NLP Assignment 4

Intent

As a part of this assignment , I implemented the Naive Bayes Classifier and also tried experimenting with bigrams .

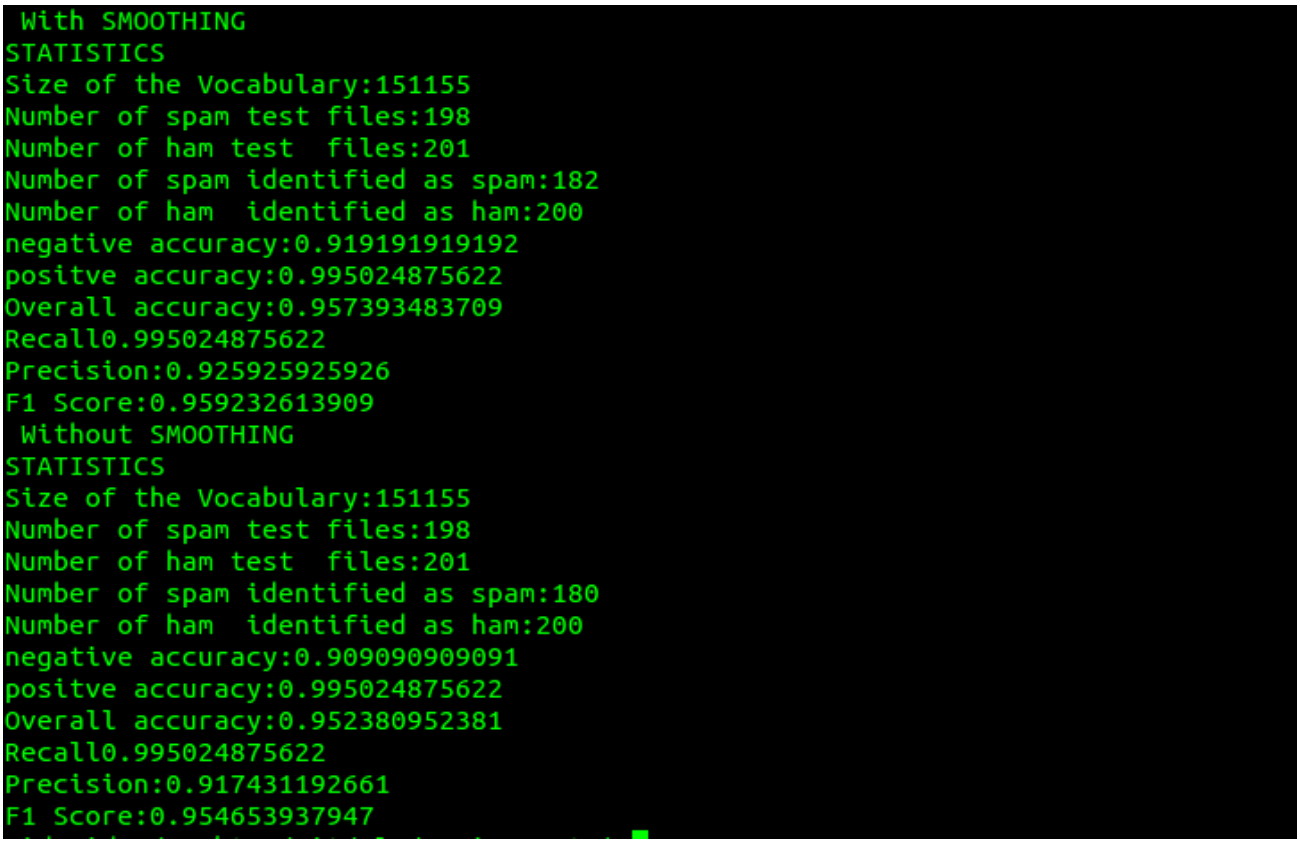
Task Accomplished

1. Implemented the Naive Bayes Classifier without smoothing and with Laplace smoothing on the Spam and Ham dataset
2. Used the Naive Bayes Classifier on the reviews dataset available at Polarity dataset in order to analyze the sentiments (Positive/negative reviews) .
3. Incorporated the bigrams to be used alongside the Bayes classifier . Instead of word independence , I assumed each word was dependent on its predecessor (bigrams) and experimented with this system

Task 1 : Naive Bayes Classifier with and without Smoothing

The first task involved running the Naive Bayes Classifier on the spam and ham dataset . I divided the data into 1200 for training and 200 for testing . I implemented Laplace smoothing to take care of the unknowns .

