Naive Bayes Classifier (HW2)

Introduction:

This project examines the creation and tuning of a Naive-Bayes Classifier that trains on movie review (and other review) logs collected from www.rateitall.com. The movies are now a few years old, but the approach would be the same regardless of the the movie vintage. There are two main parts to the project:

- The ***first phase*** creates a simple Naive-Bayes classifier that trains on the frequency of each single word (unigram) in the positive movie reviews (reviews receiving 5 out of 5 stars), and the negative reviews (reviews receiving only 1 out of 5 stars). After training, an input sentence is classified using "unigram" classification, which compares the frequency of each word in the input sentence against the frequencies of the same word in the positive and negative reviews. If the words appear more often in the positive movie reviews, then the input sentence is classified as "positive", and if the words appear more often in the negative movie review, then the review is classified as "negative". If it is too close to call, the input sentence is deemed to be "Neutral". Code for this phase is included in **bayes.py**.

- The ***second phase*** attempts to improve upon on this classifier, taking into account various weaknesses observed in the initial classification, including changes to bias and additional weighting factors for punctuation and review length. Code for the second phase is included in **bayesbest.py**.

Initial Performance:

After writing the initial bayes.py, the first thing I checked was the scenario of BOTH training AND classifying on the same 'movie_reviews/' data. The expectation was that the training accuracy would be very high. In fact, it turned out to be just okay, but not great.

Results Summary:
- positive: 9567
- neutral: 527
- negative: 3770

Accuracy:
- CORRECTLY CLASSIFIED: 12002
- INCORRECTLY CLASSIFIED: 1862
- **86.6% CORRECTLY CLASSIFIED**

Next, I tried checking the scenario of training on 'movie_reviews/' data and then classifying the 'db_txt_files/' data to see how effective the training was. It turned out to be not very effective at all.

Results Summary:
- positive: 48216
- neutral: 8612
- negative: 70372

Accuracy:

- CORRECTLY CLASSIFIED: 58008
- INCORRECTLY CLASSIFIED: 69192
- **ONLY 45.6% CORRECTLY CLASSIFIED**

NOTE: I also tried training on a subset of 'movie_reviews/' data and classifying on a separate unique subset of ‘movie_reviews/‘. This test yielded similar results as the above, which although slightly better, was still barely over 50% correctly classified -- which is garbage.

Reflections:

I examined a few sentences to try to understand some of the causes of the mis-classification. The following sentences were incorrectly classified based on the **bayes.py** classifier trained on the full 'movie_reviews/' data set:

1. "*I hate this class!*"

2. "*Rob your grandma's purse if you have to and go rent this movie!*"

3. "*Tons of action and lots of humor. Lots of gore and language, so if that turns you off then don't waste your time, but if it's action and comedy you prefer, then pick this one up.*"

Sentence 1 was inspired by the fact that "I love this class!" returned a positive classification. I immediately expected "I hate this class!" to return a negative classification, but unfortunately it returned positive, which is clearly wrong since "hate" cannot be misconstrued as a positive thing. I examined the word frequencies of each word in each of these sentences and found two significant reasons that this sentence failed in classification. The first reason is a bias based on the "prior" probabilities in the training data, and the second reason is sentence length. Given the significant majority of ***positive*** reviews in the training data, the "prior" probabilities are skewed towards to the ***positive***. A short sentence like this one just doesn't have enough words (or enough strongly correlated negative words) to overcome the initial positive bias. In this case, the words "I", "hate", and "this" all ranked slightly negative, and "class" and "!" both ranked slightly positive. Since there wasn't enough weight in the negative direction to overcome the prior probability bias, the final classification returned positive, incorrectly.

NOTE: Interestingly, although the word "hate" wasn't strong enough to prevent the misclassification as “positive", just by replacing the word "hate" with “hated”, the classifier returned a “neutral" classification, which still is wrong but is starting to head in the right direction. I believe that since the training data is a series of reviews, verbs in the past tense are more likely to register strongly in the classifier than present tense verbs.

Sentence 2 was pulled directly from one of the positive movie reviews. The meaning of the sentence is that the movie is so good, one should see it by any means necessary. However, this sentence gets classified incorrectly as "negative". Looking at the individual word frequencies, it become apparent that certain ways of human communication including sarcasm can lead to an incorrect classification. The sentence length is long enough to overcome prior probability bias, but in this case there are only 3 words that ranked positive ("you", "and", "!"), and the remaining words either ranked negative or neutral. In particular, the words "rob", "grandma's", and "rent" ranked the most strongly negative. One can easily imagine how these words would be used in negative reviews. I expect that many of the reviews that were mis-classified fall into this category of containing an expression with some level of sarcasm, or they

attempt to convey a positive or negative message, but use a unique expression that contains a few strong words that are more closely linked to the opposite class.

