NLP Classifier Project

Chris Corona cjcorona11@gmail.com
Montana State University
Bozeman, MT 59718, USA

Abstract

In today's fast-paced world with 24-hour news cycles, we are constantly being bombarded with information. Sometimes it can be hard to determine if a news article is fake or real, especially breaking news where the full details are not yet known. We have seen, time and time again, fake news used as a weapon to mislead and misinform the public. In many cases fake news has been used to undermine democracy by spreading falsehoods about political issues and even elections. It is important that we can separate truth from fiction. In this project, we will develop various machine learning models that will be trained to tell the difference between fake and real news articles. The models will take as an input the full text article and will output a binary response as either "fake" or "real". Any consumer of news might find this tool helpful to determine if a news article is worthy of reading. Social media apps where news articles are posted and shared in real-time might find interest in such a tool as a sort of censor or filter - possibly blocking fake news from being posted or adding a disclaimer message to posts that may contain fake news. Maybe even public officials would want this tool to evaluate the messaging being circulated about a particular topic, so that the officials can specifically correct any outside attempts at misinformation.

To try to solve the problem of classifying articles as real or fake, we implemented four different classification models: logistic regression, naive Bayes classification, a random forest, and a neural net. We will be training these models with a set of token frequencies (a document term matrix) derived from a text corpus of documents. The predictor set for each model will be all tokens that appear in some minimum threshold of the documents in our corpus. We will also consider models that incorporate multi-token predictors (N-grams) and evaluate these augmented predictor sets for additional prediction performance.

1. Introduction

The dataset that we'll be training and evaluating our models on is an updated version of the McIntire dataset (*https://github.com/lutzhamel/fake-news*). This is a synthesis of 6,335 fake and real news articles from various web sources. According to George McIntire, the fake articles came directly from a Kaggle data set called "Getting Real About Fake News" (*https://www.kaggle.com/datasets/mrisdal/fake-news*), and all were published in the month leading up to the 2016 presidential election from October 25 to November 25, 2016. The real articles were scraped by George from All Sides (*http://www.allsides.com/*), a website dedicated to hosting news and opinion articles from across the political spectrum. These real articles include left, center, and right-leaning articles from numerous media outlets published between 2015 and 2016. It is a balanced data set with 3,164 fake and 3,171 real articles. For each article, the data set contains 4 features: A unique ID number, the article's title, the article's text, and the response variable of whether or not the article is fake or real news.

We will attempt to fit four different classes of model for this problem: logistic regression, naive Bayes, a random forest, and a neural network. The most basic model we will fit is a logistic regression model which will serve as a baseline to compare the other, more sophisticated models. In logistic regression, we transform a linear model with the logit function to calculate the probability of a binary outcome - "fake" or "real" news.

Next, we will use a naive Bayes model, which is slightly more complex than logistic regression, bolsters independence assumptions amongst features, and uses class conditional probabilities to classify examples. It is important to note that our feature set for this investigation, a set of words or n-grams, are likely not independent due to in-article context, and so we might expect lessened performance on this model.

Third, we will fit a random forest model that will attempt to stratify articles into real and fake classes based on some combination of words or phrases that are found within these articles. We will use the basic greedy approach of identifying words/phrases that are most strongly associated with real/fake articles and split based on these first. But in a random forest, instead of choosing from all possible predictors, we will choose from a random subset. This removes correlation between the individual trees in the forest and improves performance.

Finally, we will experiment with a simple neural net to classify articles as real or fake. We will use a basic single-layer perceptron, with an input layer that accepts words or n-grams as features, an output layer with 1 node representing a probability of the positive class (similar to logistic regression), and a hidden layer with 10 nodes. Note, this simplistic network architecture is due to our lack of computing power for this project.

For each model, we will employ 10-fold cross validation and evaluate final results based on F1 score. F1 score was chosen because we think there is a roughly equal cost to false positives and false negatives for classifying fake/real news.

2. Data Preprocessing

Language is an art, and thus an inherently complex domain for machine learning. There is an entire grammar governing the rules and guidelines of how to compose a sentence, a paragraph, a complete story. Words have many meanings and be used very specifically with nuance or very broadly and loosely. There is meaning in word choice. Style, voice, and tone that can be folded in. Understanding the author, the audience, and the context adds even more depth. So how do we work with language such that machine learning algorithms can use this data?

