CS5830 Project 5: Naive Bayes

Caden Maxwell, Calvin Bell, Karl Poulson

Introduction

Our analysis delves into the domain of fake news detection, a critical area given the proliferation of misinformation in today's digital age. Leveraging a dataset sourced from Kaggle, a data science and machine learning competition platform, we utilized a Multinomial Naive Bayes model to predict whether a sample of news articles were real or fake. Additionally, based on our models, we tried to find lists of words that may be more indicative of real news than fake news or vice versa. Through our analysis, we achieved relatively high metrics and overall promising results, which validates the potential for our model to play an important role in the ongoing battle against fake news.

Github: Naive Bayes
Slides: Google Slides

Dataset

The dataset used in this analysis comprises 72,134 news articles, including 35,028 instances of real news and 37,106 instances of fake news. It was procured using information from four separate information sources, including another Kaggle dataset, and McIntire, Reuters, and BuzzFeed Political data. This diverse dataset represents a much more generalized sample than any of those data alone, which helps to ensure the robustness of our model and prevent overfitting. The dataset itself is structured around three key features: the title of the article, the text composing the article, and an authenticity label, where a label of 0 denotes an instance of fake news and 1 indicating real news.

Dataset: Kaggle

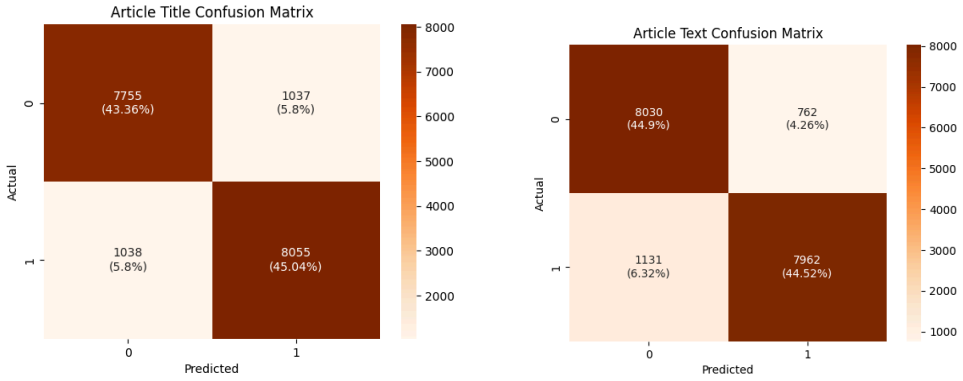
Analysis Technique

Utilizing something like word frequencies is a very typical approach to creating a Naive Bayes model. In order to do this, we needed to take every occurring word in the training set and make a new feature from it for each article. With each of these new features, we then got a word count for each of those features from each article. During this, we filtered out commonly used English words which are present for grammatical purposes but do not add any meaning to the text given. We decided to fit one model on only article titles, and another model on only article content. Doing this seemed suitable for our data since article titles and text both generally follow different structures, carry different tone and mood, etc. To assess the performance of our models, we looked at metrics such as precision, recall, and f1-scores. Additionally, we performed what was

essentially a vote between both the title and text models by taking the logical OR of the two prediction vectors; this is a common practice in the field of machine learning when multiple models are involved.

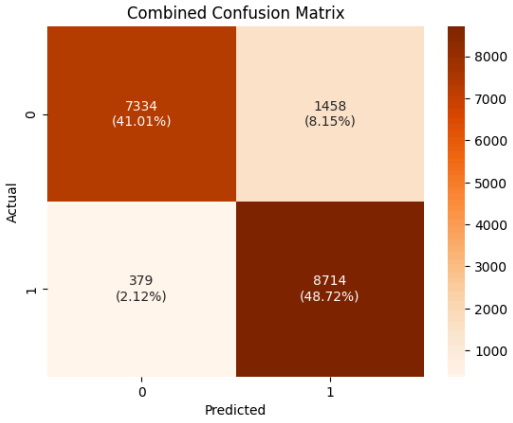
Results

We found our models to be very reliable for predicting whether a given news article was real. Our first Multinomial Naive Bayes model, which we only fit to article titles, received strong metrics, with a notable f1-score of 0.8859 (precision and recall were 0.8859 and 0.8858, respectively). Our second model, fit to only article contents, received similar metrics, with an f1-score of 0.8938, precision of 0.9127, and a recall of 0.8756. The following figures display the resulting confusion matrices for the title- and content-based models.



While the content-based model was consistently better at predicting when a fake article was actually fake, it did mislabel slightly more real cases as fake than the title-based model, which adversely affected its recall metric.

Utilizing the predicted values from both of these models, we were able to perform a simple vote, which resulted in an f1-score of 0.9046, precision of 0.8567, and a very high recall of 0.9583 for predicting the “real” label. The following is the confusion matrix for this vote.



This voting mechanism seemed to work very well for our data, as we received f1-scores (like the one above) which were consistently better than those of the raw title- and content-based predictors themselves.

Finally, we used our text-based model to find the most occurring words, and use their probabilities to decipher whether they are most used in fake articles or real articles. Below are words that had a higher probability of appearing in fake news content.

	P(w | Fake Text)	P(w | Real Text)	Difference	More Likely	Occurences
said	0.015566	0.004541	0.011025	Fake	175805
would	0.005252	0.003930	0.001322	Fake	79022
president	0.004286	0.003563	0.000723	Fake	67406
mr	0.005623	0.000580	0.005043	Fake	54788
new	0.003581	0.002600	0.000981	Fake	53227
also	0.003096	0.002605	0.000491	Fake	48947
state	0.003224	0.002286	0.000938	Fake	47467
states	0.002640	0.001767	0.000873	Fake	38002
government	0.002512	0.001856	0.000656	Fake	37604
could	0.002514	0.001838	0.000676	Fake	37466

This was interesting, because despite our efforts to remove common English words that have little to no meaning, some still appeared like “said,” “would,” and “also.” However, our results were still very informative and gave us some insight as to which words may be more indicative of that article being real or not.

Overall, our models fit the data well, boasting high f1 and recall scores. We were also able to find many words which appear consistently more frequently in fake news and real news. Thus, this analysis may provide great indicators to the general public to discern between real and fake news articles in the sea of misinformation today.

Technical

The dataset itself was pretty clean and usable. In order to extract the word count features, as well as actually get the count of each word per article, we were able to use a CountVectorizer. This did all of the heavy lifting for us, from lowercasing and tokenizing all of the words, to removing all of the stop words and turning each article instance into a vector of word counts.

We found our results by utilizing the MultinomialNB classifier provided by scikit-learn. We used a multinomial model instead of a Gaussian model since the data was neither continuous nor ordinal, and we have seen before that word counts are much more fit for a multinomial approach. We used a vote after fitting and getting predictions from both of our multinomial models in order to try and receive a better score. While this approach is not novel by any means in the world of data science, it is novel to what we’ve learned in the course up to this point.

The MultinomialNB classifier offers a way to extract the feature log probabilities, which we were able to convert to regular probabilities by exponentiating. Next, we were able to get the difference between probabilities for each word and their appearance in one of the two classes in order to determine which it was more likely to appear in. This process was not very obvious at first, and we had to dig through documentation to figure out what “feature_log_probs” really meant in order to get it right. In terms of alternative approaches, I think that it would have been a good idea to utilize some sort of cross-validation. While we got consistent results with a random test set every time we ran the program, it still would have been nice to be absolutely sure that our model was generalizing well.