COMP9417, Assignment 2
Z5045582
Yunhe Zhang
Topic 11: Text categorization

Introduction:
In this assignment, my task is implementing a version of the Naive Bayes classifier and apply it to the text documents in the 20 newgroups data set.

Since each group contain hundreds of news files, they may contain same specific word, I think Multinomial Naive Bayes might be the proper model for this task.

In this case, I will mainly focus on implementing Multinomial Naive Bayes.

Implementation:
Platform: Python 3.6.0.
The dataset I chose is

- 20news-18828.tar.gz - 20 Newsgroups; duplicates removed, only "From" and "Subject" headers (18828 documents)

Two python files: ***CS9417Proj.py*** and ***CS9417Proj_test.py***
CS9417Proj.py contains 3 classes: Prepare(), Multinomial_NB() and Bernoulli_NB().
Each class contains several functions and they will be called in ***CS9417Proj_test.py***

Specific information of code is in README file

Naïve Bayes:
Referring to Chapter 6 of Mitchell's book.

$$g_{nb} = \underset{g_j \in dataset}{\operatorname{argmax}} P(g_j) \prod_i P(w_i|g_j)$$

For each group, the class is g_j, need calculate $P(g_j)$ as well as $P(w_i|g_j)$ for each word.
Where

$$P(g_j) = \frac{counts\ of\ all\ words\ in\ group\ g_j}{counts\ of\ all\ words\ in\ whole\ dataset}$$

$$P(w_i|g_j) = \frac{counts\ of\ \text{word}\ w_j\ in\ group\ g_j}{counts\ of\ all\ words\ in\ group\ g_j}$$

since w_i may not occur in group g_j, to avoid get probability value 0.

$$P(w_i|g_j) = \frac{n_{wi} + \alpha}{n_{g_j} + \alpha * |vocabulary|}$$

$n_{wi} = counts\ of\ \text{word}\ w_j\ in\ group\ g_j$
$n_{g_j} = counts\ of\ all\ words\ in\ group\ g_j$
$|vocabulary| = total\ specifc\ words\ set$

α smoothing priors $\alpha = 1$ is called Laplace smoothing, $0 < \alpha < 1$ is called Lidstone smoothing, here I chose $\alpha = 0.01$

Get specific words:

Two problems to get words:

1. Some words are same word but in different form like 'word' and 'words' or different tense like 'run' and 'running', they should be treat as one word.
2. Some words occur quite often in News but do not contain important significance like word 'the'.

Solution:

1. Use *PorterStemmer()* function in *nltk* to get each word's stem, for example with 'running' we can get 'run'
2. Use *stopwords* list in *nltk.corpus* , this list contains the words which do not contain important significance. Check every words, and drop the words which in *stopwords* list

After these two process, get 89647 different words and 3184937 total counts of words in dataset. To get specific words that can help with classifier, I drop the words which occur less than 5 times in whole dataset. After filter process, get about 27626 different words and 3082481 total counts of words in dataset.

Training set and Testing set:

To implement cross-validation, I divide the whole dataset into n folds randomly, each fold contains similar number of files. Every time use 1 fold for test, and rest folds for training.

Here I chose 10-fold cross-validation, so I divide dataset into 10 folds, and it will give me 10 results.

Here is the table about counts of words for 20 groups for 10 different training set. (the folds are divide randomly, here I set random seed 10 in program)

Group	Counts of words									
	1	2	3	4	5	6	7	8	9	10
1	124702	127670	127438	124474	127977	122706	126877	126903	123019	127217
2	129571	138917	133885	139549	131215	130562	141562	141296	130029	135673
3	293477	307727	295479	297897	293623	308632	302479	265854	294591	320438
4	96293	96194	95462	96778	95547	96271	99009	96125	97801	97768
5	82283	87632	86668	85569	81574	85996	85114	84999	86140	85371
6	161764	156407	167011	154743	157387	164816	161621	155315	164350	162843
7	74481	74201	73129	73773	73870	74328	71565	71587	73450	74079
8	104186	104101	101104	103809	105559	105688	105715	104362	106051	104685
9	98569	98826	98836	97465	94836	98550	97038	97408	97968	97319
10	108243	109338	108312	109216	108417	107876	111109	109507	108789	107554
11	130285	134065	131951	135303	134112	135578	132737	133544	131028	126035
12	162438	158878	165191	165057	168259	163492	164196	165346	161336	164275

13	99892	99923	99660	94554	99091	99295	100968	100158	100430	99665
14	144250	141973	142005	139514	146151	142290	147573	142316	145487	145003
15	147287	144000	146802	143197	146510	147192	144541	148499	145003	149064
16	170645	172606	169092	170051	170875	169894	165397	169952	169424	172938
17	152978	152448	148122	155080	151506	150434	156531	151923	154275	147757
18	214832	218325	219997	220282	219975	222720	218335	221914	218378	220234
19	156608	162174	156935	159178	159632	162169	150952	150384	159136	156986
20	103104	100319	103469	101713	105291	104371	101852	105010	105489	104049
Total	2755888	2785724	2770548	2767202	2771407	2792860	2785171	2742402	2772174	2798953

Table A

Performance and results:

First time when I chose first column training set in 'Table A' above, the accuracy is about 42%, and the reason is sometimes $P(g_j)\prod_i P(w_i|g_j)$ is too close to zero and out range of value type 'float', so I use log to solve this problem.

$$log\left(P(g_j)\prod_i P(w_i|g_j)\right) = log(P(g_j)) + \sum_i log\left(P(w_i|g_j)\right)$$

then I get 88% accuracy.

I use accuracy and f1 score(macro) to evaluate the performance.
Here are 10 results (random seed 10):

	Accuracy	F1 Score
1	0.882072977	0.872842422
2	0.881885593	0.873086907
3	0.889183457	0.882195539
4	0.893899204	0.888554186
5	0.887413702	0.880699557
6	0.879851143	0.871845173
7	0.888888889	0.879951021
8	0.883448643	0.87612557
9	0.869542066	0.861961544
10	0.892324094	0.881299229
Average	0.884850977	0.876856115

Table B

Then I try to remove the header, remove total line of 'From: XXX' and remove 'Subject:' part of 'Subject:XXX' line because I think the content of subject is quite helpful for classify the News.

I expect better performance, but get average accuracy 0.876620832 and average f1 score 0.866435983, slightly worse than results in Table B. This means header parts' words can slightly improve the performance.

Also use other seed number (15,20,25,30) to see the performance, give me quite same results, more detail in 'Results.pdf'

I also implement the Bernoulli Naïve Bayes model, get average accuracy 0.729622214.
So choosing the Multinomial Naive Bayes is right, it get better performance for this task.

Conclusion:

I have implemented Multinomial Naive Bayes and apply it with 20-newsgroups dataset, the results show that Multinomial Naive Bayes is a good classifier for classify the texts, get about 88% accuracy and 0.87 f1 score(macro) for 20-newsgroups dataset.
The significant part might be how to get the specific words in texts, because some words are noisy data and needed to be dropped. This part is what I have considered most, from this task I know do proper pre-process with dataset and selecting proper features are very important.
The rest part training and testing are quite straightforward, since the Multinomial Naive Bayes is not a complicated classifier, it has clear formulation and process, also easy to be understood.

Reference:
Chapter 6 of Mitchell's book
https://www.cs.montana.edu/~richter/mitchell_ch6_lecture.pdf