Detection of SPAM in email

Introduction: As a part of this project, the classification of the Enron Public emails will be done. The regular emails are categorized as "ham" and spam emails are categorized as "spam". The corpus data is available in two folders "ham" and "spam". The classification will be done by preprocessing, tokenizing, filtering and by the use of features. The goal is to classify the emails as ham or spam.

Data Set , Process and Tokenize: The initial program provided, is reading the data and tokenizing the data from the ham and spam folders. Due to limitation of memory resources 250 spam and 250 ham emails were processed in which 90% were classified as training data and the remaining 10% as the test data.

Feature Function: I have added the keyword feature function document_features, this function will return true or false depending upon whether the key word is in the document. I have considered the "bag-of-words" and then did the experiment with most frequent words to be the word features.

Baseline Accuracy – I used the emails which has 250 characters and got all the words from ham and spam and trained the Naïve Bayes classifier and got the following accuracy.

All Words – 16740

Accuracy without any filter: - 0.813333333333

Experiments (Base Level)- I did the following two experiments on the all_words and observed that the accuracy is going down after I am filtering out the stop_char_set and stopwords.

- Bag of Words = 16740
- No Filter
  - ✓ No of words No Filter = 16740
  - ✓ Naïve Bayes Accuracy = 0.773333333333
- Filter by stop_char_set = set ([',','\'\'','%','\''])
  - ✓ No of words after stop_char_set Filter = 16736
  - ✓ Naïve Bayes Accuracy = 0.817777777778
- Filter by stopwords in nltk
  - ✓ No of words after stopword Filter = 16614
  - ✓ Naïve Bayes Accuracy = 0.768888888889

<u>More Experiments based on the Frequency Distributions(Combination of Base Level and Additional task)</u> – I carried out some more experiments and used the different no of most frequent words, by using the stop_char_set and stopwords word features. I took the baseline accuracy also without any filter on the words.

- No of Most Frequent Words = 16000
- No Filter
  - ✓ No of words No Filter = 16000
  - ✓ Naïve Bayes Accuracy = 0.773333333333
- Filter by stop_char_set = set ([',','\'\'','%','\''])
  - ✓ No of words after stop_char_set Filter = 15996
  - ✓ Naïve Bayes Accuracy = 0.773333333333
- Filter by stopwords in nltk
  - ✓ No of words after stopword Filter = 15875
  - ✓ Naïve Bayes Accuracy = 0.755555555556

The Accuracy Reduced in the third experiment. Let's try the experiment with 10000 most frequent words

- No of Most Frequent Words = 10000
- No Filter
  - ✓ No of words No Filter = 10000
  - ✓ Naïve Bayes Accuracy = 0.795555555556
- Filter by stop_char_set = set ([',','\'\'','%','\''])
  - ✓ No of words after stop_char_set Filter = 9996
  - ✓ Naïve Bayes Accuracy = 0.795555555556
- Filter by stopwords in nltk
  - ✓ No of words after stopword Filter = 9875
  - ✓ Naïve Bayes Accuracy = 0.764444444444

Here also the observation is that the accuracy is better than 16000, but it is still decreasing in the last featureset. Let's try with 6000 most frequent words.

- No of Most Frequent Words = 6000
- No Filter
  - ✓ No of words No Filter = 6000

    - ✓ Naïve Bayes Accuracy = 0.973333333333
- Filter by stop_char_set = set ([',','\'\'','%','\''])
    - ✓ No of words after stop_char_set Filter = 5996
    - ✓ Naïve Bayes Accuracy = **0.977777777778**
- Filter by stopwords in nltk
    - ✓ No of words after stopword Filter = 5875
    - ✓ Naïve Bayes Accuracy = 0.968888888889

In the above scenario, we got the excellent accuracy of 0.977777 in second featureset, using the 6000 most frequent words. Let's see if we can improve this model more, if not then this will be our final model to train and test the model. Let's try with 4000 most frequent words.

- No of Most Frequent Words = 4000
- No Filter
    - ✓ No of words No Filter = 4000
    - ✓ Naïve Bayes Accuracy = 0.964444444444
- Filter by stop_char_set = set ([',','\'\'','%','\''])
    - ✓ No of words after stop_char_set Filter = 3996
    - ✓ Naïve Bayes Accuracy = 0.964444444444
- Filter by stopwords in nltk
    - ✓ No of words after stopword Filter = 3877
    - ✓ Naïve Bayes Accuracy = 0.911111111111

The accuracy went down in the last scenario.

<u>Conclusion:-</u> The classifier which uses the 6000 most frequent words, excluding the stop_char_set, from the corpus is the best data to train since it gives an accuracy of **0.977777777778**