Lab2 – CSE 5243

James Lowrey

In this lab two different classifiers, K Nearest Neighbors (KNN) and Decision Trees, were used to predict class labels of Lab 1 feature vectors. The feature vectors contained info about every document in the corpus, class labels were defined as the document 'topics' tag(s), and the training data was filtered words in the document corresponding to TF-IDF data. Two feature vectors were used, a larger one with approximately 600 words, and a smaller one with approximately 200 words. Classifiers were built using Python's default Sci-Kit Learn libraries.

Accuracy testing was completed using Cross Validation methods. Each dataset contained the entire corpus of documents; for cross validation the corpus was broke into k subsets with k-1 used for training and 1 used for testing. Subsets rotated until every subset had been used for testing and training. In this lab, k was equal to 4 subsets. Accuracy was defined as the number of correct class predictions divided by the number of incorrect class predictions, summed with the number of missed class predictions, summed with the number of correct class prediction. These values were summed over the entire testing suite to result in final accuracy.

**Example Class Prediction Accuracy**:

correct: [1,5,10,12]

prediction: [1,2]

accuracy = 1/5

Both classifiers were found to be highly inaccurate at about 28.6% correct predictions. However, this is more likely due to poor feature selection than it is to classifier error. Documents were vectorized for training by first finding all unique words in the dataset. Next a corresponding list was filled in where an individual document's TF-IDF score would be copied over for each matching word and zeros would be copied if the word was not found in the document.

**Accuracy**

|  |  |  |
| --- | --- | --- |
|  | KNN | Decision Tree |
| Large TF-IDF dataset | 28.60651% | 28.67070% |
| Small TF-IDF dataset | 28.60572% | 28.62406% |

Using this method with a TF-IDF metric resulted in a sparse dataset. TF-IDF aims to select for words that are important to one document over others, making it likely that the word is unique or at least uncommon across other parsed and filtered documents' data. This hypothesis is supported by the fact that using a larger dataset did not improve accuracy much (the additional words were still document unique, and thus did not help train the classifier), and that the number of words used in cross-validation subset training widely varied depending upon the documents input (subsets were created from sections of documents, if words are unique to documents then their distribution will be skewed since there are many more documents than words in the final dataset).

There are a few methods that could be employed to improve accuracy. The first goal should be to increase training data density (reduce sparsity). Using a sparse dataset results in training data that is very helpful in some cases but useless for most. Currently the “best”, or highest/top TF-IDF values are used for feature vector selection. A better method may be to use TF-IDF that are near the median, ensuring that the words are found across most documents yet still eliminating noise (like stop words). A dense dataset would have the ancillary advantage of improving KNN running efficiency. Another method to improve accuracy would be to better tune the classifier parameters. Currently default values are used and the Python libraries are left to figure out what is best for the dataset. Running a series of experiments with varying parameter values on the dataset could yield better results.

The decision tree classifiers were found to be slower offline (during model construction), faster online (during query prediction), and slightly more accurate. The online cost difference was quite dramatic with KNN taking two orders of magnitude longer per prediction. This is likely due to the sparse dataset, with it KNN requires whole iteration for prediction while decision trees do not. Model construction time for these relatively small number of words took the decision tree classifiers about twice as long, but as number of words increased decision tree construction time increased faster than KNN.