Your Name: ANURAG GANDHI

Your Andrew ID: agandhi1

Homework 2

Collaboration and Originality

1. Did you receive help of any kind from anyone in obtaining your data for this assignment (Yes or No)? It is not necessary to describe discussions with the instructor or TA.
   No

   If you answered Yes, provide the name(s) of anyone who provided help, and describe the type of help that you received.

2. Did you give help of any kind to anyone in obtaining their data for this assignment (Yes or No)?
   No

   If you answered Yes, provide the name(s) of anyone that you helped, and describe the type of help that you provided.

3. Are you the author of every word of your report (Yes or No)?
   Yes

   If you answered No:
   a. identify the text that you did not write,
   b. explain where it came from, and
   c. explain why you used it.

Your Name

Your Andrew ID

Homework 2

1 Experiment #1: Baselines

1.1 Parameters

Baseline 1: terms, frequency threshold = 10, all features
Baseline 2: terms, frequency threshold = 10, top 10 features (by kappa) per class.

1.2 Results

Reuters Baseline #1						
	NB			**SVM**		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
Gold	0.424	0.566	0.485	0.744	0.616	0.674
Heat	0.769	0.714	0.741	0.500	0.714	0.588
Housing	0.667	0.267	0.381	0.412	0.467	0.437
Coffee	0.579	0.807	0.674	0.835	0.754	0.793
Earn	0.840	0.903	0.870	0.901	0.901	0.901
Acq	0.438	0.947	0.599	0.790	0.779	0.784
Macro Average	**0.620**	**0.701**	**0.658**	**0.697**	**0.705**	**0.701**

Reuters Baseline #2						
	NB			**SVM**		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
Gold	0.459	0.798	0.583	0.582	0.394	0.470
Heat	0.647	0.786	0.710	0.909	0.714	0.800
housing	0.474	0.600	0.529	1.000	0.200	0.333
coffee	0.744	0.816	0.778	0.878	0.754	0.811
Earn	0.913	0.786	0.845	0.897	0.815	0.854
Acq	0.578	0.639	0.607	0.719	0.510	0.596
Macro Average	**0.636**	**0.738**	**0.683**	**0.831**	**0.565**	**0.673**

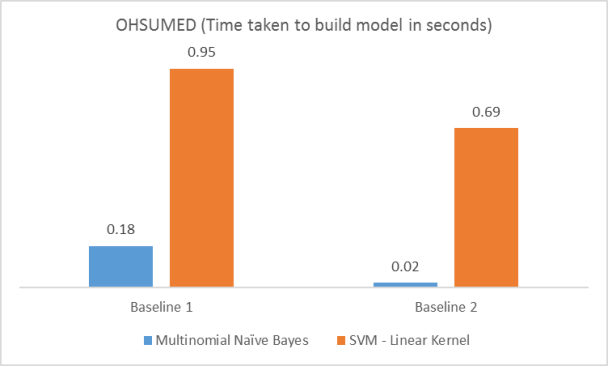
OHSUMED Baseline #1						
	NB			**SVM**		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
necrosis	0.333	0.163	0.219	0.514	0.287	0.368
pregnancy	0.813	0.836	0.825	0.824	0.805	0.814
hyperplasia	0.143	0.020	0.036	0.650	0.265	0.377
Rats	0.759	0.943	0.841	0.931	0.930	0.930
mitosis	0.000	0.000	0.000	0.000	0.000	0.000
pediatric	0.287	0.417	0.340	0.522	0.200	0.289
Macro Average	**0.389**	**0.397**	**0.393**	**0.574**	**0.415**	**0.482**

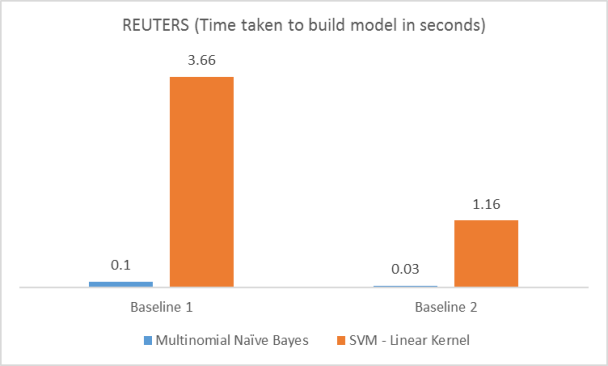
OHSUMED Baseline #2						
	NB			**SVM**		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
necrosis	0.652	0.333	0.441	0.639	0.302	0.411
pregnancy	0.886	0.770	0.824	0.912	0.729	0.810
hyperplasia	0.488	0.408	0.444	0.471	0.490	0.480
rats	0.814	0.954	0.878	0.986	0.887	0.934
mitosis	0.500	0.235	0.320	0.500	0.235	0.320
pediatric	0.526	0.500	0.513	0.724	0.350	0.472
Macro Average	**0.644**	**0.533**	**0.583**	**0.705**	**0.499**	**0.584**

1.3 Analysis

Speed:

Let's analyze the difference in speed for the two algorithms on both datasets:





Clearly, Naïve Bayes takes much shorter time to execute on both datasets than SVM. For cross-validation also, I found SVM takes much longer than NB. For comparable accuracy, this might be one of the factor to take into consideration while choosing an algorithm.

