Fake News Detection with Naive Bayes algorithm

*Machine Learning Project

Muhammad Shahzeb Kayani
Dept. of computer engineering
Air University
171209

***Abstract*—WWith the current political climate and parties more polarized than ever, the dissemination of accurate and unbiased information is more important than ever. However, there are limited resources for automatically identifying fake news articles on the Internet. Most modern machine learning pipelines that utilize deep neural networks (DNNs), which can model complex language features such as grammar, have been proven to be effective but computationally expensive.**

***Index Terms*—Machine, Learning, Clustering, unsupervised learning**

I. Methodology

We treat the problem as a conditional probability task using Bayes' theorem, where our classifier attempts to predict the likelihood of a class (i.e. fake or real) given a document D. More specifically, we attempt to find the likelihood that a word W is likely to be from a fake news document with class F, and then combine those probabilities to generate a prediction for the whole document. The simple Bayes' formula can be expressed as follows.

$$P(F/W) = \frac{(P(W/F)P(F))}{P(W)}$$

For our purposes, the extended form of Bayes' theorem was particularly useful, because P(W) could be expressed in terms of P(W/F) and P(F), under the assumption that classes F (fake) and ¬F (real) are mutually exclusive. P(W/F) is the frequency of word W in all fake documents in the training set, P(F) is the frequency of documents of class F.

$$P(F/W) = \frac{(P(W/F)P(F))}{(P(W/F)P(F)) + (P(W/-F)P(-F))}$$

Finally, we calculate the likelihood of a document being fake, under the assumption that the likelihoods of words appearing in fake and real documents are independent, by multiplying the P(F/Wi) for all words in the given document.Because this is a binary classification problem, if the P(F/D) ¿ P(¬F/D), we can infer that the document is likely to be fake.

II. Solution

The quantity of data required in order to produce accurate predictions is well beyond the realm of manual calculation, so we wrote a program using the Python programming language to calculate the probabilities for the Bayes classifier. This also gave us the opportunity to further optimize our data set.

III. Dataset

Our dataset came from a compilation of two datasets independently published on the popular opensource software website GitHub. The dataset was used in research published by Horne et. al, 2017. The first half of the dataset consisted of Buzzfeed Political News Data, categorized accordingly by analysts at Buzzfeed. The second half of the dataset included randomly collected political news data from real sources (i.e. Wall Street Journal, The Economist, BBC, NPR, USA Today, etc.) and fake sources (i.e. True Pundit, DC Gazette, Liberty Writers News, InfoWars, etc.). For the random political news partition, the sources were verified as real or fake based on Business Insider's "Most Trusted" list and Zimdars 2016 Fake News list.

For our purposes, we combined the two partitions of the original dataset into two categories, one sample of 128 real news articles and another sample of 123 fake news articles.

IV. Document Preprocessing and Tokenization

Using Regular Expression tokenization, we were able to remove punctuation and more easily segment words in a given document. We also removed stop words, or words that appear frequently and give little indication of class, such as "the", "and" or "as". Given stop words' frequency, even small variations in their usage could be enough to drown out the words that are good indicators of a particular class.

V. Results

Using the methodology described in Section 1, we trained our classifier on a sample of 128 real and 123 fake documents consisting of 128,858 words (71,731 after the removal of stop words). As a result, our program was able to predict the class of a document it had never seen before with 77.6 accuracy. The false positive rate for misclassifying a real news article as fake was 5.4 and the false negative rate was 38.5.

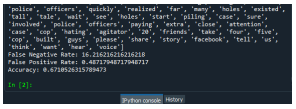

Fig. 1. Final Figure

VI. Advantages

Due to the simplicity of Bayes theorem, training a Bayesian classifier requires very little computational power and can be performed within computer applications such as email. In addition, a user can fine-tune a model to make predictions on certain words that are specific to that user, such as names of relatives or services that a user knows are not spammers. Most importantly, the Bayes classifier does not need to have any working knowledge of English language, an abstract form of communication, and can perform generally accurate predictions without knowing the actual meaning of words.

The simple technique we explored has a plethora of applications: sentiment analysis, bias removal, spam filtering, and almost any other classification problem where the features analyzed can be said to be independent. On that note, there are a variety of disadvantages.

VII. Disadvantages

Because the Bayes classifier learns to make predictions based on the likelihoods of certain words, the classifier is susceptible to Bayesian poisoning, a type of attack where spammers or news writers embed phrases from real sources to confuse the classifier. In addition, disguising text by embedding it in an non-machine-readable image prevents the computer from taking that text into account when making its prediction.

Our model's accuracy also suffered from the fact that it was trained on a relatively small dataset. Initially, we trained our model on half of the original dataset, and received an accuracy score of 60 on our test data, which indicates that the Bayesian model improves with more data. In general, training on more data will educate the model on other contexts, regularize its output, and prevent optimistic prediction. However, little can be done to prevent the model from being unreliable when large quantities of data are not available.

VIII. Conclusion

One commonly used technique that goes beyond stop word removal is to only consider a small set of the top N words most associated with a class. However, choosing how large N should be is difficult, as too small of an N will sacrifice some of the model's flexibility. Given more time, we could have performed additional experiments with varying values of N and integrated this optimization into our program. As previously discussed, techniques that can model language have been developed, but require calculus-based methods that can be computationally expensive and are beyond the scope of this project. They would, however, be able to solve problems that arise from our assumption that word occurrence is independent when in reality word occurrence is inherently dependent. Finally, our accuracy could have been improved with access to more, higher quality data.