algorithms including Naive Bayes, K-Nearest Neighbors, Support Vector Machine Classifier, and a Recurrent Neural Network in an attempt to find an algorithm that reliably differentiates between fake and real articles. Ultimately, we find that while our recurrent neural network gave us near-perfect accuracy and the other algorithms achieved moderate success, there is concern that the same success would not occur on a different dataset. Finally, based on what we learned, we offer some possible procedures that could be used in the future to achieve better results.

**Introduction**

Though the rise of the internet, the amount of news that is readily available to the general populace has increased substantially. According to Pew Research Center, 50% of individuals aged 18 - 49 identify the internet as their main source of news. The information distributed over the internet may not be written by credible sources, causing the spread of fake news. Some fake news articles may be blatantly obvious, while others are more subtle. While fake news can sometimes be hard to differentiate, the attributes of the articles can be quantified for analysis and may reveal some underlying trends that can be exploited using machine learning.

We obtained a dataset containing approximately 21,000 real news articles and 23,000 fake news articles. The dataset was broken up into two csv files: one containing the true news articles and the other containing the fake news articles. Both had four columns: “title”, “text”, “subject”, and “date”. The title is the title of the article, the text is the body contents of the article itself, the subject is the subject of the article, and the date is the date when the article was published. We combined the two csv files into one and added a new column: “fake”. This column is either a 0 indicating true news or a 1 indicating fake news.

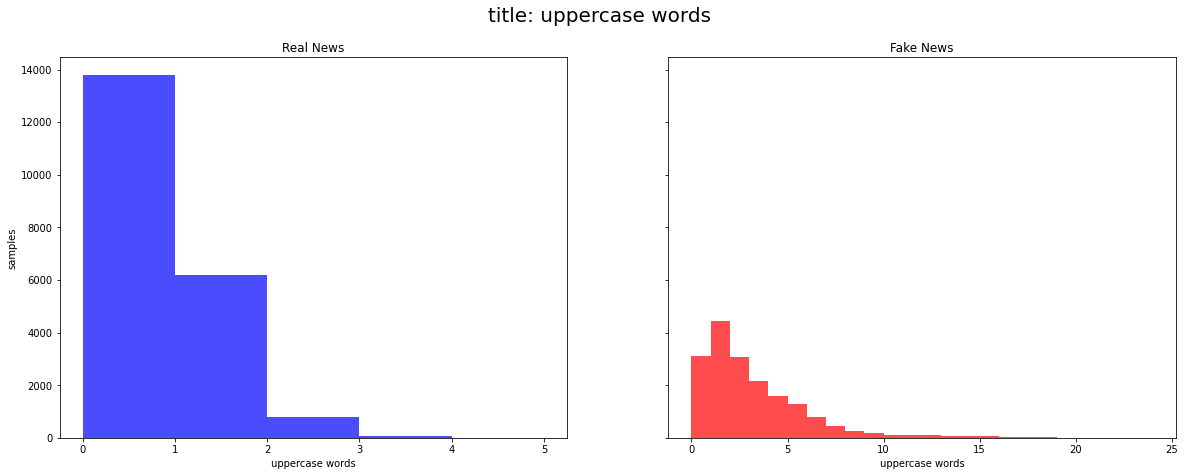
Duplicate news articles were found by searching for identical titles. Of the original 44,919 news articles in this combined dataset, 6,169 were discarded as duplicates and 21 were discarded for having missing values. Thus, this dataset had a total of 38,729 news articles we could use for classification.

**Methodology**

Our team approached the fake news classification using various machine learning techniques. Some classifiers were trained using an extracted featureset while others used the original featureset. The extracted features are highlighted below.

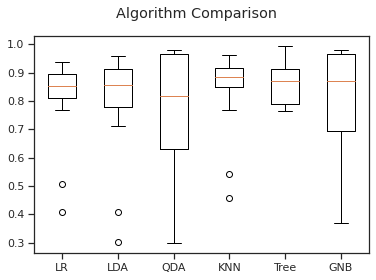
|  |  |  |
| --- | --- | --- |
| **Extracted Feature** | **Data type** | **Description** |
| num\_words\_title | int | Number of words in the title |
| num\_chars\_title | int | Number of characters in the title |
| avg\_word\_len\_title | float | Average length of a word in the title |
| num\_uppercase\_words\_title | int | Number of all uppercase words in the title |
| num\_words\_text | int | Number of words in the text |
| num\_chars\_text | int | Number of characters in text |
| avg\_word\_len\_text | float | Average length of a word in the text |
| num\_uppercase\_words\_text | int | Number of uppercase words in the text |

The uppercase words feature was added after noticing that many of the fake news articles had titles in all capital letters. As real news articles didn’t seem to have this characteristic, we believed it may be a good feature for classification. We defined an uppercase word as having more than 1 letter, with all letters in uppercase. Below is a box plot which shows the number of news articles that have varying frequencies of uppercase words in the title of the article.