Comparison of models:

On Reuters dataset, SVM does better in terms of macro-averaged F1 measure than NB for Baseline #1. For Baseline #2, NB does better:

1. NB performs better on data with less features. Multinomial Naïve Bayes (MNB) assumes conditional and positional independence of terms. Increasing the number of features gives the algorithm more data to train but it also increases the noise in data. And since, practically, terms in a news document cannot possibly be conditionally independent, adding all features will decrease the accuracy of Naïve Bayes algorithm. This is especially true for classes such as "gold", "coffee", and "housing". These are relatively smaller classes than "earn" and "acq". This might be due to MLE errors in rare classes. The accuracy gains by reducing these features are a lot larger than bigger classes.
   This also visible in confusion matrix for NB when dealing with all features:

```
=== Confusion Matrix ===

   a    b    c    d    e    f    g   <-- classified as
  56    0   34    0    0    2    7 |    a = gold
   0   10    4    0    0    0    0 |    b = heat
  75    3 9716    2   67  629 2448 |    c = neg
   0    0   11    4    0    0    0 |    d = housing
   0    0   22    0   92    0    0 |    e = coffee
   0    0  239    0    0 3372  124 |    f = earn
   1    0   99    0    0   13 2012 |    g = acq
```

acq | coffee | earn | gold | heat | housing | neg

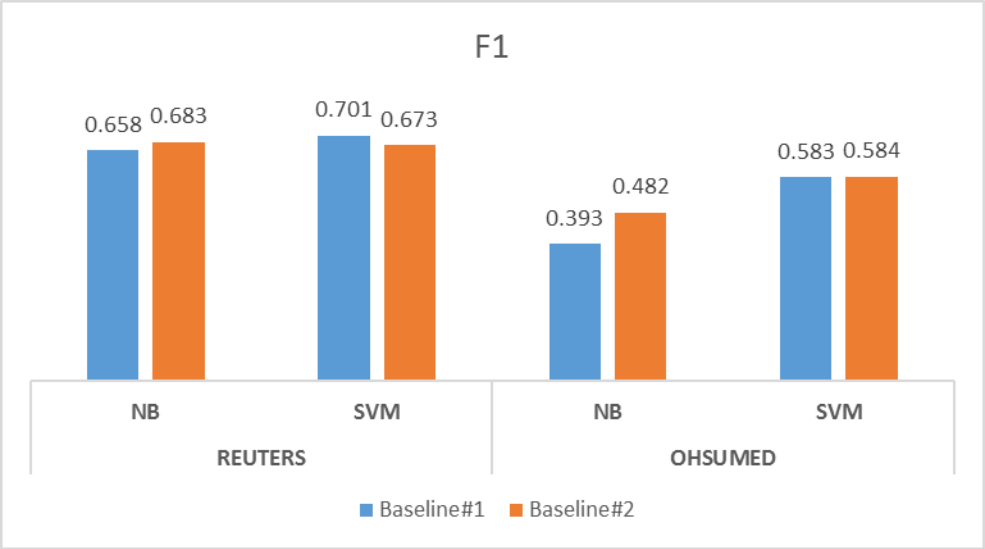
Feature	Type	Score
housing	term	0.18072
successive	term	0.13547
multi	term	0.12185
6.4	term	0.10611
seasonal	term	0.10309
family	term	0.09005
single	term	0.08356
4.15	term	0.07631
59,000	term	0.07631
6.7	term	0.07477
5.6	term	0.07381
2.16	term	0.07075
10.8	term	0.06909
27.4	term	0.06826
baldrige	term	0.06788
4.1	term	0.06775
0.4	term	0.06696
7.0	term	0.0667
128,000	term	0.06593
26.8	term	0.06593
malcolm	term	0.06446
0.2	term	0.06259
urban	term	0.06171
surprising	term	0.0598
2.18	term	0.05799
5.0	term	0.05767

A large percentage of gold, coffee, housing, and heat is classified as neg. Noise when using all features is a possible reason. The poorest performing class is arguably, housing.
Let's look at the top few terms for housing in the table on the left. Most of these are numbers and do not look like representative text. This might be another reason why the features are not able to represent this class effectively in both NB and SVM.

2. SVM performs better with more features and more data: SVM tends to find hyperplane to divide the classes that maximizes the margin and minimizes the number of misclassified examples. So feeding it more features will help it compute the distances between larger number of documents at the cost of computational complexity. In general, SVM outperforms NB in precision. Recall statistics for both algorithms are comparable when given all features, but with less features, NB scores better on recall for almost all classes.

OHSUMED:

A visual comparison between baselines, datasets, and algorithms is shown below:



For OHSUMED dataset, MNB behaves the similar way. With all features, performance is reduced. However, SVM shown no considerable difference in performance with 10 or all features. One of the reasons can be that SVM is not really dependent on number of features (as long as they are enough to distinguish between documents) because the input to algorithm is their transformation determined by the linear kernel used. OHSUMED features are arguably more linearly separable than Reuters. Therefore, SVM performs just as well with 10 feature space. Let's see the confusion matrix for NB for all features on this dataset:

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g   <-- classified as
 3822   34  305    3  781    0   55 |    a = neg
   91   21    0    2   15    0    0 |    b = necrosis
  160    2 1412    0  107    0    7 |    c = pregnancy
   37    2    1    1    8    0    0 |    d = hyperplasia
  161    4    8    1 2883    0    0 |    e = rats
   10    0    1    0    6    0    0 |    f = mitosis
   26    0    9    0    0    0   25 |    g = pediatric
```

Similar to "housing" class in REUTERS, "mitosis" and "pediatric" are poor performing classes here. In fact, no algorithm is able to predict any label correctly for "mitosis" in Baseline #1, but with Baseline #2 representation, both algorithms give similar accuracy scores. The larger classes are easier to train for both baselines and both algorithms because they have large amount of data to assign probabilities (NB) and to linearly discriminate between documents (SVM).

2 Experiment #2: Feature Selection

2.1 Parameters

terms, frequency threshold = 10, number of features = 5, 20, 50, 100, 500 for Reuters

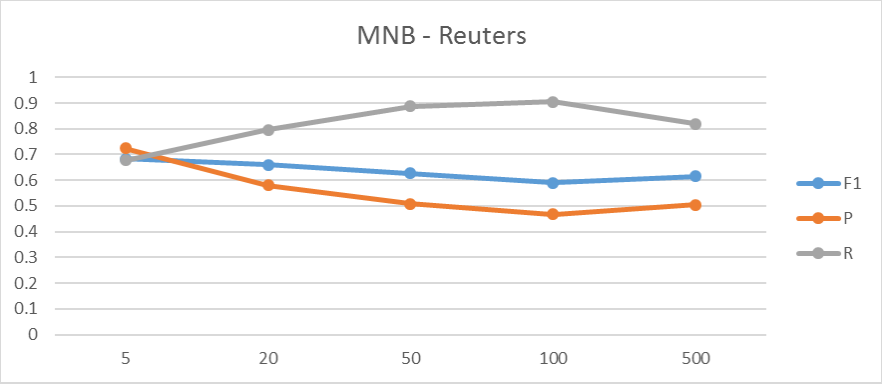
terms, frequency threshold = 10, number of features = 5, 20, 30, 50, 100 for OHSUMED

2.2 Results

	Reuters Number of Features														
	n_1=5			**n_2=20**			**n_3=50**			**n_4=100**			**n_5=500**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
NB	0.724	0.677	0.700	0.58	0.796	0.671	0.508	0.887	0.646	0.468	0.906	0.617	0.505	0.819	0.625
SVM	0.847	0.526	0.649	0.826	0.624	0.711	0.866	0.705	0.777	0.899	0.694	0.783	0.705	0.685	0.695

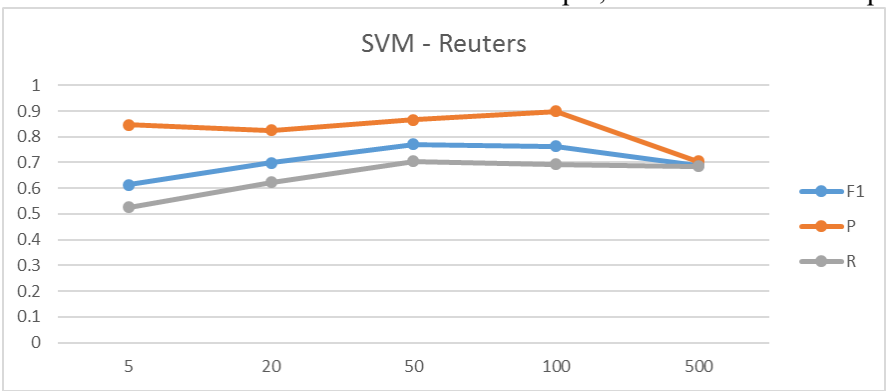
	OHSUMED Number of Features														
	n_1=5			**n_2=20**			**n_3=30**			**n_4=50**			**n_5=100**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
NB	0.75	0.467	**0.576**	0.598	0.634	**0.615**	0.581	0.631	**0.605**	0.508	0.657	**0.573**	0.524	0.664	**0.586**
SVM	0.69	0.463	**0.554**	0.703	0.492	**0.579**	0.672	0.488	**0.565**	0.616	0.484	**0.542**	0.679	0.49	**0.569**

2.3 Analysis



Refer to the plots showing accuracy for different number of features for Reuters dataset. Some observations:

1. For MNB, F1 tends to decrease till n = 100 then increases marginally for n = 500. Recall tends to increase with more features up to n = 100. F1 for baseline 1 was 0.658 which is less than F1 for n = 500. This indicates if we add more features, F1 will decrease. One of the reasons for decline in accuracy can be the increase in errors associated with expectation of algorithm of conditional and positional independence of terms. With increase in number of terms this is harder to maintain. For example, features of class "acq" – acquisition and acquire are highly correlated.

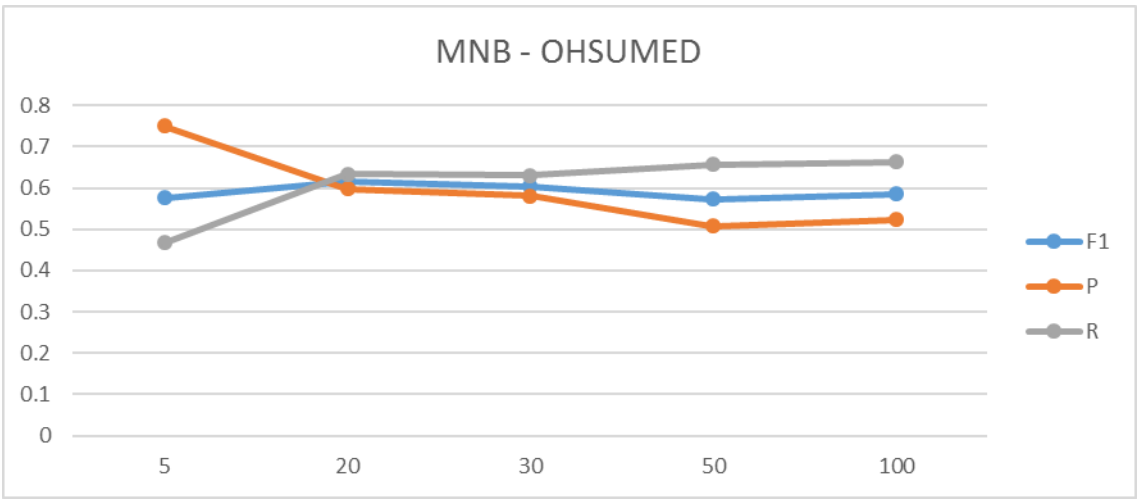


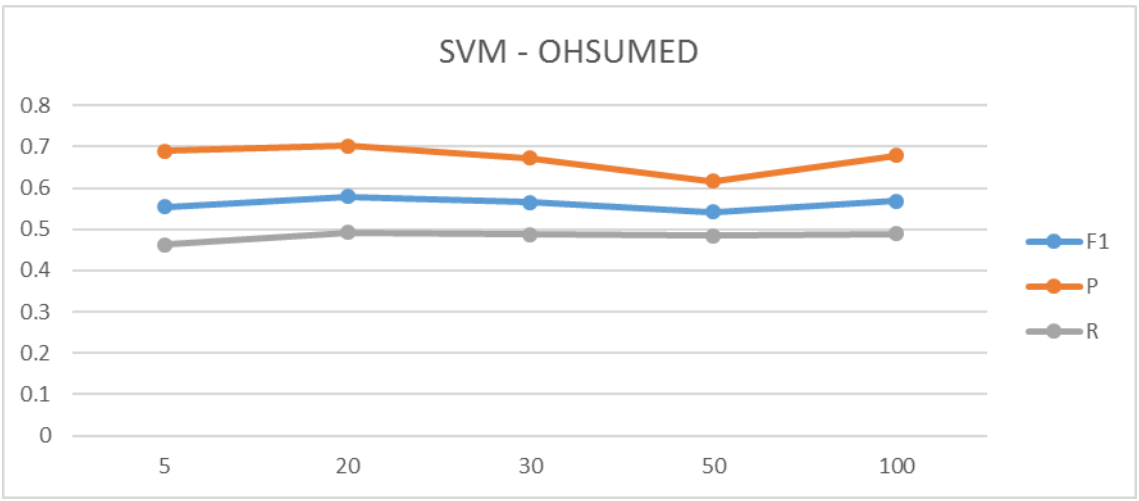
For housing, a name "Malcolm balbrige" appeared in several department. He held a position in Commerce Department. Therefore, terms Malcolm, balbrige, commerce are highly correlated and may not satisfy conditional independence.

2. SVM essentially shows a complete opposite trend with these settings. As I add more features, the number of documents represented also increase which helps the discriminative algorithm to separate the documents more easily. But notice after n = 50, accuracy statistics are essentially flat. Although adding more features after this point does increase the amount of data, the addition might not be significant to add an impact to accuracy.

3. The running and cross validation times also shoot up for SVM with addition of features because of added dimensional complexity.

Refer to the plots for OHSUMED dataset:





Insensitivity of SVM to number of features is even more visible in this dataset. Precision falls as the number of features are increased from 5 to 30. Let's compare the confusion matrices for Baseline #2 (10 features) and n = 30.

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
4170	22	156	16	608	3	25	a = neg
76	43	1	0	9	0	0	b = necrosis
345	1	1300	1	39	0	2	c = pregnancy