To start, we are going to first clean the text. To do this, we must tokenize the body of the text into the individual words. We used the **'tm'** package to achieve this and the subsequent steps. We converted all letters to lowercase and removed punctuation. For our purposes, these conventions are not going to be necessary. Next we must consider that not every word in a document is important. Many words do not add much meaning but are usually just grammatical (eg "the", "and", "of", "is"). We call these stopwords - common words that contain no significant meaning. There is a stopword list built in to the 'tm' package that we used here. We also want to do what is called stemming and lemmatization. This essentially factors a word down to its base component, reducing the many derivations of a word to one single item (eg "ran", "running", "runner" = "run"). We use the **'SnowballC'** package for stemming. Somehow, this stemming and lematization process converted some words back into stopwords, so we performed stopword removal again here. The final step is to vectorize or transform the text into a numerical format that the model can process. This means generating a matrix where each row is an article, and each column is a possible word in the dictionary. The matrix counts how many of each word are found in the given article. So an article is transformed into a sparse list of word counts - mostly zeros because an article contains only a small selection of all possible words in the dictionary.

We also decided to experiment with n-grams, or n-sized word groupings. Individual words contain meaning, but there is more meaning in multiple words strung together. For example in "trading good" vs. "trading" and "good". The different encodings have very different meanings. The 2-gram "trading good" refers to an item that is traded while the 1-gram interpretation "trading" and "good" puts a positive spin on a trade. Of course as we increase n, we stand to gain more information but at the cost of expensive computation. We will evaluate all of our models on the predictor set of 1-Gram tokens, 1-Gram tokens and 2-Gram tokens, and 1, 2, and 3-Gram tokens.

3. Exploratory Analysis

We began by performing some exploratory data analysis, in the form of generating the 30 "realest" and the 30 "fakest" words, decided by evaluating the ratios $\frac{Pr(word|REAL)}{Pr(word|FAKE)}$ (30 realest) and $\frac{Pr(word|FAKE)}{Pr(word|REAL)}$ (30 fakest) for every token in our document term matrix. The results are as follows for the set of single word tokens:

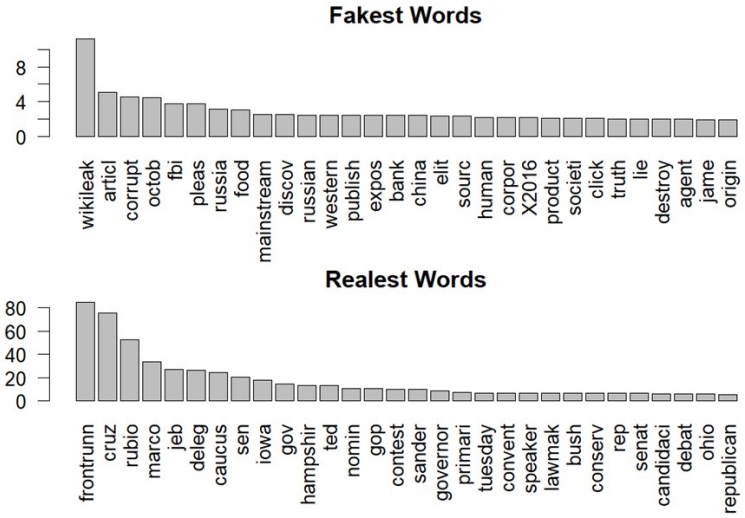

Figure 1: 30 Realest and 30 Fakest Words, Single Tokens

4. Performance Evaluation

To attempt to classify articles as real or fake, we developed a logistic regression model, a naive Bayes model, a random forest model, and a neural net, as described above. We used the $F1$ scores of all of these classifiers to evaluate classification performance. The results are as follows:

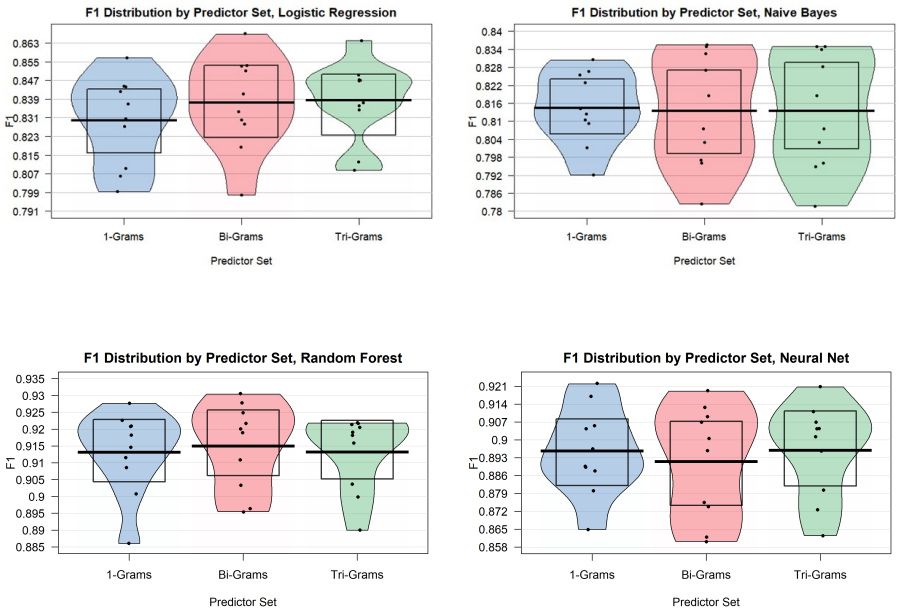

Figure 2: Distributions of 10-Fold F1 scores for each Model and predictor set

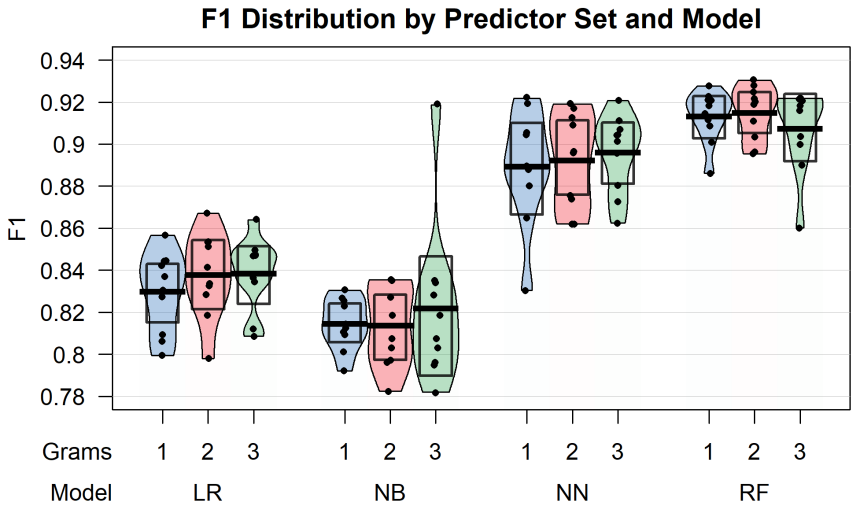

Figure 3: Distributions of all 10-Fold F1 scores for all models and predictor sets

5. Conclusion

Based on our results above, it seems clear that the two best models were the neural network and the random forest, with the logistic regression and naive Bayes models experiencing almost a 10% performance difference. This suggests that this is likely a non-linear classification problem, as both linear models struggled relative to their non-linear counterparts. Additionally, there was very little impact of adding the Bi-Grams and Tri-Grams to the predictor set - we think that this is likely because there were not a significant amount of either that made the sparsity cutoff. Only roughly 50 Bi-Grams and 3 Tri-Grams were present in 5% or more of the documents, so they had minimal impact in most models.

In the future, we would like to acquire more computing power and fit more complex neural networks - more layers and more nodes - and see just how well such networks can learn the classification problem. Additionally, we would like to tune our corpus cleaning methods - optimizing processes like stopword removal, stemming, lemmatization, and sparsity removal could yield much more effective predictor sets than that of which we worked with. Finally, we are interesting in adding part-of-speech tagging as a covariate to continue to try to emulate the context and flow of syntax that is present in coherent written text like the articles we're evaluating.