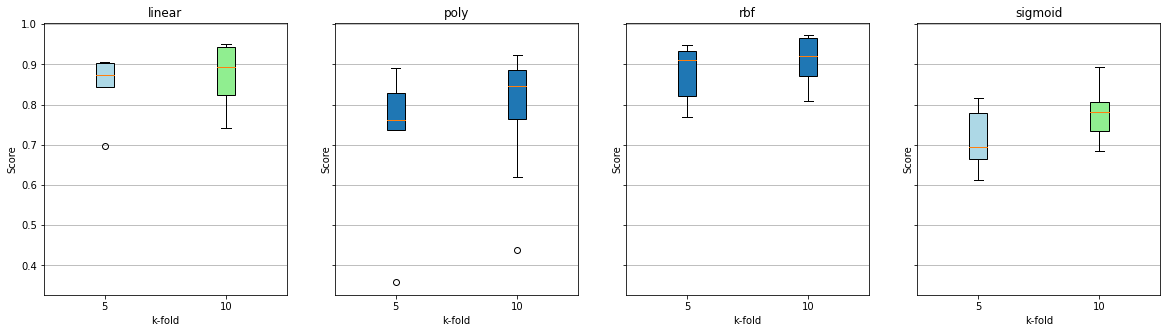
As can be seen, the number of uppercase words in the title of the article can be a good indication of whether or not the article is real or fake. For example, if there are more than 3 uppercase words in the title, the article is almost certainly fake.

The boxplot below shows some initial accuracy scores for various models.

We were surprised by the high accuracies for all of these models. These accuracy scores come from cross validation with a k-fold of 25. It should be noted that using a k-fold of 25 yielded significantly better accuracies as compared with a k-fold of 5.

The K-nearest Neighbors classifier (KNN) was selected after comparing it with five other algorithms: Logistic regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, a decision tree classifier, and Gaussian Naive Bayes. As the above graph shows, KNN outperformed the other tested algorithms based on average accuracy with an average of 86%. Additionally, it had the highest minimum score of all the algorithms, though most of the other tested algorithms had a higher maximum score. Using k-fold cross validation with folds of 5 and 10 and testing all k’s from 1-100, we determined that the best k-value for our purpose was 41 because it yielded the highest average accuracy.

The Support Vector Machine Classifier was trained on the extracted featureset. It was chosen because it had a relatively high accuracy score for initial training for default settings. SVC is a maximum soft max classifier which creates a margin to separate features. As the dataset has 8 dimensions, excluding the target, the hyperplane has a dimensionality of 7. The dataset seemed to have slight nonlinearity after comparing features with a pairplot graph. Although SVM is a linear classifier, we trained the dataset on various SVM kernels, to introduce some non-linearity to the SVM model, in an attempt to improve its accuracy score. Below, we have included a comparison of 4 SVM kernels and their accuracy scores. Each kernel’s accuracy score was determined using cross validation accuracy score with a k-fold of 5 and 10.



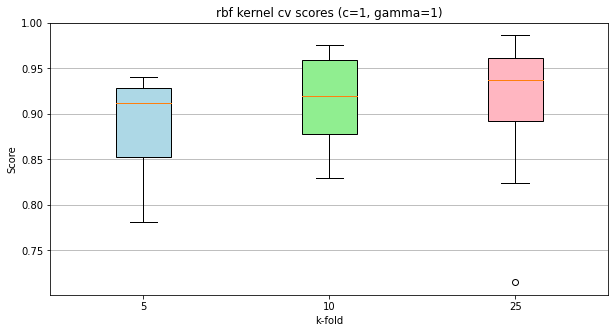
As expected, the linear kernel performed fairly well. The rbf kernel performed the best as it was able to introduce slight nonlinearity to the model. Because our dataset is fairly linear, the other kernels were not as accurate. For a k-fold of 10 using rbf kernel, our SVM yielded an accuracy mean score of 90.9% with a standard deviation of 5.8%.

