Meghan Hickey
Stat NLP Fall 2017

Assignment 1: Naïve Bayes

This Naïve Bayes classifier is constructed using Lidstone smoothing. With a basic bag-of-words approach to features and a smoothing factor alpha of 0.01, it achieved 66% accuracy. With that as a baseline, I tested out other features in an attempt to improve the classifier's performance. To measure performance, I used accuracy and an unweighted averaged F measure. Results are below, with feature combination that performed the best on the balanced data set (bag of words, smoothing factor of 0.1) highlighted in yellow.

Smoothing Factor	Feature Type	Balanced?	Accuracy	F-score
1.0	Bag of words	Yes	62%	0.580
1.0	Bag of words	No	89%	0.473
1.0	Tokenization	Yes	62%	0.587
0.1	Bag of words	Yes	72%	0.718
0.1	Bag of words	No	77%	0.528
0.1	Tokenization	Yes	71%	0.707
0.01	Bag of words	Yes	68%	0.680
0.01	Bag of words	No	76%	0.532
0.01	Tokenization	Yes	67%	0.674
0.001	Bag of words	Yes	63%	0.632
0.001	Bag of words	No	84%	0.511
0.001	Tokenization	Yes	62%	0.601
0.01	Bigrams	Yes	69%	0.702
0.01	Trigrams	Yes	62%	0.624
0.001	Bigrams	Yes	66%	0.663
0.001	Trigrams	Yes	61%	0.619

Tokenization

In addition to trivially tokenizing the text (as is done in the pre-built BagofWords document class), I tested what would happen if the text was tokenized more intelligently and if stop words were removed. To do this, I implemented a separate Tokenized Document class. The feature method of this class considers stop words to be all punctuation, plus words in NLTK's stop words corpus. It tokenizes the text using NLTK's Penn Treebank word tokenizer and then removes stopwords. Tokenization did not significantly improve results. In fact, the non-tokenized model performed slightly better than the tokenized model, though the differences in performance between the two were really negligible.

Smoothing factor

I changed the smoothing factor alpha, using factors between 1.0 and 0.001. The best results were achieved with a smoothing factor of 0.1. Factors at both ends of the

tested range (1.0 and 0.001) produced poor results on the balanced data set but had high accuracy on the imbalanced data set. Models with extreme learning rates had low F-scores on the imbalanced data set, though, suggesting that they achieved such high accuracy by highly favoring one class over the other.

N-grams

I tried bigrams and trigrams with learning rates of 0.01 and 0.001. The bigram and trigrams models took considerably longer to train and test that the unigram models did, likely because there are many, many more different bigrams and trigrams in the data than there are unigrams.

I expected the n-gram models to outperform the simple tokenized and bag of words models, because the n gram models capture information about word co-occurrence. Interestingly, while the bigram models outperformed the trigram models, neither one beat the performance of a simple bag of words.