Homework #3: Sentiment

Deadline: 11:59pm Sunday 27 October 2013

Please present a report with all your answers, explanations, and sample images or plots. Submit also a soft copy of the source code and binaries used to generate these results. Please note that copying of any results or source code will result in ZERO credit for the whole homework.

Acknowledgment: This homework is adapted from Chris Manning and Dan Jurafsky's Coursera NLP class from 2012.

Your goal for this homework is to perform **Sentiment Analysis**: classifying movie reviews as positive or negative. Recall from the lecture that sentiment analysis can be used to extract people's opinions about all sorts of things (congressional debates, presidential speeches, reviews, blogs) and at many levels of granularity (the sentence, the paragraph, the entire document). Our goal in this task is to look at an entire movie review and classify it as positive or negative.

Algorithm

You will be using **Naïve Bayes**, following the pseudocode in Manning, Raghavan, and Schütze (page 241 in the paper, offline edition; page 260 in the "pdf for printing" or "pdf for online viewing" version that's online), using Laplace (add-1) smoothing. Your classifier will use words as features, add the logprob scores for each token, and make a binary decision between positive and negative. You will also explore:

1. The effects of stop-word filtering. This means removing common words like "the", "a" and "it" from your train and test sets. We have provided a stop list with the starter code in the file: `data/english.stop`
2. The effects of Boolean Naive Bayes. This means removing duplicate words in each document (review) before training.

The algorithm is a simplified version of this paper:

•**Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan.** 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 79-86.

Assignment

Train a **Naïve Bayes** classifier on the **imdb1** data set provided with the starter code. This is the actual Internet Movie Database review data used in the original Pang and Lee paper. The starter code comes already set up for 10-fold cross-validation training and testing on this data. Recall that cross-validation involves dividing the data into several sections (10 in this case), then training and testing the classifier repeatedly, with a different section as the held-out test set each time. Your final accuracy is the average of the 10 runs. When using a movie review for training, you use the fact that it is positive or negative (the hand-labeled "true class") to help compute the correct statistics. But when the same review is used for testing, you only use this label to compute your accuracy. The data comes with the cross-validation sections; they are defined in the file:

```
data/poldata.README.2.0
```

Your first task is to implement the classifier training and testing code and evaluate them using the cross-validation mechanism. Next, evaluate your model again with the stop words removed. Does this approach affect average accuracy (for the current given data set)?

Where to Make Your Changes

You need to make your changes in at least three functions:

- `addExample()`: which adds a new training example.
- `classify()`: which classifies a test example.
- `filterStopWords()`: which implements stop-word filtering.

Evaluation

Your classifier will be evaluated **imdb1** mentioned above, once with and once without invoking stop-word filtering.

Running the code

```
$ cd python

$ python NaiveBayes.py [-f] ../data/imdb1
```

or press **F5** from Python(x,y) (see default parameters in main()). This will train the language models for each cross-validation and output their performance. Adding a *-f* flag invokes the stop-word filtering.

If you're curious how your classifier performs on separate training and test data sets, you can specify a second directory, in which case the program should train on the entirety of the first set (i.e., without cross-validation) then classify the entire held-out second set.

```
$ python NaiveBayes.py (-f) train test
```

Requirements

You are required to implement:

1. Naive Bayes classifier, and specifically the functions mentioned above.
2. Boolean Naive Bayes classifier.
3. Experiment with/without using stop words.

You are required to obtain an average performance of at least 80% for the normal Naive Bayes. Please include in your report a printout of your runs with/without stop word filtering, and mention if the Boolean Naive Bayes improves your performance or not.

Please submit your code and report in one zip file, named **CMPN463.HW03.firstname.lastname.zip**. For example, if your name is Mohamed Aly, your file should be named **CMPN463.HW03.Mohamed.Aly.zip**.

Grading

4 pts: implementation of Naive Bayes
2 pts: implementation of Boolean Naive Bayes
1 pt: implementation of stop word filtering
2 pts: report and submission file name
1 pts: performance better than 80%.