CS 534 Machine Learning Programming Assignment - 2

Project group

1) Nikhil Lingutla (931-692-837)
2) Arwa Hamid (931-673-658)

1.1) NaiveBayes Classifier using Bernoulli Distribution

We are using the following formula based on Bayes theorem to calculate posteriors

$$Posterior = \frac{(Likelihood * Prior)}{Evidence}$$

The above formula can also be written as

$$Posterior \propto (Likelihood * Prior)$$

To be specific

$$P(Class/Document) = P(Class/words\ in\ document) = P(words\ in\ document/Class) * P$$

$$=> P(Class/Document) \propto P(word1/Class) * P(word2/Class) ..... * P(Class)$$

where

(without laplacian smoothing)

$$P(word_i/Class_j) = \frac{(number\ of\ documents\ that\ contain\ word_i)}{(total\ number\ of\ documents\ in\ class_j)}$$

(with laplacian smoothing)

$$P(word_i/Class_j) = \frac{(number\ of\ documents\ that\ contain\ word_i + k)}{(total\ number\ of\ documents\ in\ class_j + (k * V))}$$

where k is laplacian constant and V is size of vocabulary

Since the multiplication of probabilities can become smaller and smaller as the number of words increases, we are using the logarithms of likelihoods and priors and summing them as follow(since log is monotonic function of Probability, we have)

$$log(P(Class/Document))$$
$$\propto (log(P(word1/class)) + log(P(word2/Class)) + .. + log(P(Class)))$$

Using logarithmic equation avoids the underflow issues as the probability get added for each word. Hence it prevents underflow issues.

1.2)

We have empirically demonstrated the difference of using the normal probabilities and the applying logarithms to the probabilities.

Plot showing Number of Documents in each class for Test Data using

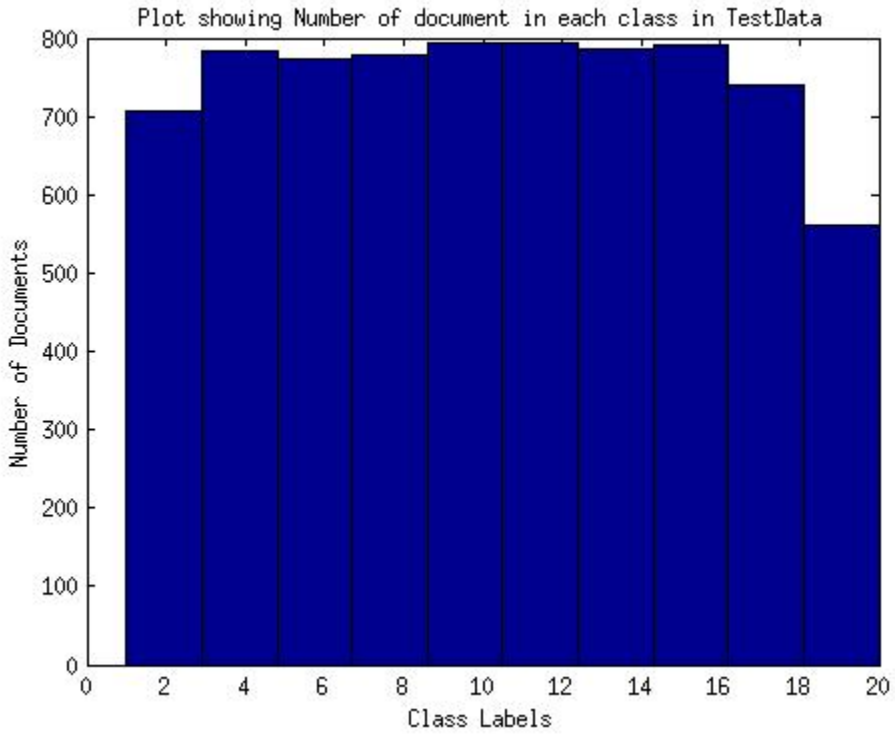

Fig 1.1) Number of Documents in each class for Test Data

Using normal probabilities (AccuracyRate is low because of Undeflow)

We achieved accuracy rate of **9.713524%** for NaiveBayes classifier using Bernoulli distribution

Total number of misclassifications are 6776 out of 7505 documents

Plot showing Number of Documents classified using our NaiveBayes classifier for Test Data using Normal probabilities