The statistics I presented included accuracy , Precision , Recall and the F1 score . The results were very promising for the spam , ham dataset



Analysis :

The results are extra ordinary with/without smoothing . However with Laplace smoothing , we see an improvement . This just goes to show that giving some probability to the unknowns does improve the effectiveness of the Classifier

Task 2 : On the Reviews Dataset

The second task was to use the Naive Bayes Classifier on the reviews dataset to perform sentiment analysis . This dataset just analysis the polarity of the review (Positive/negative) . I divided the dataset into 900 for training and 100 for testing for both positive and negative reviews.

When I ran the Naive Bayes Classifier on the original dataset , the accuracies were average . Here are the results

```
 With SMOOTHING
STATISTICS
Size of the Vocabulary:48272
Number of neg test files:100
Number of pos test  files:100
Number of neg identified as neg:81
Number of pos  identified as pos:79
negative accuracy:0.81
positve accuracy:0.79
Overall accuracy:0.8
Recall0.79
Precision:0.80612244898
F1 Score:0.79797979798
 Without SMOOTHING
STATISTICS
Size of the Vocabulary:48272
Number of neg test files:100
Number of pos test  files:100
Number of neg identified as neg:81
Number of pos  identified as pos:77
negative accuracy:0.81
positve accuracy:0.77
Overall accuracy:0.79
Recall0.77
Precision:0.802083333333
F1 Score:0.785714285714
```

When , I had a look at the reviews , I realized that they were raw text and hence , I decided to perform some preprocessing and removed all the stop words from the files and re ran the test on the clean negative and positive reviews

The results in this case seemed to improve a bit , but the improvement was not substantial .

```
 With SMOOTHING
STATISTICS
Size of the Vocabulary:48300
Number of neg test files:100
Number of pos test  files:100
Number of neg identified as neg:81
Number of pos  identified as pos:81
negative accuracy:0.81
positve accuracy:0.81
Overall accuracy:0.81
Recall0.81
Precision:0.81
F1 Score:0.81
 Without SMOOTHING
STATISTICS
Size of the Vocabulary:48300
Number of neg test files:100
Number of pos test  files:100
Number of neg identified as neg:83
Number of pos  identified as pos:80
negative accuracy:0.83
positve accuracy:0.8
Overall accuracy:0.815
Recall0.8
Precision:0.824742268041
F1 Score:0.812182741117
```

Analysis :

From all the above results it is clear that the smoothing need not improve the classification all the time (in the last case) . Moreover the Naive Bayes Classifier does a pretty good job in classification which means we should be able to do better , if we assume some dependence between words in the text .

An Interesting Experiment :

I wrote a movie review which had both positive and negative words in it

```
This was an average movie . The lead actors were pretty good but the storyline was bad .
The first half was pretty good. The second half was very bad .
~
```

The results given by the system :

```
sid@sid:~/Desktop/git/nlp/Assignment4$ python naivebayes.py neg pos moviereview
[(-198.19159367508536, 'pos'), (-194.4785574313193, 'neg')]
neg
-194.478557431
```

It is very encouraging to see that the log probability for both positive and negative are so close in this case.

Task 3

I had implemented the bigrams.py as a part of the previous submission . Then I was thinking , if the bigrams were to be used with the Bayes Classifier , it would produce better results and instead of assuming word independence , I said each word was dependent on its predecessor and made use of the bigrams.py

When the bigram bayes classifier was applied to the spam and ham dataset :

```
sid@sid:~/Desktop/git/nlp/Assignment4$ python bigrambayes.py spam ham  testing/ham testing/spam
argv ['bigrambayes.py', 'spam', 'ham', 'testing/ham', 'testing/spam']
Usage: bigrambayes.py classdir1 classdir2 [classdir3...] pos_test neg_test
STATISTICS
Number of positive review files:198
Number of negative review  files:201
Number of negative reviews identified as negative:186
Number of positive reviews  identified as positive:200
negative accuracy:0.939393939394
positve accuracy:0.995024875622
Overall accuracy:0.967418546366
Recall0.995024875622
Precision:0.943396226415
F1 Score:0.968523002421
```

For the reviews dataset :

```
argv ['bigrambayes.py', 'neg', 'pos', 'testing/pos', 'testing/neg']
Usage: bigrambayes.py classdir1 classdir2 [classdir3...] pos_test neg_test
STATISTICS
Number of positive review files:100
Number of negative review  files:100
Number of negative reviews identified as negative:79
Number of positive reviews  identified as positive:83
negative accuracy:0.79
positve accuracy:0.83
Overall accuracy:0.81
Recall0.83
Precision:0.798076923077
F1 Score:0.813725490196
```

The results seem to be on par or slightly better than the Naive Bayes Classifier .