Abtahi Manzur

CS-UY 4563

**Real/Fake News Project Report**

**Description:**

The objective of this project is to create a simple binary classifier using supervised learning methods. This binary classifier will take in a transcript of a news article as an input and it will output whether or not that new article is “real” or “fake”. The creator of the dataset claims that he obtained the data from [www.politifact.com](http://www.politifact.com), and that he would like to diversify his sources but he could not find any other website that classifies news articles that were at the very least as trustworthy as this particular website. Looking through the dataset, it does not seem ridiculous to assume that the news articles were likely classified correctly. However, even if the labels are biased, training a classifier on the data can give us insight on what the people who labelled the articles were looking at when deciding if they are “real” or “fake”. Perhaps outside of even the factual content of the articles, there are certain words and speech patterns that raise red flags.

The training dataset consists of 38,729 transcripts of news articles that were initially labelled as “real” or “fake” news. Specifically, 20,826 of the articles were labelled as “real” news and 17,903 of the articles were labelled “fake”. So, the training dataset is roughly split as 53% “real” news and 47% fake news which can be considered a baseline for the classification methods. The data also consists of 4 columns for each article; the title of the article, its transcript, the date that the article was written, and what type of article it is, although they are all news articles, so this column is pointless. All of the articles also were written in the month of December 2017, with the dates ranging from December 1st, 2017 to December 31st, 2017.

**Data Pre-Processing:**

The field of machine learning is referred to as natural language processing, or NLP for short. The text that makes up a news article contains a lot of information unnecessary for classification. For example, words like “the” or “a”, commonly referred to as “stop words”, emojis, capitalizations, punctuation, numbers etc. are all unnecessary when trying to train a simple classifier on the data. A more complex classifier may want to take some of these things into account. Simple classifiers want to look at unique words which make up the bulk of what makes a certain piece of text unique. Then those unique words must be converted into a data vector of 1s and 0s which is what classifiers can actually work with. To do this, however, first the data must be tokenized, which means the text is split up word by word for example. There is also another concept referred to as stemming and lemmatization. This is when words are standardized in the sense that every variation of a single word is converted to its base form. For example, for the word “car”, any word like “car’s”, “cars’”, “cars”, etc are converted to just “car”. For words like “resting”, “rested”, “restful”, etc, are just converted to “rest”. This is done to simplify the data and because of the fact that these stems and tenses carry little of the meaning compared to the base word itself.

A screenshot of a cell phone

Description automatically generated Using The natural language toolkit, the data can be tokenized and then capitalization on can be converted to simple lower case, stop words can be removed, and then punctuation can also be removed. Using the scikitlearn library later on, we can remove the stems and lemmatization from words and vectorize the data using a bag of words model in order to apply classification techniques on them. Below is an example of a few articles using the .head() command to see the difference between the text and the filtered text using the natural language toolkit:

Figure : 4 rows of data matrix with raw and filtered texts

**Analyzing the Data:**

First, I was interested in what exactly makes up fake news articles. Using the word cloud library, you can look at the processed data pre vectorization and display it in a word cloud to see the frequency of the words that appear in the news articles. Below is the word cloud for the real news:

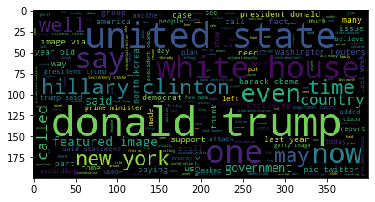


Figure : Real news word cloud

Below is the word cloud for the fake news:

A close up of a black background

Description automatically generated

Figure : Fake news word cloud

It is interesting to note the vocabulary differences between the labelled, “fake” and “real” news. There are quite a few similarities, for example, the word “Donald Trump” was mentioned many times in both “real” and “fake” news. This is unsurprising, of course. Some of the major differences, however, include the fact that “real” news mentions the word “United States” very often compared to not much at all in the “fake” news. The “fake” news has words like “said”, “twitter”, “american”, “featured”, “image”, that are not mentioned as much in the “real” news.

**Classifying the Data:**

A few basic classifications methods were trained on the dataset using functions in the scikitlearn library and their performances were compared to each other. The sklearn library has a pipeline that allows many classification models to be easily implemented on the data. For this project I looked at the K-nearest-neighbors, Naïve Bayes, Support vector machine, and Logistic regression.

**K-nearest-neighbors:**

The Knn classifier achieved a maximum accuracy of about 79.62% and that was when was near 1. As n increased as seen by the graph below, the accuracy started to decrease. It takes too long to test every n so only the first 10 are shown but when n is 50 for example, the accuracy decreased to around 50% as well. This isn’t surprising. Knn is best used in multi class classification where you are trying to match data to training data that ideally look identical to it and output that class label. For news, “fake” or “real” are based on facts and potentially word choice as well, but that doesn’t mean two articles will contain the same words. Thus looking for a larger number of neighbors introduces much more variation in what type of articles you encounter where with a single neighbor, chances are much higher they share the same binary class label if they are that similar.

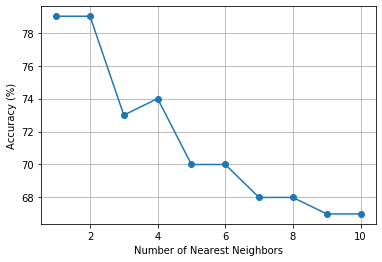


Figure : Accuracy as nearest neighbors increased

**Bernoulli Naïve Bayes:**

Bernoulli Naïve Bayes is a type of Naïve Bayes classifier that is useful when each feature in the input is a binary number with a certain probability of being 1 or 0. This is true for a Bag of words model to vectorize filtered text, which is why it is useful for classifying text. Applying this classification method, we get a much better accuracy of 94.14%. Below we see the precision and recall of this method.

A picture containing meter

Description automatically generated

Figure : Precision and Recall of Bernoulli Naive Bayes

**Support Vector Machine/Logistic Regression:**

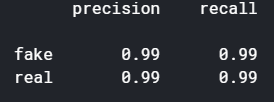
 Both of these classification methods yielded the best results with 98.76% accuracy for the SVM classifier and 99.55% accuracy for the logistic regression classifier. The precision and recall were 0.99 which is practically 1.

Figure : Precision and recall of SVM and logistic regression classifier

**Conclusion:**

Based on the most common words in “fake” and “real” news. It seems that “fake” news tends to base a lot of assumptions on hearsay and jumping to conclusions based on images without fact checking first. I am basing this assumption off of the fact that words like “said” and “image” are so much more common in “fake” news than in “real” news. In “real” news the word “united states” is more common than in “fake” but I’m not sure as to why. Perhaps the “real” news is more objective about news concerning the United States?

If I had more time for this project, I could have trained more complex models involving kernels and neural networks and could have also tried to implement a lot of basic models manually. I could have also attempt to input articles outside of the dataset from particular websites to see what the SVM/Logistic regression classifier would classify them as. I could have also looked at some individual articles that are most people would classify as “real” or “fake” and see how the classifier classifies them. Based on the vocab, however, it is likely that the classifier would be blind to articles that are simply factually incorrect but it would easily detect articles that appeal to emotion to try and rile up the reader and conspiracy articles. Potentially I could have also performed some more advanced analysis such as looking at things like negative word usage, punctuation usage, word distribution. I would imagine a lot of fake news would use negative words to spin a story a certain way and use more exclamation and question marks.