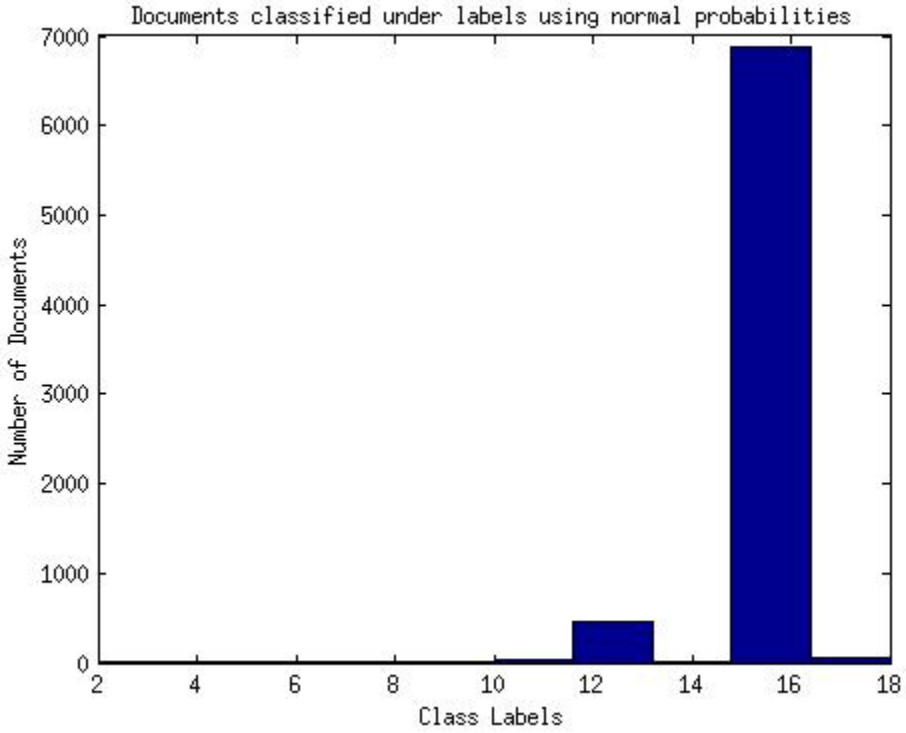

Fig 1.2) Number of Documents classified using our NaiveBayes classifier with normal probabilities for Test Data (Underflow error)

Applying logarithms to the probabilities(Eliminating underflow)

We achieved maximum accuracy rate of **69.94004%** for NaiveBayes classifier using Bernoulli distribution