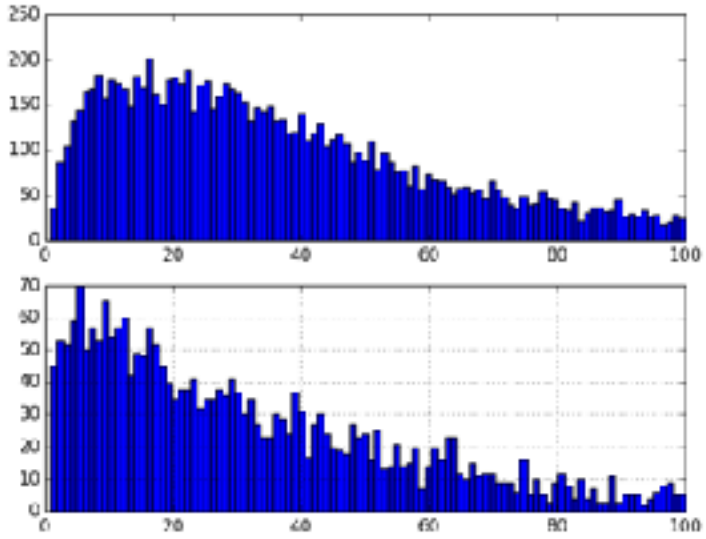
Sentence 3 was also pulled directly from one of the positive movie reviews. In this case, the sentence length is long enough to overcome prior probability bias, and the number of positively ranked words (15) and negatively ranked words (14) are pretty close (with the remaining words being fairly neutral). However, this review includes one particular word, "waste", which very strongly ranks negative. This single word is strong enough to skew the entire classification to be negative. I believe that this is an artifact of the qualifying nature of many of the reviews. In order to present a detailed review, this writer is attempting to qualify his/her rating by saying who the movie is intended for and who for it isn't intended for. This is confusing for a single-word classifier, which won't take into account the difference in meaning of these different sections of the review -- or any sentence structure in general. I think that the confrontational nature of movie reviews on a public forum like rateitall.com would have driven many humans to write their reviews with qualifiers like this build into them, and that will throw off a simple unigram classifier, like in this case.

NOTE: Two other scenarios in which the classifier seems to perform weakly are: (1) mis-spellings and (2) exaggerated spellings, for example a review that contains one word: "BOOOOOORRRRRRIIIIINNGGGG." This word is very unlikely to be spelled the exact same way as anyone else's review and there are no other words to go by, so effectively the classifier has no data to classify the input by.

Exploration and Improvements:

Sentence Length:

The first area I tried exploring was sentence length. Looking at the movie reviews, I realized that the mean sentence length of the positive reviews was 55 words and the mean sentence review of a bad review was 47 words. Perhaps many writers didn't think it was worth their time to write a long review on a bad movie, or perhaps it is just easier to convey badness of a movie in a few succinct words. Either way, not only was the average different, but the histograms were different too (see the positive reviews on top and the negative reviews on the bottom).



I added a dictionary to the Naive-Bases classifier that tracked the number of reviews with each different sentence length (number of words). I added the conditional probability of the sentence to be classified to also consider the sentence length in addition to the unigrams in the sentence. Unfortunately, I did not see any improvement in classification accuracy. I also tried classifying reviews strictly by movie length, ignoring all unigrams, and that performed worse. At this point, I was ready to move and try another area of exploration.

Neutrality Bias

In my original Naive-Bases classifier, I have a "neutrality bias" term, which was the threshold between the difference of the classification probabilities of a sentence as positive or negative. I originally set this number to 0.2 (in the Log10 Scale, because 10^0.2 = 1.58, and I was looking for a factor around 50% difference between the classification probabilities). I realized quickly that since I was considering a "neutral" classification to be a wrong one, by removing the neutrality bias (setting it to 0.0), I could only improve my accuracy. This was true, but then I reduced the bias even further and found out some more interesting results. The numbers below show the results of training AND testing on the same "movie_reviews/" dataset.

Neutrality Bias:	# Correctly Classified:
0.2	12002
0.0	12263
-0.2	12427
-0.4	12511
-0.6	12556
-0.8	12587
-1.0	12597
-1.2	12610

Beyond -1.2, there wasn't much improvement. I realized that a significant majority of the reviews in the training set were positive, and that multiplying the sum of conditional probabilities by this prior probability was skewing the results to be more positive. However, the test set ALSO has a significant skew in the positive direction, so just by biasing the classifier more in the positive direction, it would get more of the sentences correctly classified.