23	0	0	20	6	0	0	d = hyperplasia
128	0	9	3	2916	1	0	e = rats
8	0	0	1	4	4	0	f = mitosis
29	0	1	0	0	0	30	g = pediatric

Baseline #2 (n = 10)

vs

=== Confusion Matrix ===

a	b	c	d	e	f	g	<-- classified as
3916	38	155	36	761	2	92	a = neg
51	61	1	2	14	0	0	b = necrosis
239	5	1362	7	63	0	12	c = pregnancy
17	1	0	25	6	0	0	d = hyperplasia
96	4	8	8	2940	1	0	e = rats
4	1	0	2	4	6	0	f = mitosis
18	0	1	0	0	0	41	g = pediatric

n = 30

It is clear that for smaller classes, increasing the number of features from 10 to 30 improves the number of **true positives.** This effect seems to be greater on this data because medical terms are usually more distinct than terms

in news topics. This is why recall increases as we add more **relevant** features. After a certain point though, like in Reuters, accuracy starts to suffer. F1 score for n = 100 is 0.569 while in Baseline #1, we saw F1 was 0.482

For larger classes like rats and pregnancy, there is only a slight improvement in number of true positives. SVM already does a pretty good job segregating these two classes. Features for a class like rats are also inherently distinctive.

3 Experiment #3: Your Representations

Custom #1:

Text representation: terms and noun-phrases, frequency threshold = 10, number of features = 50

Reasons for choosing this representation:

1. From the previous section, for Reuters dataset n = 50 to 100 seems to be the range where both NB and SVM seem to perform fairly well in terms of accuracy. SVM outperforms NB in this range and for number of features greater than 100. But n > 100 increases the computation time with no significant addition to accuracy. Therefore, I chose n = 50 for the number of features.
2. Noun phrases: Since this is a news dataset, noun phrases might contain some information not represented by unigrams. For example, "housing" class had numeric unigrams – using phrases might better explain what those numbers mean. This might be even more relevant in medical context where names of diseases, symptoms, medication etc. are often noun phrases. I tried using only noun phrases but except for top few phrases, most of the phrases seemed irrelevant and redundant. So, I use a combination of terms and noun phrases and choose the top 50 features based on kappa.

Custom #2:

Text representation: terms + person + organization, frequency threshold = 5, number of features = 50

Reasons for choosing this representation:

1. I want to see the effect of lowering frequency threshold. There might be some terms that are missed due to frequency threshold > 10, especially for smaller classes. They might help better discriminate between classes.
2. Apart from noun phrases, news articles and medical journals also have influential persons and organizations. One of my hypothesis for choosing this representation is that by including variety – i.e. important persons and organizations might make the terms more independent. Noun phrases and terms might have a greater overlap or correlation between them than these entities.
3. Reasoning for number of features is same as for previous representation

Reuters Custom #1						
	NB			**SVM**		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
gold	0.322	0.949	0.481	0.750	0.667	0.706
heat	0.295	0.929	0.448	1.000	0.714	0.833
housing	0.297	0.733	0.423	1.000	0.467	0.636
coffee	0.519	0.956	0.673	0.807	0.772	0.789
Earn	0.888	0.857	0.872	0.886	0.884	0.885
Acq	0.501	0.874	0.637	0.786	0.698	0.739
Macro Average	**0.470**	**0.883**	**0.613**	**0.872**	**0.700**	**0.777**

Analysis:

Let's compare Custom#1 with n=50, terms representation. The only difference here is addition of noun phrases.

F1 measure for Naïve Bayes has actually reduced after adding noun phrases, while for SVM there is no observable change. I suspected noun-phrases might give more information for smaller classes but that does not seem to be the case. Comparing confusion matrices for n = 50 from previous experiment and Custom#1 on Reuters:

```
=== Confusion Matrix ===

   a    b    c    d    e    f    g   <-- classified as
  81    0   17    0    0    0    1 |    a = gold
   0   14    0    0    0    0    0 |    b = heat
 124   19 10649   28   92  372 1656 |    c = neg
   0    0    2   13    0    0    0 |    d = housing
   0    0    8    0  102    4    0 |    e = coffee
   2    0  385    0    0 3224  124 |    f = earn
   8    0  242    0    0    8 1867 |    g = acq
```

N = 50 Experiment 2

```
=== Confusion Matrix ===

   a    b    c    d    e    f    g   <-- classified as
  94    0    5    0    0    0    0 |    a = gold
   0   13    1    0    0    0    0 |    b = heat
 170   31 10493   26  100  395 1725 |    c = neg
   0    0    4   11    0    0    0 |    d = housing
   0    0    2    0  109    3    0 |    e = coffee
   9    0  405    0    0 3200  121 |    f = earn
  19    0  241    0    1    7 1857 |    g = acq
```

Custom #1

The Custom#1 model performs poorly not only for smaller classes, but also for bigger classes. The plausible reason seems to be a greater overlap between terms and noun-phrases violating the conditional independence assumption for Naïve Bayes. Recall did increase for class: coffee as is seen from confusion matrix. Some of the top noun phrases for coffee were: international coffee organization, costa rica - which when separated into individual terms might not have higher probability for coffee class.

Note: I also tried using only top 50 noun-phrases (no terms) for this analysis and the results were far worse. Except for "gold" class, precision dropped significantly for all classes. The macro avergaed precision in that case was 0.33 for Naive Bayes. This drop in precision was common across datasets and algorithms. This shows either unigrams contain more information than noun phrases for classification accuracy and/or noun phrases contain some redundant and irrelavant information that confuses the classifier.

Reuters

Custom #2						
	NB			**SVM**		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
Gold	0.389	0.798	0.523	0.767	0.667	0.714
heat	0.389	1.000	0.56	1.000	0.714	0.833
housing	0.424	0.933	0.583	0.833	0.333	0.476
coffee	0.568	0.947	0.711	0.846	0.772	0.807
earn	0.892	0.862	0.877	0.885	0.891	0.888
acq	0.514	0.876	0.648	0.790	0.695	0.740
Macro Average	**0.529**	**0.903**	**0.667**	**0.854**	**0.679**	**0.757**

Analysis:

In Custom#2, I combine different entities – person, organization and term. Compared to n = 50 from Experiment 2, there is improvement in Naïve Bayes precision for classes – gold, housing and coffee. Note that these are not very big classes. So, overall there is improvement in accuracy for Naïve Bayes. Shown below is the confusion matrix for NB:

```
=== Confusion Matrix ===

   a    b     c    d    e    f    g   <-- classified as
  79    0    18    0    0    0    2 |    a = gold
   0   14     0    0    0    0    0 |    b = heat
 116   22 10691   19   82  379 1631 |    c = neg
   0    0     1   14    0    0    0 |    d = housing
   0    0     3    0  108    3    0 |    e = coffee
   2    0   385    0    0 3221  127 |    f = earn
   6    0   250    0    0    8 1861 |    g = acq
```

Results for "housing" class are best so far. Out of 15 documents, classifier was able to correctly classify 14. There can be two reasons for this improvement:

1. Since I included more rare words by lowering the frequency threshold, the classifier now predicts correctly for smaller classes – we might have been ignoring these words before.
2. Organizations and people bring in some new information that was missing before. For example – "commerce department" for housing class or "international coffee organization" (this was true for noun phrases too).

Looking at the top 50 terms, there are not many people entities. So, it looks like adding people does not make that much of an impact. Also, organization entities are a lot fewer than unigrams. A lot of unigrams also cover the information contained in these entities but kappa score of the compound term is generally higher than individual terms, so they might have been missed before.

Based on these observations, lowering frequency threshold is the major factor driving improvement in Naïve Bayes accuracy for smaller classes.

SVM: Again, for larger classes SVM seems to be insensitive to feature representation. But for smaller classes like housing – there is reduction in performance due to lowering frequency threshold. Housing has very low recall with only 5 out of 15 documents correctly classified. One reason can be the fact that since SVM does not look at conditional probabilities, addition of "rare" words only adds *noise* in the kernel function. This is interesting because SVM and Naive Bayes behave completely opposite for this class for this representation.

OHSUMED **Custom #1**

	NB			SVM		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
necrosis	0.478	0.512	0.494	0.573	0.395	0.468
pregnancy	0.887	0.814	0.849	0.925	0.816	0.867
hyperplasia	0.315	0.571	0.406	0.621	0.367	0.462
rats	0.774	0.964	0.859	0.967	0.904	0.934
mitosis	0.667	0.353	0.462	0.500	0.118	0.190
pediatric	0.256	0.767	0.383	0.618	0.350	0.447
Macro Average	**0.563**	**0.664**	**0.609**	**0.701**	**0.492**	**0.578**

Analysis:

Unlike Reuters, in this dataset macro averaged F1, precision and recall for both Naïve Bayes and SVM have improved compared to Experiment 2: n = 50 representation. A contextual case can be made for effect of noun-phrases in this dataset. Below the confusion matrices for Experiment 2 and this representation:

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g   <-- classified as
 3853   62  161   46  764    4  110 |    a = neg
   48   66    1    1   12    1    0 |    b = necrosis
  213    5 1375    8   62    1   24 |    c = pregnancy
   15    0    0   29    4    1    0 |    d = hyperplasia
   91    3    9    7 2944    3    0 |    e = rats
    5    0    0    4    3    5    0 |    f = mitosis
   13    0    1    0    0    0   46 |    g = pediatric
```

