CSE 498 Text Mining

Fall 2016

Project 1

Sachin Joshi

Task 1

The extracted reviews of many products in the category of musical instruments from Amazon are stored in the text file named **review.txt** and the rating of each respective review is stored in the text file **rating.txt**. This information is extracted from the dataset **Musical_Instruments_5.json** with the help of the function that has been already provided. The complete reviews along with their corresponding ratings are stored in the file named **reviews.csv**.

I have used R programming language to partition the data into two subsets namely training and test datasets and to discretize the ratings into class 1 and class 0 based on the ratings of the particular review. The R file named **hw.R** contains the code for the above mentioned tasks. We need to install the R programming environment so as to execute this file. However the file can be opened in any of the text editors such as Notepad++ in order to review the code. After running the program, the desired results have been categorized into 4 different files:

1. **NegativeReviews.csv**: This file contains those reviews that have a negative rating (1-3) and belong to class 0.
2. **PositiveReviews.csv**: This file contains those reviews that have a positive rating (4-5) and belong to class 1.
3. **TrainingDataset.csv**: This file contains randomly partitioned data that contribute to approximately 80% of the entire dataset and belongs to the training dataset.
4. **TestingDataset.csv**: This file contains randomly partitioned data that contribute to approximately 20% of the entire dataset and belongs to the testing dataset.

The Naive Bayes categorization system predicts whether the reviews of the products in the category of musical instruments from Amazon are positive or negative. It classifies strings of words to categories using Bayes rule. I have used Java programming language to classify the reviews into positive and negative categories. The Java file to depict this functionality is **NaiveBayesClassifier.java**. Other supporting Java files are stored in the package **saj415**.

Summary

My implementation is very similar to a mathematical standard Naive Bayes classifier using tokens consisting of one, two, and three-word sequences from the current document. It achieves an accuracy of 82.6% using the provided training and testing data sets, though it performs somewhat better (in the 85% range) when the training and testing data sets are shuffled together

and the training and testing sets are selected randomly but in the same proportions as the original sets.

Working

Baye's rule states that P(x|y), the probability of x occuring given that y has occured, is equal to P(y|x)P(x)/P(y).

Let **s** be the string which we would like to classify. Let **w1,w2,...,wn** be the words in s. Let c be an arbitrary category.

P(c|s) = P(s|c) * P(c) / P(s) = [P(w1|c)P(w2|c)...P(wn|c)] * P(c) / [P(w1)P(w2)...P(wn)]

Our aim is to calculate **P(c|s)** for all categories **c**, and classify **s**, in whichever **c** maximizes **P(c|s)**.

Accuracy and Results

After randomly selecting 80% of the data (training data) to train with, the performance is tested on the remaining 20% of the data (test data). The accuracy is found out to be approximately 83%.

The algorithm produces 82.6% accuracy on the provided testing data set after training on the training set, as well as 96.65% accuracy on the testing dataset. If these sets are shuffled together and new sets with the same proportions are selected, the algorithm performs somewhat better (85% range), indicating that it does not overfit the provided data and has good generality. With regard to time, building the model takes slightly over a second, and classifying all of the testing data takes a little over four fifth of a second.

	Testing Data (20%)	**External Testing Data**	**Training Data Set (80%)**
Accuracy (%)	**83**	**97**	**99**

- **External testing data:** Accuracy is approximately 97%
- **Testing on the training set:** Accuracy > 99%