Total number of misclassifications are 2553 out of 7505 documents

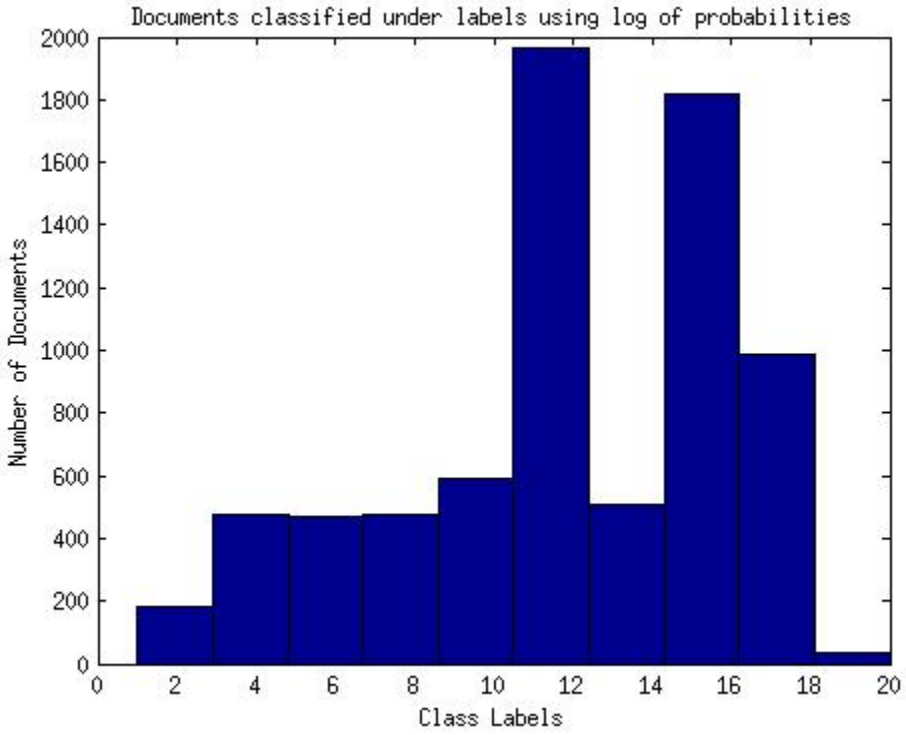

Fig 1.3) Number of Documents classified using our NaiveBayes classifier with logarithmic probabilities for Test Data

	Using Normal Probabilities (Underflow error included)	Applying log probabilities (Eliminated underflow)
AccuracyRate	**9.713524%**	**69.94004%**

1.3) Confusion Matrix

Confusion matrix for NaiveBayes using Bernoulli distribution
Cij species the total number of times that a class **i** document is classified as class **j**

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	0	0	0	0	1	0	0	0	0	0	6	0	6	7	120	2	5	2	1
0	0	7	10	2	50	0	1	0	0	2	71	4	9	18	9	0	1	0	0
0	10	0	48	1	47	0	0	1	0	0	92	2	13	14	17	3	1	2	0
0	6	15	0	8	6	3	2	0	0	1	49	24	8	9	4	1	2	0	0
0	10	4	36	0	8	3	3	0	0	0	67	15	23	12	8	3	4	0	0
0	13	2	4	0	0	0	1	1	0	0	48	0	6	7	1	0	2	0	0
0	8	0	43	12	2	0	28	4	0	2	25	16	19	19	11	7	14	0	0
0	1	0	0	0	0	2	0	9	0	0	12	2	4	7	4	15	12	0	0
0	1	0	0	0	0	1	22	0	0	0	6	1	4	2	1	6	8	0	0
0	0	0	0	0	1	0	0	1	0	20	12	2	6	5	20	8	7	0	0
0	0	1	0	0	0	0	0	1	1	0	3	0	0	0	2	2	2	1	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1	4	2	5	2	0	0
0	6	0	5	2	2	1	8	4	0	0	101	0	24	21	8	3	4	0	0
0	3	0	1	0	1	0	1	0	0	0	5	3	0	8	32	7	12	0	0
0	1	0	0	0	1	0	0	0	0	1	14	3	2	0	10	1	10	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	3	2	0	1	3	0	0
0	0	0	0	0	0	1	0	0	0	1	11	0	4	1	0	0	16	2	0
0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	10	6	0	0	0
1	0	0	0	0	0	0	0	0	0	0	14	0	5	5	20	85	43	0	0
16	1	0	0	0	0	0	0	2	0	0	8	0	1	8	147	22	21	3	0

Code snippet showing the use of laplacian and applying logarithms to probabilities

```
posteriors_combined(i,j) = sum(log((words_labels(wordIds,j)
+ laplacian)/(numel(find(train_labels == j)) + (laplacian *
no_of_words))) + log(priors(j))) ;
```

Code snippet showing the use of laplacian and normal probabilities

```
posteriors_combined(i,j) = sum(((words_labels(wordIds,j) +
laplacian)/(numel(find(train_labels == j)) + (laplacian *
no_of_words))) * (priors(j))) ;
```

2.1) Naive Bayes Classifier using Multinomial Distribution

We are using the following formula based on Bayes theorem to calculate posteriors

$$Posterior = \frac{(Likelihood * Prior)}{Evidence}$$

The above formula can also be written as

$$Posterior \propto (Likelihood * Prior)$$

To be specific

$$P(Class/Document) = P(Class/words\ in\ document) = P(words\ in\ document/Class) * P$$

$$=> P(Class/Document) \propto P(word1/Class) * P(word2/Class) ..... * P(Class)$$

where

(without laplacian smoothing)

$$P(word_i/Class_j) = \frac{(number\ of\ word_i\ in\ class_j)}{(total\ number\ of\ words\ in\ class_j)}$$

(with laplacian smoothing)

$$P(word_i/Class_j) = \frac{(number\ of\ word_i\ in\ class_j + k)}{(total\ number\ of\ words\ in\ class_j + (k * V))}$$

where k is laplacian constant and V is size of vocabulary

Since the multiplication of probabilities can become smaller and smaller as the number of words increases, we are using the logarithms of likelihoods and priors and summing them as follows and since log is monotonic function of Probability, we have

$$log(P(Class/Document))$$
$$\propto (log(P(word1/class)) + log(P(word2/Class)) + .. + log(P(Class)))$$

Using logarithmic equation avoids the underflow issues as the probability get added for each word. Hence it prevents underflow issues.