N = 50 Experiment 2

```
=== Confusion Matrix ===

    a    b    c    d    e    f    g   <-- classified as
 3845   59  163   43  777    1  112 |    a = neg
   48   66    1    1   12    1    0 |    b = necrosis
  214    6 1374    8   64    0   22 |    c = pregnancy
   15    1    0   28    4    1    0 |    d = hyperplasia
   88    6   10    6 2947    0    0 |    e = rats
    5    0    0    3    3    6    0 |    f = mitosis
   13    0    1    0    0    0   46 |    g = pediatric
```

Custom #1

These actually are not very different. There are minor improvements in smaller classes like mitosis and hyperplasia, but except for that all other classes have similar distribution for accurate predictions. A document which was earlier confused for hyperplasia is now correctly classified as mitosis. The conditional independence is not "worsened" here like it was for Naïve Bayes because the medical noun-phrases are either names of medicines or phrases for conditions such as "blood transfusion" for mitosis. Another reason for noun-phrases brining only minor improvements can be the fact that only few of them appear in top 50.

SVM: For most of the classes, SVM's accuracy has only changes by only minor insignificant amounts. The only class where I could see some change was again, mitosis. In Experiment 2: n= 50 representation, classifier did not get even a single document right for this class. But now, there are 2 documents correctly classified. This shows that although SVM does not perform that well with very less data, if given more relevant features it shows some improvement.

OHSUMED Custom #2						
	NB			SVM		
Category	**P**	**R**	**F1**	**P**	**R**	**F1**
necrosis	0.478	0.496	0.487	0.591	0.426	0.495
pregnancy	0.889	0.815	0.851	0.923	0.816	0.866
hyperplasia	0.314	0.551	0.400	0.625	0.408	0.494
rats	0.777	0.964	0.860	0.970	0.903	0.935
mitosis	0.750	0.353	0.480	0.000	0.000	0.000
pediatric	0.269	0.767	0.398	0.590	0.383	0.465
Macro Average	**0.580**	**0.658**	**0.617**	**0.617**	**0.489**	**0.546**

Analysis:

This result is very similar to result of Custom#2 on Reuters dataset. There is marginal improvement in precision for Naïve Bayes if we compare this to Experiment 2 and Cusom#1. Adding more rare words increased precision for smaller classes like mitosis and pediatric. SVM is almost insensitive to this customization if we compare Experiment 2 and this result.

4 Experiment #4: Sentiment Baselines

4.1 Parameters

4.2

Baseline #1: terms, frequency threshold = 10, all features
Baseline #2: terms, frequency threshold = 10, top 10 features (by kappa) per class.

4.3 Results

Apps for Android							
		NB			SVM		
	Category	**P**	**R**	**F1**	**P**	**R**	**F1**
Baseline 1	poor	0.588	0.601	0.594	0.594	0.574	0.584
	ok	0.327	0.301	0.313	0.335	0.217	0.263
	good	0.873	0.88	0.876	0.861	0.913	0.887
	Macro Average	**0.596**	**0.594**	**0.595**	**0.597**	**0.568**	**0.582**
Baseline 2	poor	0.596	0.305	0.404	0.63	0.255	0.363
	ok	0.475	0.125	0.198	0.511	0.099	0.166
	good	0.791	0.959	0.867	0.782	0.974	0.868
	Macro Average	**0.621**	**0.463**	**0.530**	**0.641**	**0.443**	**0.524**
Custom 1	poor	0.587	0.547	0.567	0.658	0.499	0.567
	ok	0.359	0.258	0.300	0.508	0.115	0.188
	good	0.854	0.903	0.878	0.825	0.963	0.889
	Macro Average	**0.600**	**0.569**	**0.584**	**0.664**	**0.526**	**0.587**
Custom 2	Poor	0.602	0.610	0.606	0.659	0.594	0.625
	ok	0.351	0.321	0.335	0.444	0.176	0.253
	good	0.877	0.886	0.882	0.853	0.948	0.898
	Macro Average	**0.610**	**0.606**	**0.608**	**0.652**	**0.573**	**0.610**

