1.

\*nltk.PorterStemmer has been used on my script.

|  |  |  |
| --- | --- | --- |
| Stopwords removed | text features | Accuracy (test set) |
| yes | unigrams | 0.799 |
| yes | bigrams | 0.796 |
| yes | Unigrams+bigrams | 0.82 |
| no | unigrams | 0.801 |
| no | bigrams | 0.827 |
| no | Unigrams+bigrams | 0.83 |

2. Answer the following two questions:

a. Which condition performed better: with or without stopwords? Write a brief paragraph (5-6 sentences) discussing why you think there is a difference in performance.

Withstopwords performs better. As we can see from the result, withstopword has more accuracy using Naïve Bayes Multinomial because comments from Amazon could be counted as context of sentiment analysis, the stopwords from nltk.corpus contain ‘no’, “few” ‘did’, ‘not’ certain of words which are negation that can alter the valence of the passage. If we remove those words from comments, it will be so hard to tell whether they are positive or negative comments. Consequently, so we can think of keeping stop words can classify positive/negative instance more accurately.

b. Which condition performed better: unigrams, bigrams or unigrams+bigrams? Briefly (in 5-6 sentences) discuss why you think there is a difference? Uni&bigram model performs the best under among conditions. Bigram is better than unigrams because bigrams include all the information in the unigrams because the system will become more "conservative" as n becomes larger. Hence, as an example, when a sentence is classified as "positive" using bigram, it will be classified as " positive " as well in unigram in most cases. Moreover, we can have more information if we mix unigrams and bigrams together because even if we have enough examples of individual words to get good unigram accuracy, we might not have enough examples of every bigram, so combination of unigram and bigram could help system to increase accuracy. The one thing we need to notice is that overfitting might decrease accuracy by using bigram instead of unigram.