Noble Menchaca

movieReviewer

Objectives – This is a document classification program using a machine learning algorithm. To complete this project my goal was to find a corpus and see if I could develop features to distinguish the ratings products were given on a 5 point scale.

Methods – I found Julian McAuley's corpus online and instead of using the full 20 GB package on his website, ultimately I switched to the smallest available slice of the corpus, a collection of musical instrument reviews spanning about 10,000 samples. These were stored in Gzip format.

The next step was getting the reviews unzipped as dictionaries into a list that could be called. Then to build a string of all the words in the corpus that could then be lemmatized using the built in NLTK function. This was to use the frequency distribution in order to find the most common words.

The feature design was the part that I was most looking forward to. In class we had some fun looking at what words suggested high or low ratings, but I wasn't satisfied at stopping there. In my proposal I mentioned looking at lexicon size of the reviewer as well as redundancy.

After the errors I was receiving I realized that empty text reviews were killing my code. I put in a safe_divide function finally saw results.

To analyze the results I implemented NLTK's naïve Bayes classifier and performance functions. The results were not terrible, about .652 accuracy. But according to the most_informative_features print out, there was a lot of overlap in using binary features to predict ratings.

```
Most Informative Features
            redundant_2+ = True              3.0 : 5.0    =      3.5 : 1.0
        redundant_1.75-2 = True              2.0 : 5.0    =      3.2 : 1.0
        redundant_1-1.25 = True              1.0 : 2.0    =      2.2 : 1.0
     redundant_1.25-1.5 = True               2.0 : 1.0    =      2.1 : 1.0
     redundant_1.5-1.75 = True               1.0 : 3.0    =      2.0 : 1.0
         No Review Text = True               3.0 : 4.0    =      1.8 : 1.0
             vocab_100+ = True               2.0 : 5.0    =      1.8 : 1.0
              vocab_20- = True               1.0 : 4.0    =      1.7 : 1.0
     redundant_1.25-1.5 = None               1.0 : 2.0    =      1.4 : 1.0
            vocab_20-50 = True               5.0 : 3.0    =      1.4 : 1.0
        redundant_1-1.25 = None              2.0 : 1.0    =      1.4 : 1.0
             vocab_20-50 = None              3.0 : 5.0    =      1.4 : 1.0
           vocab_50-100 = True               2.0 : 5.0    =      1.3 : 1.0
     redundant_1.5-1.75 = None               3.0 : 1.0    =      1.2 : 1.0
           vocab_50-100 = None               5.0 : 2.0    =      1.1 : 1.0
        redundant_1.75-2 = None              5.0 : 2.0    =      1.1 : 1.0
             vocab_100+ = None               5.0 : 2.0    =      1.1 : 1.0
              vocab_20- = None               2.0 : 1.0    =      1.1 : 1.0
            redundant_2+ = None              1.0 : 3.0    =      1.1 : 1.0
          No Review Text = None              4.0 : 3.0    =      1.0 : 1.0
```

After noticing this, I decided to remove extraneous features. This upped the accuracy to .674, and out of a 5 star rating system, is pretty good, over 3x better than chance. This is what I left the final code as. The full output is listed below.

Results

```
Loading corpus into dictionaries...
Loading searchWords from frequency distribution...
Parsing reviews into tuples...
Making feature sets using searchWords...
Size of trainSet: 2000
Size of testSet: 8261
Training...

Accuracy on test set: 0.6744946132429488

Most Informative Features
contains((';', 1957)) = False                3.0 : 5.0    =      1.0 : 1.0
contains(('many', 758)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('bright', 357)) = False            3.0 : 5.0    =      1.0 : 1.0
contains(('expect', 273)) = False            3.0 : 5.0    =      1.0 : 1.0
contains(('seems', 598)) = False             3.0 : 5.0    =      1.0 : 1.0
contains(('34', 1156)) = False               3.0 : 5.0    =      1.0 : 1.0
contains(('high', 721)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('fret', 326)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('line', 437)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('took', 231)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('wrong', 302)) = False             3.0 : 5.0    =      1.0 : 1.0
contains(('drum', 234)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('after', 197)) = False             3.0 : 5.0    =      1.0 : 1.0
contains(('exactly', 292)) = False           3.0 : 5.0    =      1.0 : 1.0
contains(('lock', 335)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('give', 859)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('material', 224)) = False          3.0 : 5.0    =      1.0 : 1.0
contains(('!', 4234)) = False                3.0 : 5.0    =      1.0 : 1.0
contains(('chord', 201)) = False             3.0 : 5.0    =      1.0 : 1.0
contains(('regular', 205)) = False           3.0 : 5.0    =      1.0 : 1.0
contains(('side', 434)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('front', 210)) = False             3.0 : 5.0    =      1.0 : 1.0
contains(('volume', 629)) = False            3.0 : 5.0    =      1.0 : 1.0
contains(('cost', 461)) = False              3.0 : 5.0    =      1.0 : 1.0
contains(('display', 209)) = False           3.0 : 5.0    =      1.0 : 1.0
None
```