Richard Young
CS165A MP1

MP1 Report

Architecture

There are 5 class files, one of them being used as the “main” function to execute for the .jar. The Classification.java file’s purpose is to act as a storage container for features and the result of their classification, which is used in the Classifier.java file. The Classifier file’s purpose is to provide an abstract base for the BayesClassifier file to build off of, implementing very basic functionality. FeatProbability is a basic interface used to calculate individual feature probability that is used in many calculations of the BayesClassifier. Lastly, the BayesClassifier file is an extension of the Classifier, but adds the functionalities that make it a BayesClassifier, with the ability to classify given documents.

Preprocessing

The document is read through a BufferedReader in java, and is taken line by line then broken into an array of words to be either trained on, or classified by the Bayes Classifier. Currently each word is it’s own separate feature, as I did not go as far as to implement more complex things such as a list of words to ignore, or words that couple well together.

Model Building

The Classifier is trained by being passed a collection and a category, which is then turned into a Classification object that holds the features and the corresponding category. Each feature is then stored and incremented depending on how many times each feature shows in the collection. This classification object is also sent to the memory queue, and then determines if it needs to forget any older classifications based upon how often they’re used.

Results

On the provided data set, I got an average training time of 1 second, and an average labeing time of 2 seconds. In addition, the training and testing accuracies both average out to about 66%. I think it’s fairly accurate given I had not yet implemented more complex methods of learning and document cleaning.

Challenges

The biggest challenge of this entire project was trying to think of a structure for the project that works well, and makes logical sense. Java can be a very intuitive and clean language if you put the time into it, and I think that it came together relatively well.

Further than that, the actual implementation of classification in the BayesClassifier.java file took me forever to work through, especially to get it to output the way I had wanted it to. The last challenge I faced was attempting to get everything to work and output the way the validate file wanted it, in addition to the generation of the runnable .jar file required.

Weaknesses

As mentioned before, some sort of document cleaning, or ignore word list would make this machine much more accurate than it already is, that’s probably the most glaring weakness. I also did not take into account run time for my functions, so given an appropriately large data set, I imagine the system would not run as efficiently.