Jim Sears
ADS652Z2 – Text Mining
Assignment 2

1. Preprocessing Modifications to Count Vectorizer

The goal of vectorizing text documents is to bring structure to unstructured text data; to create a numerical representation of the words that can be used to gain insight into their meaning, sentiment and relationships and measure the accuracy of those insights. In the case of this project of classifying movie reviews as fresh (positive) or rotten (negative), the SKLearn Count Vectorizer has been utilized to count the instances of each word in each critics' review commentary (document). To optimize the vectorization, several preprocessing steps are taken to prepare the documents and avoid some of the common issues with text mining.

By default, the SKLearn Count Vectorizer includes a normalization tool that that converts all words to lowercase to ensure that, for example, a capitalized word at the start of a sentence is treated the same as that same word that is in lower case in the middle of a sentence. It also includes a tokenizer that performs the task of separating out each word and piece of punctuation so that they may be categorized, manipulated, counted, etc. Additional steps may be taken to clean and further normalize the text to remove the "noise" and better prepare it for document classification. For this project, the following additional preprocessing methods were implemented and evaluated (in addition to changing the tokenizer to NLTK's):

a) **Stemming:** this process reduces words (now tokens) to their roots by removing common or specific patterns or word endings (-ed, -ing, -s, etc.) to normalize tense, pluralization, etc. This boils a list of related - or essentially the same - words down to a single term that not only groups them and reduces the overall number of tokens, but also enables the more frequent tokens to become more significant.
   A Porter Stemmer was implemented as the first preprocessing modification (vectorizer_1a) and the classifier's resulting weighted average F-1 score remained unchanged (0.73). One thought for the relative ineffectiveness of stemming movie reviews is that they are likely to be written after viewing the movie (and not before, obviously, or during) such that the past tense is so prevalent, and the present tense is not. Therefore, the stemmer will simply be converting tokens from past to present tense, reducing the count of the former and increasing the count of the former, and not grouping tokens and reducing the number of unique tokens. Though this is essentially a lateral move, this modification will be kept.
b) **Cleaning:** this process removes tokens that are deemed to be "noise" or not useful, specifically stop words and punctuation. The exclusion of punctuation is obvious as the frequency of periods, commas, and semicolons will not lead to any meaningful result. Stop words are those whose use is extremely common with the English language, including articles (e.g. the, a, an), pronouns (e.g. he, she, it, you), conjunctions (e.g. and, but, or, while), prepositions (e.g. by, with, about, until), common verbs (e.g. do, does, were, are), and so on. Their frequency is so high that including them only dilutes the population of tokens being analyzed, giving them value devalues the less frequent and unique words that are the best representation of the document in which they are found.
   The second preprocessing modification to the classifier's vectorizer (vectorizer_1b) was the implementation of the NLTK stop word and punctuation remover. The resulting weighted average F-1 score was 0.72, which is lower, but only nominally. This result is surprising and can

likely be explained by the nature of movie reviews that are short and concise. Reviewing the actual quotes, most are one sentence, and many read like a headline. This brevity and concise style results in the use of very little punction (often just a period at the end) and a tendency toward fewer words that are impactful (not stop words). Though this is essentially a lateral move, this modification will be kept.

c) **Document Frequency Minimum:** this later step in preprocessing involves further cleaning of the list of tokens that will be analyzed. The minimum document frequency sets a lower limit on the number of documents in which a word must appear for it to be considered for analysis. The rationale behind this step is that if a word only appears in handful of documents, then it is relatively less important in terms of the entire corpus (collection of documents). If a word appears in more documents more frequently – and is not a stop word of course – it is likely to be related to a common theme amongst the corpus. The additional benefit of this step is greatly reducing the dimensionality of the vectorized data.
The third preprocessing modification that was made to the classifier's vectorizer (vectorizer_1c) was setting the document frequency minimum. Several values were evaluated (with their resulting weighted average F-1 score), including: 10 (0.74), 15 (0.74), 20 (0.73), and 100 (0.65). Higher values resulted in poorer scores. This is because as the limit increases, the number of eligible words decreases and therefore the classifier has fewer words to evaluate and determine as associated with fresh (positive) or rotten (negative) reviews. Lower limits also resulted in poorer scores, this is because as the limit is lowered, the classifier has more words to evaluate and determine as associated with fresh (positive) or rotten (negative) reviews, which dilutes the importance of the more important terms.
A document frequency minimum of 15 will be used going forward. The resulting weighted average F-1 score increase (0.73 to 0.74) was nominal, but positive, nonetheless.