Cell Phones and Accessories							
		NB			SVM		
	Category	P	R	F1	P	R	F1
Baseline 1	poor	0.543	0.571	0.557	0.547	0.519	0.533
	ok	0.319	0.259	0.286	0.312	0.223	0.26
	good	0.876	0.893	0.884	0.866	0.909	0.887
	Macro Average	**0.579**	**0.574**	**0.577**	**0.575**	**0.550**	**0.562**
Baseline 2	poor	0.556	0.223	0.319	0.608	0.186	0.285
	ok	0.316	0.039	0.069	0.421	0.026	0.049
	good	0.795	0.973	0.875	0.79	0.985	0.876
	Macro Average	**0.556**	**0.412**	**0.473**	**0.606**	**0.399**	**0.481**
Custom 1	poor	0.585	0.492	0.535	0.661	0.431	0.522
	ok	0.338	0.177	0.233	0.492	0.063	0.112
	good	0.847	0.927	0.885	0.826	0.976	0.895
	Macro Average	**0.590**	**0.532**	**0.560**	**0.660**	**0.490**	**0.562**
Custom 2	poor	0.566	0.579	0.572	0.640	0.529	0.579
	ok	0.331	0.28	0.303	0.403	0.127	0.193
	good	0.879	0.895	0.887	0.852	0.960	0.903
	Macro Average	**0.592**	**0.585**	**0.588**	**0.632**	**0.539**	**0.581**

4.4 Custom Representations

Custom #1:

Text representation: terms and noun-phrases, frequency threshold = 10, number of features = 100

This representation showed mixed results for news and medical datasets. But I wanted to test this again for sentiment because they can better represent technology terms.

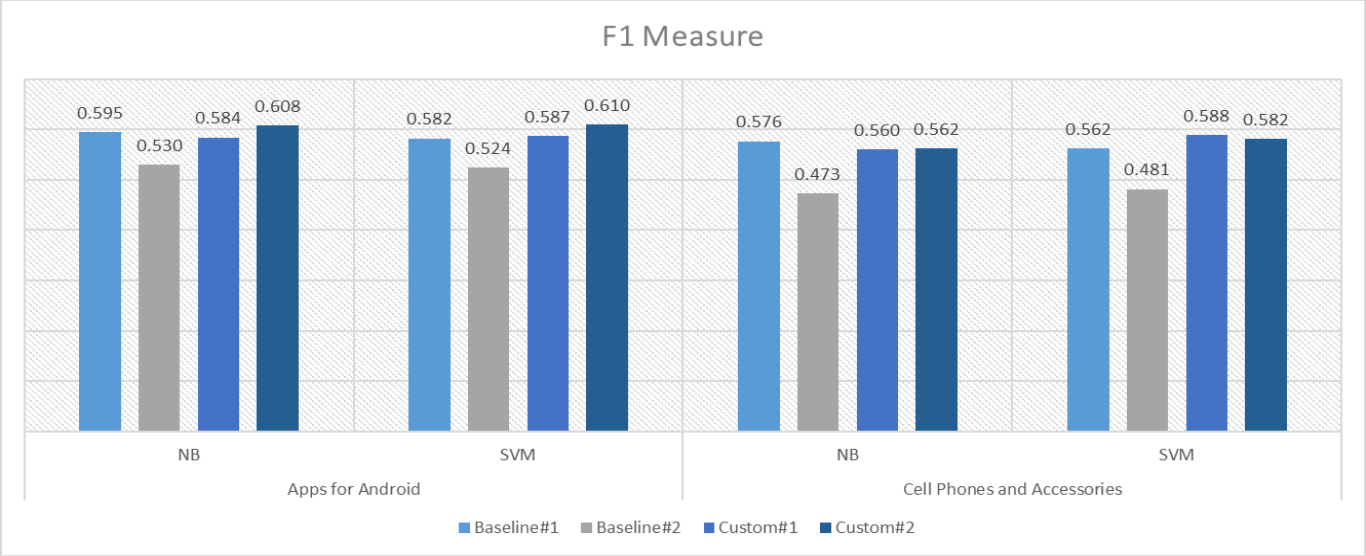
Custom #2:

Text representation: terms, frequency threshold = 5, number of features = 500

This time I do not use person + organization combination because for sentiment, I believe they should not matter a lot. Reducing frequency threshold did give some gains, especially for Naïve Bayes so I will try this on these datasets as well. I also increased the number of features because there is a large number of documents and only 3 classes so greater number of features might improve accuracy.

4.5 Analysis

One common observation here is that across both datasets, both baselines, and both custom representations, SVM is only marginally better than NB, and in a few cases marginally worse. A visual summary for F1 measure is shown below:



Baseline 1 vs Baseline 2:

Baseline #2 – with only top 10 features perform poorest in all cases. Unlike topic classification, sentiment classification seem to improve with a lot of features across all algorithms and datasets. One of the reasons can be that this time we have balanced classes, and we saw that larger classes generally perform better with more features. Also, it might be due to fact that lexicon for sentiments is just too big to be explained by only 10 features. Using all features – Baseline #1 – improves accuracy for both SVM and Naïve Bayes.

Ok class:

Another common thread across representations and algorithms is that positive sentiments are easier to learn than other two categories. **Ok** class is the hardest to learn which makes sense because looking at some top features for this class in Cell Phones dataset, there are words like "better", "bad", "pretty", "hard" – which makes it more ambiguous for the classifier. Documents for this class seem to follow a common thread: The user lists both advantages and disadvantages of the product in a balanced way. For example:

body

The idea of the design is a sweet idea but it wears off as the paint is over the case and not under a sealant or anything. It has a cool 3D effect but at the cost of the paint rubbing off.It is quite pretty though.

Words like "though" and "but" are good predictors for this class, but again some documents are confusing even for humans to classify so I would expect a classifier to behave this way.

I also noticed that these review documents are much shorter than the news articles and medical documents. This might also be one of the reasons why Naïve Bayes performs as well as SVM for this task as it can work well with less amount of data.

Custom Representations:

Custom 1: Custom 1 did not perform better than Baseline 1 for Naïve Bayes in both datasets. Recall for "good" class improved slightly but "ok" performed much worse. Since "ok" is the ambiguous class, I suspect giving it a lot of features might be the reason the classifier is able to assign better probabilities to each term.

