Final Lab Option #1 - Sentiment Analysis

Acknowledgment: This homework is adapted from **Mohamed Alaa El-Dien Aly's** work, which was adapted from **Chris Manning and Dan Jurafsky's Coursera NLP class from 2012**.

This lab does not have interim due dates. I strongly suggest that you and your partner make a timeline and stick to it.

Part 1

Deliverables: A modified version of the code file; a report containing copies of all of your results.

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.

Before you start, you should retrieve `lab_e_sent_data` from the `~barbeda/cs365-share` directory. That contains both Python files you will modify and the data you will test with.

Before you change anything, take some time to make notes with your partner about the different components of the provided Python code.

Algorithm

You will be using Naïve Bayes, following the pseudocode in Manning, Raghavan, and Schütze, using Laplace (add-1) smoothing. It can be found here. 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.

You should use the stop-world list found at `data/english.stop`

You will use n-fold cross-validation to test, which 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. 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 it 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 using the imdb1 dataset mentioned above, once with and once without invoking stopword filtering. Running the code

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

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 tes

Requirements

You are required to implement a Naive Bayes classifier, and specifically the functions mentioned above. You are required to obtain an average performance of at least 80%.

Please include in your **report** a printout of your runs with/without stop word filtering in your report.

Part 2

Before starting Part 2, make sure that your report for Part 1 is complete!

In this homework you will be trying several ideas to improve upon the performance of your sentiment analysis from the previous homework. You should explore **at least two of** these ideas and comment on your experiments in your **report**.

1. Using a Boolean vs normal representation for Naive Bayes. With a boolean representation, your vector contains a 0 if the word does not appear and a 1 if it does. With a normal representation, you use the full count. (This is the one you have already used above.)
2. Using an SVM for classification instead of Naive Bayes. You can use the SVM implementation of the sklearn package.
3. Implementing simple negation features. For example, detect some negation words (use Not, n't, and never) and add NOT_ to each word until the next punctuation (e.g. period, comma, question mark, … etc.). You will likely need to use regular expressions to do this.

Requirements: You are required to explore two of the ideas above. Explore these ideas **independently** using the best settings from Part 1. For each idea, implement it and comment (in your report) on whether it improves the performance of Part 1, and by how much. You should also comment briefly on your implementation choices.

I would highly recommend planning out how you will organize your code so that you can test these new ideas without interfering with reproducibility of Part 1. To be safe, you might choose to preserve a copy of the file as it existed after you finished Part 1, and make your edits to a different copy.

Deliverables: Report, plus two additional code files that implement ideas from that list. Your README file should be clear about how to test each of the conditions.

Grading

Part 1: 40 Pts
Part 2: 40 Pts