2.2)

We have empirically demonstrated the difference of using the normal probabilities and the applying logarithms to the probabilities.

Plot showing Number of Documents in each class for Test Data using

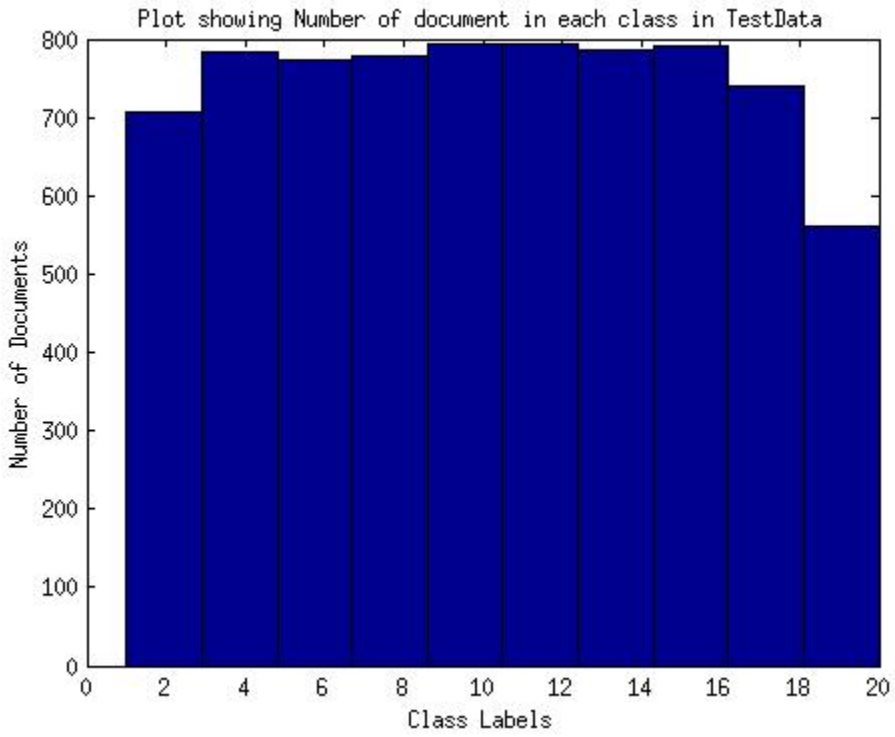

Fig 2.1) Number of Documents in each class for Test Data

Using normal probabilities

We achieved accuracy rate of **7.208528%** for NaiveBayes classifier using Multinomial distribution

Total number of misclassifications are 6964 out of 7505 documents

Plot showing Number of Documents classified using our NaiveBayes classifier for Test Data using Normal probabilities