Having determined rbf as the best performing kernel, to improve accuracy we retrained the rbf kernel using various tuning parameters. The tuning parameters we tested are listed in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **C** | 0.1 | 1 | 10 |
| **Gamma** | 0.1 | 1 | 10 |

The C value helps to determine the size of the margin. As it increases, the margin decreases and the model overfits. The Gamma value determines how far a support vector influences the decision, and as it increases, the model overfits. When gamma is small, it doesn’t accurately capture the shape of the data. The gamma value is used by the rbf kernel exclusively.

After retraining using these tuning parameters, a C value of 1 with a Gamma value of 1 yielded the best accuracy. Below, we have included cross validation scores for the rbf kernel using a C value of 1 and a Gamma value of 1, with K-folds 5, 10, and 25.



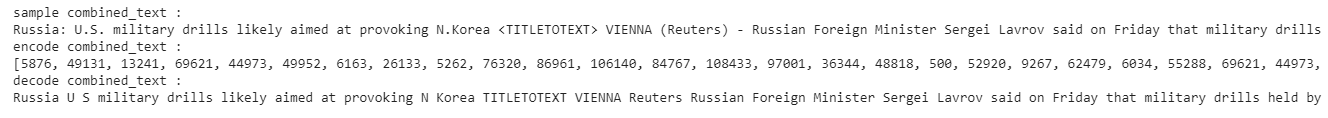
A cross validation score with a k-value of 25 yielded a mean accuracy score of 92.2% with a standard deviation of 5.9%. The median accuracy score, as depicted in the above boxplot, yielded a higher result because there were some low scoring outliers affecting the mean accuracy score.

Naive Bayes was chosen for text classification. While researching multiple methods for text classification, Naive Bayes appeared frequently. Because of its popularity and the need to classify the data into two categories, fake and real news, Naive Bayes was chosen to use on the dataset. First, the data needed to be preprocessed. After preprocessing the title and text columns, two new columns were added to the dataframe: “clean\_title” and “clean\_text”. In addition, repetitive words called stop words were removed from the columns with the help of the Natural Language Toolkit Library (NLTK). These included words such as “to”, “he”, “is”, “not”, and more. A little over 100 more words were removed from the titles and text. TF-IDF was used to transform the text and title columns into feature vectors. A new category called “fake” was used to separate the number of fake articles and titles. The most frequent words found in each title and column were used to determine the count of articles titles and text found to be fake with fake value 0 meaning real news, and fake value 1 meaning fake news. The model was then fit with Multinomial Naive Bayes to produce an f1 score and accuracy.

|  |  |
| --- | --- |
| Train Data | Test Data |
|  |  |

To complement the results, word clouds were produced to visualize the results of the Naive Bayes model for the most frequent real news and fake news words. Some noticeable words found in the real news word cloud were “country”, “north korea”, “reuters”, and “government”. Some found in the fake news word cloud were “via”, “image”, “donald”, “trump”, and more. It was immediately apparent that the words “via” and “image” in the fake news word clouds possibly provided a strong indicator between fake or real news because these words did not also appear in the real news word cloud.

|  |  |
| --- | --- |
| Real News Word Cloud | Fake News Word Cloud |
|  |  |

Recurrent Neural Networks are well known to be useful in text processing. We built an RNN model using the original dataset as an input. Strings cannot be used by an RNN, so we first needed to encode the words in the title and text into numerical values. The title and text were combined into one string separated by the keyword <TITLETOTEXT>. The keyword would ensure the RNN would recognize the transition from the title to the text. The text was encoded using the TensorFlow library. All unique words were given an unsigned integer value and special characters were eliminated. The encoded string could now be used as an input for the RNN model.

Two RNN models were created. The first model consisted of one embedding layer, one long short - term memory layer, and two dense layers. The second model consisted of one embedding layer, two long short - term memory layers, and two dense layers. The training data for this model consisted of 9% of the total dataset and the test data consisted of 1% of the total dataset. The set of train data and test data did not intersect and both contained roughly equal numbers of true and fake news articles. The two models were trained to 50 epochs initially to identify the best epoch value to train each epoch. 15 epochs was chosen for the first model and 10 epochs was chosen for the second.

|  |  |
| --- | --- |
| RNN Model 1 Accuracy(50 epochs) | RNN Model 1 Loss(50 epochs) |
|  |  |
| RNN Model 2 Loss(50 epochs) | RNN Model 2 Loss(50 epochs) |
|  |  |

**Data Analysis**

The two best performing models were RNN and NB.