I did consider this to be sort of a cheat though, because I felt we should not be able to predict what any given future input sentence prior probability would be (for example we might be given a test test of 100% negative reviews), so I did lower my neutrality bias a little, but I kept it above 0.0.

Word Length

Next, I wanted to see what the most common words were in the positive and negative reviews. I was expecting that there would be a lot of meaningless common words (punctuation, articles, prepositions, etc.) that would be in both negative reviews and positive reviews, but that didn't have a real denotation or connotation that would be considered positive or negative, and could thus be removed from the dictionary / database to remove any weird biases that they might have added.

The top 15 common words in positive reviews and in negative reviews are shown below.

MOST COMMON POSITIVE WORDS

.	:	40343
the	:	30590
,	:	21303
and	:	16293
a	:	14835
of	:	13414
i	:	11065
is	:	10894
it	:	10885
movie	:	10760
to	:	9970
this	:	9502
!	:	7820
in	:	7316
was	:	6901

MOST COMMON NEGATIVE WORDS

.	:	9032
the	:	5646
,	:	4451
a	:	2856
and	:	2741
i	:	2629
of	:	2547
this	:	2522
to	:	2409
movie	:	2193
it	:	2103
is	:	1819
was	:	1784
!	:	1515
in	:	1371

I realized that most of these words were the same, but also that they were very small, and so instead of removing specific words, this gave me inspiration to try another filter: word length. I tried removing short words (words with less than # Letters) from the database used to classify and got the following results. The numbers below show the results of training AND testing on the same “movie_reviews/” dataset.

Training (# Letters):	**Classifying (# Letter):**	**# Correctly Classified:**
0	0	12002
0	1	12112
1	1	11973
0	2	12085
1	2	11998
2	2	12005
0	3	11889
3	3	11208

I did not see any significant improvement here, however this did inspire me to try my next two explorations.

Punctuation

Next, I decided to explore punctuation. If a word was punctuation only, I figured it would either more positively or more negatively impact a review. For example, a person might use multiple exclamation points to describe an exciting and good movie. The same person might use question marks to describe a questionable movie choice (i.e. a bad movie). I add a tunable parameter **punctuation weight**, by which I multiplied the conditional probabilities of words that were only punctuation. Using a punctuation weight of 2 improved the number of correct classifications. Using a weight of 10 improved it more. 100 improved it more. I ended up settling on 1000. I tried punctuation weights which were higher still, and they also worked, but I didn't want to go too overboard and I already felt 1000 was significantly high.

Sentence Length

Having found some success, I returned to the idea of sentence length. However, instead of multiplying the product of conditional probabilities of the unigrams in a sentence by the conditional probability of the sentence length (or adding in the Log scale) like I did before, this time, I checked to see if the sentence length was below some threshold number of words, and if so I would then add an **extra short review weight** to the count of the words in the database, which increase its overall conditional probability factor each time it was found in a short review. I figured short reviews were more succinct, contained more telling words, and lessened the chance that the review was qualifying in nature. This made short reviews much more important in my mind.

Turns out I was completely wrong. Scaling the words in short reviews higher reduced me overall classification correctness. So I reversed the scale factor and lessened the weight of words in short reviews. This improved classification accuracy. I think in the end that this might have been the case because longer reviews might have been more carefully thought out by a reviewer and could therefore contain better queues for the classier to pick up on.

The last step I did was to add a **review length weight**. This was a number that I scaled each input sentence to be classified by depending on the number of words in the review. If the review has more words, then it was scaled more likely to be positive. I reasoned that even thought my original effort of multiplying the product of conditional probabilities of the unigrams in a sentence by the conditional probability of the sentence length (or adding in the Log scale) did not work, a simple weighting by review length might work, and after some experimentation, I arrived at a weighting factor of 0.12 times the number of words in the sentence, and saw significant classification accuracy improvement.

