Homework 3: Sentiment Analysis [starter code]

Due: Tuesday Apr 10 at 11:59pm

Your goal for this homework is to perform **Sentiment Analysis**: classifying movie reviews as positive or negative. Recall from 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 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 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:

```
sentiment/data/english.stop
```

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:

```
sentiment/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

To ensure that your code works properly not only when run from the command line but also when executed via the submit process, you should limit your changes to addExample() and classify() since these are the hooks that Submit uses to exercise your code. You're free to add other elements further invoked from addExample() or classify(), but note well that main() is **NOT** executed by Submit so you cannot rely on anything added there.

Changes beyond addExample() and classify(), if you choose to make any, should be done with caution. For java specifically, Submit.java makes calls to getTrainSplits(), train(), readTest(), and evaluate(). Submit also relies on the definition of TrainSplit and directly manipulates boolean FILTER_STOP_WORDS. Note that classify() is called both directly by Submit.java and also from within evaluate(). (As well as from main(), of course.)

For python, Submit.py calls the following: crossValidationSplits(), trainSplit(), train(), test(). Submit also relies on the definitions of classes TrainSplit and Example and directly manipulates boolean FILTER_STOP_WORDS.

Evaluation

Your classifier will be evaluated on two different data sets—both **imdb1** mentioned above and a second held-out test set—and each of these is then in turn evaluated twice, once with and once without invoking stop-word filtering.

Running the code

For Java, execute

```
$ cd java
$ mkdir classes
$ javac -d classes -target 1.6 *.java
$ java -cp classes NaiveBayes ../data/imdb1
```

This will train the language models for each cross-validation and output their performance.

Note that the directory argument allows you to train on other data sets in the process of your development, but for submission, the Submit.java script expects to be able to find the imdb1 directory in its default location noted here.

Adding a *-f* flag...

```
$ java -cp classes NaiveBayes -f ../data/imdb1
```

...invokes the stop-word filering.

Similarly, for python, execute

```
$ cd python
$ python NaiveBayes.py (-f) ../data/imdb1
```

If you're curious how your classifier peforms 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.

```
$ java -cp classes NaiveBayes (-f) train test
```

or

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

Submitting Your Solution

For each language, we have provided a submission script/program which will run your program and classify both **imdb1** (via 10-fold cross-validation) and a second unseen data set, both with and without stop-word filtering. Your classifier's decisions are sent to our servers where they are stored and graded once the assignment is due. For each language, the submission program will programmatically call your script and then prompt you for your username and a password. The password can be found at the top of the **Assignments** page on this website. It will also ask you about which parts of the assignment you wish to submit. For this assignment your system will be evaluated in four parts: each of the two data sets both with and without stop-word filtering.

Here are the basic shell commands which you will need to execute for Java

```
$ cd java
$ mkdir classes
$ javac -d classes *.java
$ java -cp classes Submit
```

and for Python

```
$cd python
$python submit.py
```

It is important that these scripts be executed from either the java or python directories so that they can easily import the code you've written, but also—and this is critical—so that the relative path "../data/" correctly points to the training and development data (it is hard-coded into the submission scripts).