The RNN model was evaluated using accuracy and loss as metrics. Both RNN models performed extremely well. Model 1 had an accuracy of about 96.9% and a loss of 0.0746 and model 2 had an accuracy of about 98.4% and a loss of 0.0598.

|  |  |
| --- | --- |
| RNN Model 1 Accuracy(15 epochs) | RNN Model 1 Loss(15 epochs) |
|  |  |
| RNN Model 2 Loss(10 epochs) | RNN Model 2 Loss(10 epochs) |
|  |  |

The Naive Bayes model was evaluated using F1 score and Accuracy as metrics. The model performed well with 95.22% F1 score and accuracy on the test data, 95.47% F1 score on the train data, and 95.48% accuracy on the train data.

|  |  |  |
| --- | --- | --- |
| Test Data Results | F1 Score | Accuracy |
| Percentage | 95.22% | 95.22% |

|  |  |  |
| --- | --- | --- |
| Train Data Results | F1 Score | Accuracy |
| Percentage | 95.47% | 95.48% |

**Dataset Analysis**

Our models’ performances were very high which caused us some concern. Upon further analysis, we discovered some patterns in the dataset that may explain the high performance. The first pattern we discovered was that the word “Reuters” was in many of the true datasets. Reuters is a trustworthy media organization that produces credible news. Upon further inspection, most of the entries had the media organization the article came from written somewhere at the beginning of the body text. This could have had significant influence on the predictions of the NB and RNN models as these models could learn what news outlets are reputable or not. The second pattern we discovered was that many of the fake news articles contained uppercase letters while the true news articles did not. This could have increased the performance of the models using the string characteristics substantially. The website we retrieved the dataset from had discussion boards that confirmed the patterns we noticed. Some individuals were claiming their models reached up to 99% accuracy and were blaming these patterns for the high accuracy[4]. Any work done with this dataset in the future must address these problems to produce models of value. In the current state, the models’ high performances are most likely misleading.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | KNN | SVM | NB | RNN |
| Accuracy | 0.866 | .922 | 0.952 | 0.985 |
| Loss |  |  |  | 0.077 |

**6. Conclusions**

To conclude, there was success using multiple machine learning approaches to solve the problem of finding out whether news is real or fake. Some accuracy may appear to be too high in the models bringing concern to whether the models would perform well given a new dataset. Some procedures that possibly could help are removing words that are too frequent in the dataset such as “reuters”, “president”, or other possible words which occur frequently in either the real news or fake news data. Also, many fake news articles from outside this dataset will not have the high frequency of all uppercase words seen in this dataset, lending doubt to the sourcing of the fake news articles.

**7. References**

|  |  |
| --- | --- |
| [1] | Amy Mitchell, Jeffrey Gottfried, Michael Barthel, and Elisa Shearer. 2016. How Americans get their news. (July 2016). Retrieved May 6, 2020 from <https://www.journalism.org/2016/07/07/pathways-to-news/> |
| [2] | Ahmed H, Traore I, Saad S. “Detecting opinion spams and fake news using text classification”, Journal of Security and Privacy, Volume 1, Issue 1, Wiley, January/February 2018`. |
| [3] | Ahmed H, Traore I, Saad S. (2017) “Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127- 138). |
| [4] | Amy Mitchell, Jeffrey Gottfried, Michael Barthel, and Elisa Shearer. 2016. How Americans get their news. (July 2016). Retrieved May 6, 2020 from <https://www.journalism.org/2016/07/07/pathways-to-news/> |

**8. Appendix**

1. Paper
   1. Abstract - Caleb Fowler
   2. Introduction - Zachary Vasey
   3. Methodology
      1. Feature analysis - Jonathan Pierre
      2. KNN - Caleb Fowler
      3. SVM - Jonathan Pierre
      4. NB - Zachary Almaraz
      5. RNN - Zachary Vasey
   4. Data Analysis
      1. NB - Zachary Almaraz
      2. RNN - Zachary Vasey
      3. Dataset Analysis - Zachary Vasey
   5. Conclusions - Zachary Almaraz
   6. References - Zachary Vasey
   7. Appendix - Zachary Vasey
2. Code
   1. Modified Dataset - Zachary Vasey and Jonathan Pierre
   2. Comparing Algorithms - Zachary Vasey, Jonathan Pierre, and Caleb Fowler
   3. KNN - Caleb Fowler and Zachary Vasey
   4. SVM - Jonathan Pierre
   5. NB - Zachary Almaraz
   6. RNN - Zachary Vasey
   7. Code Organization - Jonathan Pierre