Charles Calder
November 7th, 2017
CMSC 251 – Homework #4

Version 1: Raw Naive Bayes

Findings:

Based on the results of my testing, the raw Naive Bayes (NB) classifier is extremely unreliable in classifying test data. Over three tests, the mean percentage of correctly guessed categories based on the words in each line was 7.3%. This percentage was calculated outside of the rec.motorcycles category, because this category has a mean of ~99% correct guess rate.

Source of inaccuracy:

The source of the low average correct guess rate is the same as the source of the extremely high average correct guess rate. The raw probability of any word is directly influenced by the number of times that word occurs in the training data. When the program runs into a word that occurred zero times in the training data, the raw probability is zero. Here we run into an issue when trying to pick the maximum probability from a list of zero's. Python defaults to returning the first value. Since rec.motorcycles is the first index in the list of categories this results in an extremely inflated percentage of correct guesses. Over three tests, the mean number of times that rec.motorcycles was guessed was 7044. This value is two orders of magnitude higher than the number of guesses for any other category.

These inaccuracies indicate that there are many lines in the test data that contain at least one word that did not occur in each category in the training data.

Version 2: M-Estimate Naive Bayes

Findings:

The m-estimate NB classifier greatly improves on the accuracy of classifying test data based on each line. Over the course of three tests, the mean accuracy over all categories was 80.3%. If the program checks a word from the test data against the training data and finds that the word does not occur in a given category in the training data, the word is given a very small percentage value based on the training data, instead of being given a zero percent probability. Thus, the extreme growth in accuracy occurs by completely removing the problem of returning a zero percent probability for any given category.

Version 3: TF-IDF Naive Bayes

Findings:

The tf-idf NB classifier is meant to improve on the M-Estimate NB classifier by weighting each probability by the number of categories each given word can be found in. This means that if the program runs into a word that is not in a given category in the test data, the program weights the likely hood of finding that word in the category by one over the number of categories that word occurs in generally. In this case, however, over the course of three tests, the tf-idf NB classifier scored only slightly higher than the m-estimate (82.01%). I believe this stems from the fact that there are few words that span

across multiple categories, resulting in only small amounts of weighting for almost every probability.

Final Evaluation:

The Best Version:

I found though trial and error that on average the raw NB test takes around 15 – 20 seconds, the m-estimate NB test takes around 30 – 35 seconds, and the tf-idf takes around 40 – 50 seconds. Based on these values and the accuracy of each type of probability explained above, tf-idf is this best version for this task. Tf-idf has the best accuracy and only takes slightly longer than m-estimate.