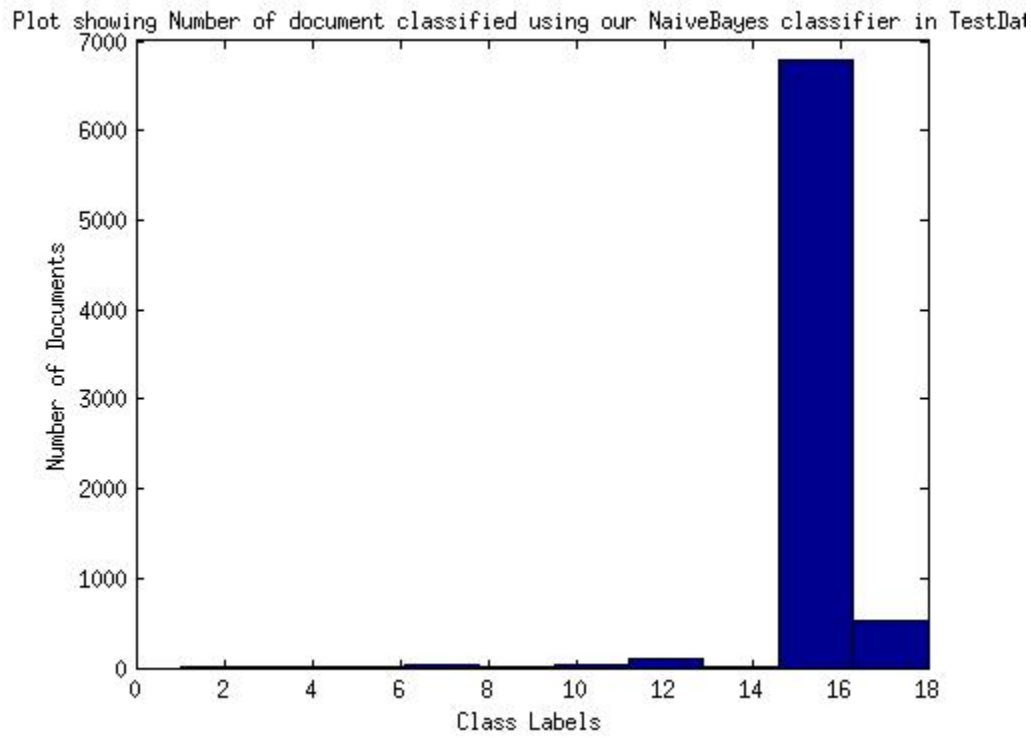

Fig 2.2) Number of Documents classified using our NaiveBayes classifier with normal probabilities for Test Data (Underflow error)

Applying logarithms to the probabilities

We achieved accuracy rate of **79.0861%** for NaiveBayes classifier using Multinomial distribution

Total number of misclassifications are 1852 out of 7505 documents

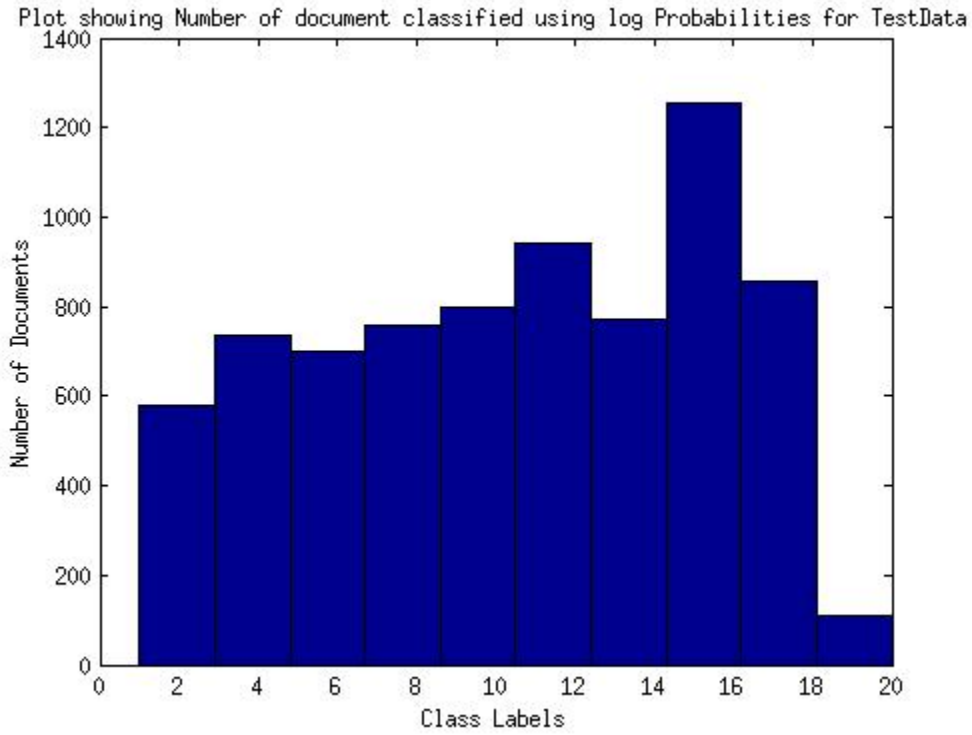

Fig 2.3) Number of Documents classified using our NaiveBayes classifier with logarithmic probabilities for Test Data

