CS5785: Applied Machine Learning

Homework #3

James Leung (JL958), Jai Nanavati (JAN266)

**Part I: Sentiment Analysis from Bag of Words**

Abstract

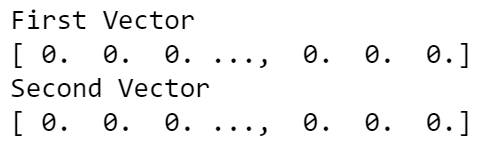
The dataset we used contained 1000 reviews from Yelp, Amazon, and IMDB, each of which had 500 positive and negative reviews. Using a simple bag of words model, we were able to predict positive or negative sentiment with about 81% accuracy, which is much better than random chance (50%). The Naïve Bayes (Bernoulli Assumption) model slightly outperformed the Logistic Regression model, but when PCA is used to simplify the data, Logistic Regression outperformed Naïve Byes. Given that NB is a generative model and LR is discriminative, NB will have higher predictive power with small amount of data, but when we have infinite observations, LR will be better.

**Methods**

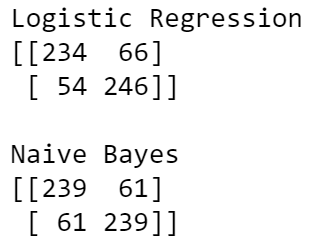
We first created a class called SentenceCleaner to perform our pre-processing. We created functions for the following activities:

* **Stemming**: “Running -> run.” We implemented our own function to check if the stemmed result was a true word or not (with PyEnchant). If the result was not an English word, we reverted the stemming.
* **Split Conjoined Words**: “appletree -> apple tree.” This prevented some unexplainable unique words that appeared as an output. We compiled the nltk’s dictionary of words into a regular expression and identified conjoined words that needed a space.
* **Remove Stopwords**:Remove words that likely carry no meaning: “e.g. the, and, of”
* **Remove punctuation**: self-explanatory

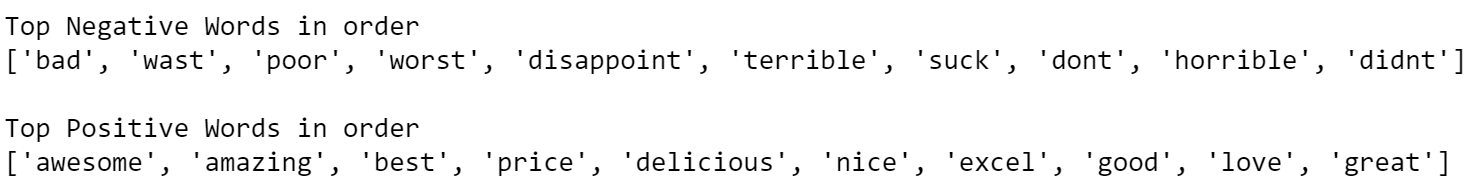
Then, we cleaned the reviews with our sentencecleaner and transformed the list of remaining words into a feature vector (where xi is the number of times wordi appears in the sentence). With basic intuition, we can tell that the vectors were sparse. We then normalized these vectors using l2-normalization in order maintain sparsity. Log-normalization would be preferred if we expected certain word counts to be extreme and we wished to reduce their impact. Here is two reviews as requested:



Finally, we trained a Logistic Regression and a Naïve Bayes classifier on these features with a balanced training set. The accuracy of both our models was 80% with the following confusion matrix:

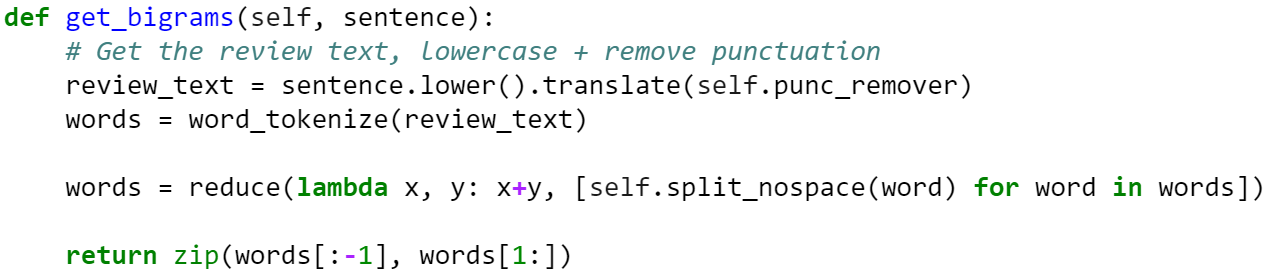


These were the unigrams that had the highest weights:

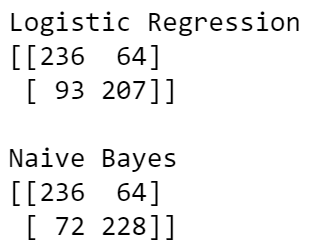


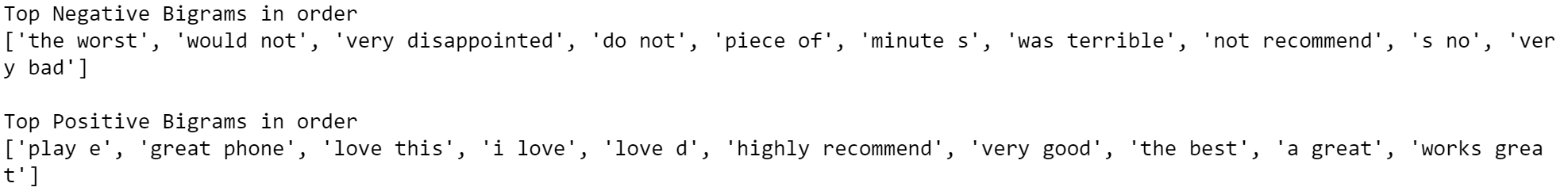
Our bigrams were created with our sentence cleaner class as well.

1. Remove punctuation and transform to lowercase
2. Tokenize the remaining text
3. Look at the tokens and split conjoined words in a token.
4. Pick two tokens at a time from beginning to end



Similar to the first part, we created a feature vector of bigrams where xi is the number of times bigrami appears in the sentence. The accuracy of our Logistic Regression was 73.83%, while the accuracy of our Bayes classifier was 77.33% with the following confusion matrix:





Once we applied PCA (and reconstructed projected matrices) and created training data with 10, 50, 100 features, Naïve Bayes was more powerful in the unigram case, but not the bigram case.

**Result and Discussion:**

In the unigram case, the Bayes classifier and Logistic Regression do not differ by much (only by 1/3 of a percent). But in the bigram case, the Bayes classifier outperforms regression by 4%. This is likely because the number of bigrams is much larger than the number of unigrams and Naïve Bayes – and Naïve Bayes handles the curse of dimensionality **extremely well**. Furthermore, Naïve bayes is a generative method instead of a discriminative method and likely handles sparse data more accurately.

What we do see though, is when dimensionality reduction via PCA is performed, Logistic Regression works better than Naïve Bayes in the bigram case, but not the unigram case. This may be because the training data features were more independent as bigrams than when trained in the projected direction (rotated), some independence is lost. Conversely, in the unigram case, the features may have had dependence that was decreased in the principle component direction (e.g. [‘apple’, ‘sauce’] has dependence).

Judging from the results of the language that people used in these reviews, it seemed that most of the value captured from bigrams were also captured in the unigrams (hence, our performance dropped). For example, most of the negative bigrams had negative words such as “bad”, “worst”, “terrible”, “not”. An interesting exercise would be to combine the features for both unigrams and bigrams and see if the performance increases – however, the independence assumption of Bayes classifier will be clearly broken.