Urvi Patel
Assignment #6
CSC 480
June 1, 2018

The were a few approaches taken to changing the given trading set. The first was try calculating tf*idf for each word in the document and using the scores to find the top words in the documents. Data vector values would then correspond to the tf x idf scores for the top words for each paragraph. This was a very computationally expensive approach and it took about 6 hours to create the arff file for this dataset so this approach was abandoned.

Another approach was found in a posting on SO* that mentioned a study where the text was divided (roughly) into chapters a list of the top 70 words were used as attributes. To create the vector, the values would correspond to the number of times each of of the 70 words was ecountered in each of the chapters in the entire corpus. All trials shown below reflect the use of the J48 algorithm with 4-fold cross-validation.

This resulted in the following scores for the given dataset:

Statistics:

Correctly Classified Instances: 82.6087%
Precision - .828
Recall - .826
F-Measure - .826

Files:

AARF: chap_freq_count_4_fold_J48.arff:
Python: chap_freq.py
Results: chap_freq_count_4_fold_J48

The next approach consisted of keeping the document size as chapters but then instead of counting the number of times the most common words was seen in a chapter, a binary value of 1 or 0 was assigned, corresponding to whether a word was present in the chapter or not. This approach, which was much simpler yielded similar results to that shown above.

This resulted in the following scores for the given dataset

Statistics:

Correctly Classified Instances: 82.1739%
Precision - .826
Recall - .822
F-Measure - .823

Files:

AARF: chap_freq_binary_4_fold_J48.arff:
Python: chap_freq_binary_values.py
Results: chap_freq_binary_4_fold_J48

This time the approach was to use the example given in class wherein the top words are chosen then a binary value of 1 or 0 was assigned, corresponding to whether a word was present in the paragraph or not. However, this time, instead simply assigning a 1 or 0, the value assigned was the count of that word in a paragraph. This yielded results similar to the dataset given in class and was significantly worse than the 2 approaches given above.

This resulted in the following scores for the given dataset

Statistics

Correctly Classified Instances: 57.4176%
Precision - .584
Recall - .574
F-Measure - .566

Files:

AARF: para_freq_count_4_fold_J48:.arff:
Python: paragraph_freq.py
Results: para_freq_count_4_fold_J48:

The above approaches were used with the Multilayer Perceptron algorithm which produced excellent results when the document size was a chapter. With this algorithm, assigning vectors a 1 or 0 depending on whether the top words were in the chapter gave better results than counting the frequencies. For the Paragraph Frequency Count, however, the building of the model was not complete after 10 hours so that run was killed and no data could be collected.

Chapter Frequency Count

Correctly Classified Instances: 98.2609%
Precision - .983
Recall - .983
F-Measure - .983

Chapter Binary Count

Correctly Classified Instances: 98.6957%

Precision - .987
Recall - .987
F-Measure - .987

It would appear that using chapters instead of paragraphs as the document size greatly improved the classfication rate. This is not very surprising since this approach gives the algorithms more text from which to make an assessment. It was interesting however, that merely denoting the absence or presence of a word in a chapter yielded comparable results to actually counting the number of occurences of a word in a chapter. This would lead one to believe that sample size (i.e., paragraph vs. chapter) is the overwhelming factor in the success of the classification process.

* https://stackoverflow.com/questions/4771293/can-an-authors-unique-literary-style-be-used-to-identify-him-her-as-the-autho

*** Notes: In order to split the Dickens set into chapters, the chapters were named Chapter I, Chapter III, Chapter III instead of just I, II, III to avoid dealing with roman numerals. Also, this was done a Mac so to run the code that generates the .arff file, path names should be changed as needed.