2. Different Hyperparameters for Naïve Bayes Classifier

The Bayesian classifier utilized in this project incorporates the probabilities of a number of variables, including the probability that a review (document) that contains a certain word is either fresh (positive) or rotten (negative). It does this by determining how often that word appears in fresh and rotten reviews. When building the model, the data is split into training and test sets. If a certain word happens to only appear in the documents in the test set, the model does not have the opportunity to calculate its probability of being in a fresh or rotten review and it is therefore considered to be zero. As a zero nullifies the Bayes Rule equation, they are countered with a hyperparameter (α, alpha; greater than zero) that is manually set.

The above weighted average F-1 scores were based on a Naïve Bayes Classifier with an alpha value of 0.01. The classifier was fitted with alpha values of 0.5 and then 0.7. In both cases, the resulting weighted average F-1 scores remained at 0.74, without any notable improvement to the model with preprocessing modifications mentioned above. However, when applied to the original SKLearn Count Vectorizer (lowercase normalization, default tokenizer, document frequency of zero), the alpha values of 0.5 and then 0.7 both resulted in weighted average F-1 scores of 0.77. Increasing the alpha value resulted in a more accurate model. This indicates that there were a relatively large number of tokens in the test set that were not encountered in the training set. This is expected as the reviews in general are short and succinct, so there is a higher likelihood that the training set is not exactly representative of the entire population.

3. Including N-grams in the Naïve Bayes Classifier

The modified Count Vectorizers described above all treat each token (word) as an independent entity. Assuming that words are in fact independent of each other enables the Bayesian Classifier to operate with a reasonably sized vector of potentially significant tokens. More importantly, it avoids many of the complexities of the English language, namely context. The meaning, sentiment and/or significance of a word can be influenced by the word or words that appear before or after it, or elsewhere within in the same sentence.

For example, the phrase "not great" has a negative connotation. But if divided, the components have different connotations: one is negative (not) and one is positive (great). One issue this may cause is a conflict for the classifier when calculating each words' probability of positive or negative association. Another concern is the removal of the word "not" as a stop word during preprocessing, leaving the word "great" alone and falsely obscuring or concurring and negative or positive review, respectively.

To counter these concerns, adjacent tokens may be grouped and treated as unique tokens. Typically, two-word (bigrams) or three-word tokens (trigrams) are created and analyzed in addition to the original one-word tokens. While incorporating N-grams (N is the number of tokens) enables some degree of context to be considered, the resulting vector is a considerably more multidimensional.

Incorporating bigrams into the Count Vectorizer described above (lowercase, stemmed, cleaned stop words / punctuation, 15 document frequency minimum) resulted in an unchanged weighted average F-1 score of 0.74 (vectorizer_3a). This was a surprising outcome until the preprocessing step taken to remove stop words was considered. A fresh (positive) review that includes the phrase "won't disappoint" is interpreted as "disappoint" because "won't" is removed. That creates a conflict (lower probability) for the association between the word "disappoint" and a fresh review. When stop words were not excluded (vectorizer_3b), the score increased to 0.75. This is a nominal increase but was improved to 0.77 when the minimum frequency of documents was reduced from 15 to 5. The latter change increased the number of tokens (monograms and bigrams) analyzed from 4,342 to 14,749 which proved to be a robust yet manageable number of dimensions.