Brent Kirkland
Machine Problem 1

Architecture

The code is split into three parts: **data input, data handling, model handling.** The code was written in python 2.7.

Data Input

To ensure we create an accurate model, the code splits the training set into a smaller training set and a validation set. The training file is split randomly to limit bias in the validation file. Next the program reads in the new training file, validation file, and the testing file. The code then takes each of these files and split them by line and word, counts words, and creates bigrams.

Data Processing

Next the code removes unimportant words by checking which removed words from the training set yield the best accuracy on the validation set. This takes sometime as it iterates through many possibilities. This creates our model.

Model Handling

Now that the code thinks it has sustainable parameters it checks them on the testing set. If the testing set performs poorly, the parameters are recalculated.

Preprocessing

To clean the movie removes first the code makes all words uppercase. This will make words with different cases be the same. The code also removes all punctuation. It also removes any html syntax. Once the document is cleaned the data is stored in a dictionary. In the dictionary is a total word count, all the words, each line, all words from the line, word count of the line, reviews split by rating, each word from each rating, word count from each rating.

Model Building

As mentioned in the Architecture. The model starts by looking for words that are not influential and removing them from the calculation. Once it finds a model that best represents the validation set, it cross validates on the test set. If the testing set performs poorly, then we recalculate the parameters on the validation set and repeat.

Results

The running time of my algorithm if it does not have to recalculate the validation set is about 100 seconds. If the validation set needs to be recalculated then it is n*100 seconds where n is the number of times. The average accuracy of my algorithm is about .868 to .878. This also depends on the randomness of my validation set.

My most influential features for positive rating was:
MILDRED, EDIE, HAYWORTH, OFKELLS, NATALI, LUE, SELMA

For negative ratings:
WASTEOF, REDEEMING, SEAGAL, MONEYON, ANDDULL, SENTENCES

Surprisingly many of these negative ones were bigrams.

These are outliers, but obvious words like bad, brilliant, good, and terrible made impact too.

Challenges

The initial challenges I had were related to representing the data only with unigrams. Unigrams made the algorithm perform at best .845. When I transitioned to unigrams and bigrams, the model accuracy increased greatly.

Weaknesses

The weakness in this model relies on the size and randomness of the validation set. If the validation set does not accurately represent the training set, a skewed model is created. To overcome this problem, the easiest (but time consuming way) to fix this to recalculate the validation set.