Donovan Guelde
Kaggle username: Donovan
CSCI 5622
Feature Engineering

In trying to correctly predict spoilers versus non-spoilers, several difficulties were posed, first by the raw data itself, then in the training/prediction stage. The training data, made up of over 14,000 samples, contains numerous spelling, punctuation, and grammar irregularities that could conceivably interfere with accurate model training and prediction.

To improve prediction accuracy, the first step I took was to remove stop-words via the nltk.corpus functionality, with punctuation included. By removing commas, apostrophes, etc, it was my belief that accuracy would improve through the standardization of words. For the same reason, I made all characters lower case. The next step I took in feature engineering was to take the results of the first step, removing stop-words, and apply word stemming. By using word stemming, I was hoping to increase model effectiveness by eliminating word variance caused by grammar and possibly misspelling. The nltk lancaster stemming functionality was used. The lancaster function is relatively aggressive in practice. As an example, given the sentence 'Stemming is funnier than a bummer says the sushi loving computer scientist', the lancaster function returns 'stem is funny than a bum say the sush lov comput sci.' Computer is shortened to 'comput', and scientist is pared down to 'sci.' The lancaster was not my first choice of stemmers, as I thought it was too aggressive, unnecessarily decreasing the amount of data, some of which may prove to be useful. However, after trying different functions, I found that this stemmer, in fact, improved accuracy more than other models.

The next step I took was part of an effort to allow the classifier to have some sense of 'context.' Using the nltk functionality, I had every word tagged with its part of speach, and then joined the word and its tag to form a new word. For example, the word 'calculator' could be used as a noun or verb, but a bag of words model would not be able to distinguish the two. With a part of speech tag, a bag of words model will now be able to distinguish between "computer_noun" and "computer_verb". One unexpected result was that I had better results with part of speech tagging after stemming was applied. My prediction was that applying tagging to an unstemmed word, then stemming, would give good results, but the opposite happened.

The last step I took in feature engineering was to form bigrams with the stemmed, tagged samples. Through the use of bigrams, I was again hoping to give the classifier a sense of context about the text sample. A bigram, for example, may allow the classifier to discover patterns between noun-verb pairs.

The final step I took was to simply append the 'trope' value of the original sample to the end of the engineered text sample. During a superficial examination of the data, I noticed what appeared to be a correlation between the 'trope' and spoiler tag. I have no explanation for this, except that reviewers of some 'tropes' are more or less careful about

using spoilers.

By applying these steps, removing stop-words, stemming, part of speech tagging, forming bigrams, and appending 'trope' value, I was able to achieve a final accuracy of 0.68428. This was a much smaller improvement than I was expecting, but it was consistent over several runs of the classifier.