# Jaideep Whabi

A20403110

Step 1:

Initially, the work involves parsing of the data. Using imdb.py and vocab.py we obtain 2 matrices. X matrix is a document term matrix, y is a matrix of class labels for each document. Use this to build a dictionary of words with unique id's , frequency tables, and document classes.

Step 2:

The Train function is used to calculate model paramenters like total reviews, total positive words, total word frequency for each class, etc which will be used later to calculate probabilities.

Step 3:

The Eval function is used to test the model. PredLabel is called with test data to evaluate predictions.

P(c given w) is calculated by using a logarithmic sum of P(w given c) and P(c). These are calculated by using the parameters calculated in the Train function. Based on the positive sum score and negative sum score and a threshold, A label is assigned by comparing the probability to a user defined threshold and calculate accuracy, precision and recall.

Eval additionally calculates Accuracy, Recall and Precision along with a precision vs recall curve for the test data.

Step 4:

PredictProb is used to obtain the predicted labels, true label and documents for 20 documents from the test data.

Step 5:

Next we obtain the Top 20 positive or negative words. We do this by using Logoddsof every word in each class and generate the difference of these logodds for each word and class combination. (We calculate the difference between the P(w/c) for both the categories.) The extremely positively values are the words for the positive label and the extremely negative ones are for the negative label. By doing this we can eliminate the stop words, because this difference in values for both the classes will be very small.

**Output:**

**Command:** !python NaiveBayes.py "C:/Users/jaide/Documents/NLP/Assignment 1/data/data/aclImdb" 1.0

**The Precision vs Recall shows a negative curve suggesting decreasing Recall as we increase Precision.**

Additionally, for multiple iterations with different values of ALPHA, the following values of accuracy were found:

ALPHA accuracy

0.1 80.9

0.5 81.8

1. 82.2

"Significant Output for threshold 0.5 and alpha=1"

Reading Training Data

Reading Test Data

Computing Parameters

Evaluating

For Positive Class:

Test Accuracy: 0.82284

Test Recall: 0.77192

Test Precision: 0.859445978445

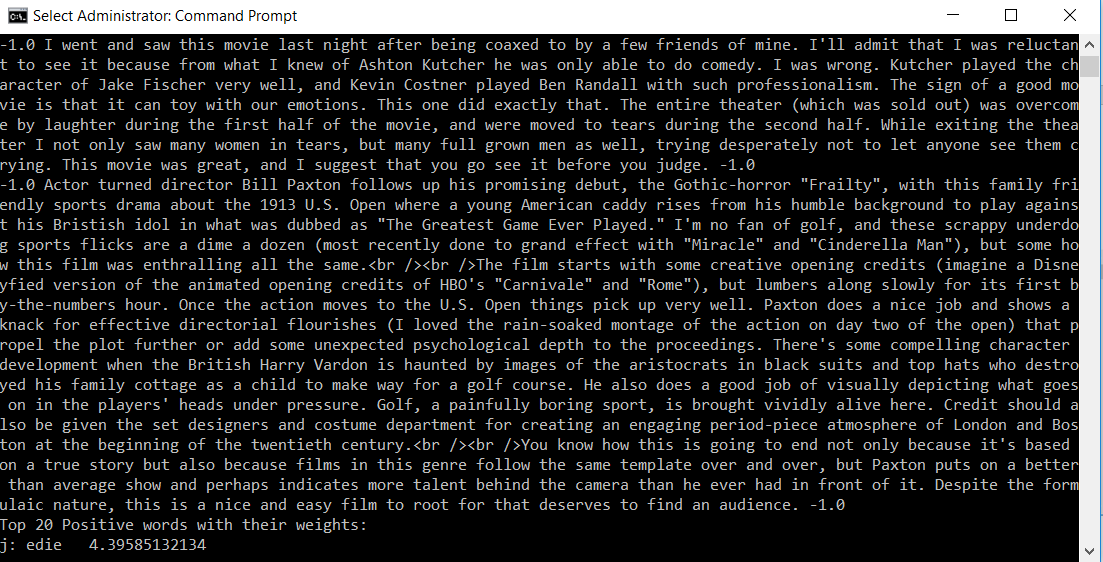
For Negative Class:

Test Accuracy: 0.82284

Test Recall: 0.87376

Test Precision: 0.793000798664





Top 20 Positive words with their weights:

j: edie 4.39585132134

j: gundam 4.3208161354

j: antwone 4.10414509859

j: yokai 3.84821172445

j: />8/10 3.84821172445

j: gunga 3.82715831525

j: />7/10 3.82715831525

j: />10/10 3.80565211003

j: gypo 3.78367320332

j: din 3.78367320332

j: othello 3.73821082924

j: 7/10. 3.61459687327

j: tsui 3.560529652

j: paulie 3.54654341003

j: blandings 3.53235877503

j: goldsworthy 3.47351827501

j: gino 3.44274661634

j: kells 3.44274661634

j: />9/10 3.44274661634

j: harilal 3.41099791803

Top 20 Negative words with their weights:

j: />4/10 -4.06604055429

j: seagal -4.05722992461

j: 2/10 -3.91480958457

j: boll -3.9045530844

j: uwe -3.89419029736

j: \*1/2 -3.85163068294

j: unwatchable. -3.82965177623

j: thunderbirds -3.76065890474

j: />3/10 -3.73656135316

j: gamera -3.73656135316

j: 4/10 -3.67364752775

j: wayans -3.6339071991

j: awful! -3.57833734794

j: slater -3.48872518925

j: />avoid -3.48872518925

j: tashan -3.45697649094

j: segal -3.45697649094

j: drivel. -3.45697649094

j: aztec -3.42418666812

j: kareena -3.42418666812

