Naive Bayes Classifier Design & Implémentation

Question#8: [12=6+6 points]

For this question, you will be implementing a naive Bayes classifier. This classifier will be used to classify fortune cookie messages into two classes: messages that predict what will happen in the future and messages that just contain a wise saying. We will label messages that predict what will happen in the future as class 1 and messages that contain a wise saying as class 0. For example,

"you are going to have a pleasant experience"   in Class 1
"speak only well of people and you need never whisper"   in Class 0

Files Provided

You are free to design the code as you wish and to implement the code in any language that you wish. There are three sets of data files provided off the course web page. You will notice that all words in these files are lower case and punctuation has been removed. These files are as follows:

1. The training data:
   - traindata.txt: This is the training data consisting of fortune cookie messages.
   - trainlabels.txt: This file contains the class labels for the training data.
2. The testing data:
   - testdata.txt: This is the testing data consisting of fortune cookie messages.
   - testlabels.txt: This file contains the class labels for the testing data. These are only used to determine the accuracy of the classifier.
3. A list of stop words: stoplist.txt

Code Requirements

Your code will take the following files as its inputs:
< stoplist file > < training data file > < training label file > < testing data file > < testing label file >

There are two steps to this assignment: the pre-processing step and the classification step.

Pre-processing step

This first step converts fortune cookie messages into features to be used by a naive Bayes classifier. You will be using the bag of words approached described in class. The following steps outline the process involved:

1. Form the vocabulary. The vocabulary consists of the set of all the words that are in the training data with stop words removed (stop words are common, uninformative words such as "a" and "the" that are listed in the file stoplist.txt). The vocabulary will now be the features of your training data. Keep the vocabulary in alphabetical order to help you with debugging your assignment.
2. Now, convert the training data into a set of features. Let M be the size of your vocabulary. For each fortune cookie message, you will convert it into a feature vector of size M+1. Each slot in that feature vector takes the value of 0 or 1. For the first M slots, if the ith slot is 1, it means that the ith word in the vocabulary is present in the fortune cookie message; otherwise, if it is 0, then the ith word is not present in the message. Most of the first M feature vector slots will be 0. Since you are keeping the vocabulary in alphabetical order, the first feature will be the first word alphabetically in the vocabulary. The (M+1)th slot corresponds to the class label. A 1 in this slot means the message is from class 1 while a 0 in this slot means the message is from class 0. I will refer to the "featurized" training data as the pre-processed training data.
3. Output the pre-processed training data to a file called preprocessed.txt. The first line should contain the words in the vocabulary, separated by commas. The lines that follow the vocabulary words should be the featurized versions of the fortune cookie messages in the training data, with the features separated by commas. Your file should look something like:

   ```
   a, aardvark, almost , anticipate,...
   0,0,1,0,...
   0,1,0,1,...
   ```

   Note that even though I am getting you to output the training data, the classification step should happen right after the pre-processing step (just pass the preprocessed data directly to your classifier. Don't save it out and reload it back in).

Classification step

Build a naive Bayes classifier as described in class.

1. In the first phase, which is the training phase, the naive Bayes classifier reads in the training data along with the training labels and learns the parameters used by the classifier
2. In the testing phase, the trained naive Bayes classifier classifies the data in the testing data file. You will need to convert the fortune cookie messages in the testing data into a feature vector, just like in the training data where a 1 in the ith slot indicates the presence of the ith word in the vocabulary while a 0 indicates the absence. If you encounter a word in the testing data that is not present in your vocabulary, ignore that word. Note that the feature vector is only of size M because the class labels are not part of the testing data.

3. Output the accuracy of the naive Bayes classifier by comparing the predicted class label of each message in the testing data to the actual class label. The accuracy is the number of correct predictions divided by the total number of predictions.

A couple of hints:

- When reading words from the stop list and from the training/testing data, you may need to trim any extra white space.
- Make sure that you follow the implementation hints given in the lecture. Specifically, do the probability calculations in log space to prevent numerical instability.

Results

Your results must be stored in a file called results.txt.

1. Run your classifier by training on traindata.txt and trainlabels.txt then testing on traindata.txt and trainlabels.txt. Report the accuracy in results.txt (along with a comment saying what files you used for the training and testing data). In this situation, you are training and testing on the same data. This is a sanity check: your accuracy should be very high ie. > 90%
2. Run your classifier by training on traindata.txt and trainlabels.txt then testing on testdata.txt and testlabels.txt. Report the accuracy in results.txt (along with a comment saying what files you used for the training and testing data). We will not be letting you know beforehand what your performance on the test set should be.

How the assignment will be marked

- Pre-processing step (6 points)
- Naive Bayes classifier (6 points)

Your code will be executed by our marker, so please make sure it will run correctly and please provide enough information in a readme file for our marker.

Acknowledgements & References

This interesting “Naive Bayes Classifier assignment” originally and entirely was designed by Prof. Weng-Keen Wong, at College of Engineering, Oregon State University, thanks to him, also Thanks to Kristin Stubbs for providing the original fortune cookie data set.