CS294-1 Behavioral Data Mining Spring 2012

Programming Assignment 1: A Naïve Bayes Classifier for Sentiment

Due: Thursday February 2

The goal of this assignment is to apply the Naïve Bayes classifier we described in lecture 2 to a dataset of labeled textual movie reviews. The reviews originally included numerical scores (-4 to +4), but they have been partitioned into positive and negative sets, and matched in size. The dataset was created by Bo Pang and Lillian Lee at Cornell. You can find it here: http://www.cs.cornell.edu/People/pabo/movie-review-data/ and the dataset you need is "Polarity dataset v2.0"

To evaluate the quality of your classifier, you should do a 10-fold cross-validation and apply an accuracy measure such as F_1 or AUC (lecture 3). You can try either Bernoulli or Multinomial models.

Although our goal here is to build and evaluate classifiers, the naïve Bayes classifier generates log likelihoods for both positive and negative class membership given a test document. The log likelihoods are linear functions of the term frequencies (or 0,1 values for Bernoulli) of the test document and the weights can be thought of as sentiment estimates for particular words. We'll develop more accurate models in future, but for now those term weight vectors can be used to do sentiment analysis on other documents, such as twitter statuses.

Using ScalaNLP

We strongly recommended that you use Scala and the ScalaNLP toolkit by David Hall and Daniel Ramage. While there are other tools that can do this, Scala has a lot of advantages and we'll be linking all the tools we use into the Scala environment. http://www.scalanlp.org/ and http://www.scala-lang.org/node/197 . Start by tokenizing the text, then build an Index (dictionary) of words, and use the Encoder class to map words to indices to build the data vectors.

Extra Credit

You can often improve the accuracy of a text-processing algorithm by **stemming** (canonicalizing word suffixes like "swimming → swim") or removing **stopwords**, like "the," "a," "of" etc.

You may get some significant gains by processing n-grams (consecutive sequences of n words) as well as words (which are "unigrams"). N-grams make negations and adjectives visible to the classifier, e.g. "no good" or "really entertaining" whereas "no" and "really" have little value by themselves. n should be small (say 2 to 4). On the other hand, using n-grams introduces very strong dependencies between the n-gram features and their subsequences, which NB is not built to handle. It will be interesting to see which way this goes.

You can also try a POS (Part-Of-Speech) tagger to distinguish adjectives and filter out other words.

How to submit We're working on that and will let you know.

This assignment borrows from Stanford CS424P "Extracting Social Meaning and Sentiment" by Dan Jurafsky and Chris Potts