	Using normal Probabilities (causes underflow error)	Using log of probabilities (avoids underflow error)
AccuracyRate	**7.208528%**	**79.0861%**

2.3) Confusion Matrix

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	1	0	0	1	1	0	5	0	1	4	1	9	8	55	3	8	1	6
0	0	8	15	10	16	3	2	0	0	0	6	9	4	12	1	0	0	0	0
1	27	0	63	19	17	5	3	4	3	1	8	3	7	10	4	1	0	0	0
0	9	22	0	24	0	10	1	0	0	1	3	29	0	1	0	0	0	0	0
0	11	10	28	0	0	10	3	2	0	0	3	17	8	2	0	0	0	0	0
0	59	17	7	3	0	2	0	3	1	0	8	0	4	2	0	1	0	0	0
0	2	4	21	11	1	0	13	6	0	2	0	6	1	2	0	0	0	0	0
0	2	1	1	0	0	9	0	18	0	0	1	9	0	2	1	1	0	0	0
0	1	0	0	0	0	2	21	0	0	0	0	1	1	0	0	1	0	0	0
0	1	0	1	1	0	4	3	1	0	12	0	1	3	3	1	1	0	0	0
0	0	1	0	0	0	1	0	2	4	0	1	1	2	0	2	0	0	0	0
0	3	0	3	5	2	2	1	0	2	0	0	2	1	2	1	5	1	0	0
0	23	3	21	9	1	7	11	5	0	0	17	0	9	6	2	0	0	0	0
0	8	0	2	0	0	4	7	7	0	0	2	8	0	10	5	5	5	0	0
0	10	0	1	0	2	0	2	1	0	1	8	5	6	0	0	2	0	0	0
1	3	0	0	0	0	1	0	1	0	0	0	1	3	2	0	1	1	0	0
0	2	0	0	0	0	1	4	2	1	0	5	0	7	3	7	0	5	4	1
1	1	0	0	0	1	2	3	1	2	0	6	0	1	1	14	9	0	4	0
10	2	0	0	2	0	0	3	3	3	0	13	0	14	8	10	103	6	0	0
31	3	0	0	0	0	0	2	4	1	0	2	0	7	12	103	21	3	3	0

Code Snippets

Code snippet showing the use of laplacian and applying logarithms to probabilities

```
posteriors_combined(i,j) = sum(log((words_labels(wordIds,j) +
laplacian)/(sum(words_labels(:,j)) + (laplacian * no_of_words)))
+ log(priors(j))) ;
```

Code snippet showing the use of laplacian and normal probabilities

```
posteriors_combined(i,j) = sum(((words_labels(wordIds,j) +
laplacian)/(sum(words_labels(:,j)) + (laplacian * no_of_words)))
* (priors(j))) ;
```

Bonus Exploration

Approach 1

Varying the magnitude of Laplacian Constant

Using Bernoulli Distribution

Laplacian Value	**Accuracy Rate**
2	53.2884
1	57.6285
0.5	60.081279
0.3	62.74484
0.1	65.98268
0.01	**69.94004 (Max Accuracy)**

We have observed an interesting inverse relation between magnitude of Laplacian and Accuracy rate. As magnitude of laplacian decreases, accuracy rate increases upto zero(excluding zero)

Using Multinomial Distribution

Laplacian Value	**Accuracy Rate**
2	69.278
1	75.32312
0.5	77.44171
0.3	78.5463
0.1	78.69420
0.01	**79.08061**

We have observed an interesting inverse relation between magnitude of Laplacian and Accuracy rate. As magnitude of laplacian decreases, accuracy rate increases upto zero(excluding zero)

Approach 2 : Using Stop Words

At first we have collected several stop words in english language from Internet (links are provided at the end). These stop words are then removed from the given vocabulary. We have eliminated 720 words from the original vocabulary and still achieved the same accuracy rate.

	Before Stop Words	After Stop Words
# of Words	61188	60469
Accuracy Rate	79.08061%	79.00067%

We were able to achieve the same accuracy rate after decreasing the size of vocabulary by 719

Links to stop words:

1) http://www.lextek.com/manuals/onix/stopwords1.html
2) http://www.link-assistant.com/seo-stop-words.html
3) http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop
4) http://norm.al/2009/04/14/list-of-english-stop-words/