Wilson Mui

MP1

Architecture

The program consists of three files: NaiveBayesClassifier.java, Document.java, and stopwords.txt. The main is in the NaiveBayesClassifier file. All stop words used in the program is stored in stopwords.txt. These stop words were by: http://www.lextek.com/manuals/onix/stopwords1.html .

NaiveBayesClassifier.java receives the training and testing file. The training file is processed line by line to train the model. Then, the training file is processed line by line to test the model. The same is done next with the testing file.

Testing each document is done by splitting the document out of the line, and then converting the document into a list of words. This list of words is passed into the model to guess, and a guessed label is returned.

Document.java contains the Category class and Model class. A Category is created for each author, while only one Model instance is used throughout this program. The model stores the vocabulary for all of the training set, the stop words from stopwords.txt, and a map linking the author labels to their Categories.

The most important methods of the Model class are guessClassLabel(ArrayList<String> document) and optimize(). The rest of the methods are mainly helper functions. guessClassLabel will return the most probable label given the document. This is to be used after training the model. optimize() will make the model run quicker and is meant to be called after training the model as well.

The Category class contains a list of all the words from all documents relating to the author. The vocabulary from the author is also stored. A map is used to quickly access the word probability of any word.

Preprocessing

Each document is represented as a list of words. To prepare the document for training, words are removed from it. All stop words from stopwords.txt are removed from the document. Then the last two/thirds of the words are simply dropped from the list. Removing stop words is intended to prevent unimportant words from having any effect on the classification. Removing the last two/thirds is done to make the classification process quicker.

The features are essentially the vocabulary of the training set without the stop words.

Model Building

Training the classifier is done in a straightforward manner. For each document in the training set, the stop words are removed. Then the vocabulary of the model is updated and the rest of the words are added to a list of words from the given author.

After all the documents are added, the model is ‘optimized.’ All word probabilities are calculated for each author. This prevents having to calculate it later. Then the six most frequent words are removed from the authors list of words.

The class of a document is ‘guessed’ by assuming independence of all the features, or words. Laplace smoothing is done to handle new words. The probability of a word in a class calculated by finding the frequency of a word from the author, adding 0.2, and then dividing by the sum of the number of words in the author’s list and the model’s vocabulary size.

Results

Using the given training and testing set, the program was done executing in about 8 minutes. The accuracy was 0.994 for the training set, and .750 for the testing set. The time took 28 seconds and 33 seconds.

The most important features were determined to be all the words in the training set’s vocabulary minus the stop words. The higher the frequency, the more important they were. Otherwise, no words were given extra weight.

Challenges

The main challenge was executing the program in a reasonable amount of time while being fairly accurate (75%). A solution implemented was to represent documents without stop words. This removed more than half of the words. The training and testing documents were also reduced to ⅓ of their words. This was done by dropping the last ⅔ of the words. Doing this significantly reduced the time spent executing.

Finding the probability of the class, P(C), was also an issue. Author 14 may have been overrepresented, as the classifier would always guess 14 if P(C) was the ratio of an author in documents. This issue was handled by making P(C) = 1/15.

Weaknesses

There are many weaknesses with the methods used here. First off, blindly removing ⅔ of the words may cause issues if the last ⅔ contained more distinctive words for the authors. Perhaps it would be better to search for words that are used commonly among all the authors, and remove those instead.

Also, the likelihood of every author occurring is assumed to be the same. This can have issues if the training set featured one obscure author who would never be seen again. Applying a negative weight to more rare authors may be a solution to this issue.