Evaluation:

I've run a few evaluations below. The first is both trained **AND** tested on Movie Reviews (13864 test reviews):

BAYES:

- Results Summary:
  - positive: 9567
  - neutral: 527
  - negative: 3770

- POS CORRECTLY CLASSIFIED: 9468
- POS MIS-CLASSIFIED: 1236
- NEG CORRECTLY CLASSIFIED: 2534
- NEG MIS-CLASSIFIED: 99

POSITIVE REVIEWS:
-- PRECISION: 0.989651928504
-- RECALL: 0.884529147982
-- F-MEASURE: 0.934142370875

NEGATIVE REVIEWS:
-- PRECISION: 0.672148541114
-- RECALL: 0.962400303836
-- F-MEASURE: **0.934142370875**

BAYESBEST:

- Results Summary:
  - positive: 12806
  - neutral: 20
  - negative: 1038

- POS CORRECTLY CLASSIFIED: 11085
- POS MIS-CLASSIFIED: 42
- NEG CORRECTLY CLASSIFIED: 996
- NEG MIS-CLASSIFIED: 1721

POSITIVE REVIEWS:
-- PRECISION: 0.865609870373
-- RECALL: 0.996225397681
-- F-MEASURE: 0.926336021393

NEGATIVE REVIEWS:
-- PRECISION: 0.959537572254
-- RECALL: 0.366580787633
-- F-MEASURE: **0.926336021393**

Training and testing on the same data does not provide very meaningful results though, and although the results skewed more positive, the F-measure stayed about the same. I next tried on different training and test sets.

Next, I trained and tested on two unique (small) subsets of Movie Reviews (578 test reviews):

BAYES:

- Results Summary:
  - positive: 156
  - neutral: 68
  - negative: 354

- POS CORRECTLY CLASSIFIED: 154
- POS MIS-CLASSIFIED: 240
- NEG CORRECTLY CLASSIFIED: 114
- NEG MIS-CLASSIFIED: 2

POSITIVE REVIEWS:
-- PRECISION: 0.987179487179
-- RECALL: 0.390862944162
-- F-MEASURE: **0.56**

NEGATIVE REVIEWS:
-- PRECISION: 0.322033898305
-- RECALL: 0.98275862069
-- F-MEASURE: **0.56**

BAYESBEST:

- Results Summary:
  - positive: 528
  - neutral: 1
  - negative: 49

- POS CORRECTLY CLASSIFIED: 439
- POS MIS-CLASSIFIED: 15
- NEG CORRECTLY CLASSIFIED: 34
- NEG MIS-CLASSIFIED: 89

POSITIVE REVIEWS:
-- PRECISION: 0.831439393939
-- RECALL: 0.966960352423
-- F-MEASURE: **0.894093686354**

NEGATIVE REVIEWS:
-- PRECISION: 0.69387755102
-- RECALL: 0.276422764228
-- F-MEASURE: **0.894093686354**

In this case, I noticed a significant improvement in F-measure between the performance of Bayes and BayesBest. Since this test was trained an a separate data set from the test set, I considered these results to be more significant. Next, I wanted to see if the results would apply to training on movie reviews and testing on a completely different set of reviews in db_txt_files (127200 test reviews in total). The results of that are shown below.

BAYES:

- Results Summary:
  - positive: 48216
  - neutral: 8612
  - negative: 70372

- POS CORRECTLY CLASSIFIED: 35625
- POS MIS-CLASSIFIED: 26467
- NEG CORRECTLY CLASSIFIED: 22383
- NEG MIS-CLASSIFIED: 2066

POSITIVE REVIEWS:
-- PRECISION: 0.945185853387
-- RECALL: 0.573745410037
-- F-MEASURE: **0.714049487388**

NEGATIVE REVIEWS:
-- PRECISION: 0.458198567042
-- RECALL: 0.915497566363
-- F-MEASURE: **0.714049487388**

BAYESBEST:

- Results Summary:
  - positive: 118145
  - neutral: 206
  - negative: 8849

- POS CORRECTLY CLASSIFIED: 65278
- POS MIS-CLASSIFIED: 1198
- NEG CORRECTLY CLASSIFIED: 5903
- NEG MIS-CLASSIFIED: 20211

POSITIVE REVIEWS:
-- PRECISION: 0.763583618945
-- RECALL: 0.981978458391
-- F-MEASURE: **0.859118876057**

NEGATIVE REVIEWS:
-- PRECISION: 0.831291367413
-- RECALL: 0.226047330934
-- F-MEASURE: **0.859118876057**

In this case, I noticed an improvement in F-measure, but maybe not as high as in my results on the previous page because the previous test was a small sample size and this test included reviews which were different subject matter than just plain movies.

Future:

Additional improvements I would like to make would be to go back and explore the common words filtering idea outlined above. I think that there are certain words that could be taken out of the test data that add extra skew to the sentiment analysis and are not needed. I would also like to explore training this on even larger data and applying the classification to a test that did not include just sentences of a review-type nature.