SVM, on the other hand had an improved F1 score (although marginally) in Custom 1. The major difference was in precision of "ok" class. It improved significantly by limiting the number of features to 50, while recall reduced. Looking at the confusion matrix, I found that compared to Baseline 1, there are less proportion of words from "good" and "poor" classes classified as "ok" class. One of the reasons can be that with most relevant 50 features, SVM does well in distinction of good and poor classes from ok class but not vice versa.

Effect of noun phrases: The noun phrases in this representation were terms like "battery life", "cell phone" etc. and they were a lot more sparse in top 100 features than in previous datasets (1 or 2 terms per class). Hence, I concluded that noun-phrases do not affect the performance of classifiers for sentiment analysis on these datasets.

Custom 2: The macro averaged F1 scores for this representation convey that this seems to be a dataset-specific tuning. Reducing the frequency threshold worked well for Apps for Android, but not for Cell Phones and Accessories. Both NB and SVM had a better F1 score after include rarer terms in Apps for Android dataset, but on Cell Phones and Accessories, this tweak did not result in a significant difference for both classifiers.

In Apps for Android, improvement was in "ok" class for Naïve Bayes and in other 2 classes for SVM. Including more features would have helped but limiting it to top 500 still ensured that noise is less than Baseline 1.

Bottom line: I do not see considerable improvements from Baseline 1 with my custom representations. Both the algorithms are pretty stable for both the datasets if given enough number of features (Custom 1 – 100, Custom 2 – 500). Also, "ok" class remains hardest to predict. There is a lot of subjectivity involved to classify neutral documents which bag-of-words model somewhat misses in my opinion.

Note: To reduce the subjectivity, I tried using bigrams as features in combination with unigrams, but I found that the performance did not improve for any class. Looking at the features I saw that stop-words also show up in bigrams. For example – "it is", "is good" etc. Preprocessing the bigrams before feeding them into Weka might have helped.

5 Effectiveness of the Different Representational Choices

Best representations for each algorithm and dataset (decided by macro-averaged F1):

Dataset	Algorithm	Best Representation	F1 Score
Reuters	Multinomial Naïve Bayes	terms, frequency threshold = 10, # features = 5	0.700
Reuters	SVM	terms, frequency threshold = 10, # features = 50	0.777
OHSUMED	Multinomial Naïve Bayes	terms, frequency threshold = 10, # features = 20	0.615
OHSUMED	SVM	terms, frequency threshold = 10, # features = 10	0.584
Cell Phones and Accessories	Multinomial Naïve Bayes	terms, frequency threshold = 5, # features = 500	0.588
Cell Phones and Accessories	SVM	terms, frequency threshold = 5, # features = 500	0.581
Aps for Android	Multinomial Naïve Bayes	terms, frequency threshold = 5, # features = 500	0.608
Aps for Android	SVM	terms, frequency threshold = 5, # features = 500	0.610

Some common trends across datasets:

1. **Increasing number of features:** This technique works well to the point where classifier has enough relevant features. Beyond that, accuracy tends to stabilize, and even reduce in some cases. SVM, in particular, seems to be independent of number of features (considering there are enough to represent documents for each class) and is more sensitive to the amount of training data. Multinomial Naïve Bayes

performs well even with very less data – in terms of number of words in the document (for example – in case of shorter documents for sentiment analysis). Choosing the number of features is a dataset and algorithm specific task.

2. **Using noun-phrases:** This, in general, did not work for me on any of the dataset. Most of the information in noun-phrases was already represented by unigrams, and using noun-phrases without unigrams produced worse results because of the low Kappa score for most noun-phrases.

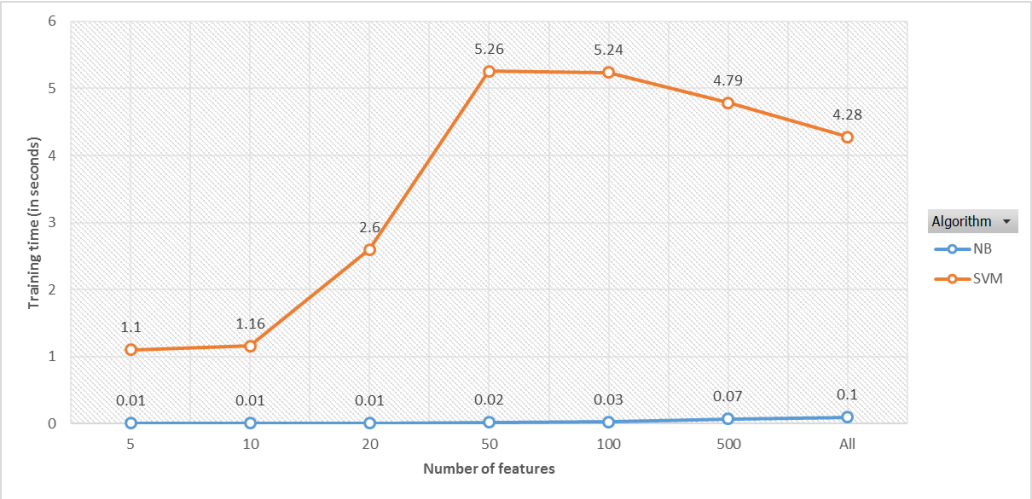
3. **Reducing the frequency threshold:** This worked well for smaller classes in topic classification experiments. However, for sentiment classification, there was not much difference. After looking at the top terms for each sentiment class in Sifaka, I also found that most of these were common vocabulary terms and reducing the frequency threshold did not really make a change to top 100, top 500 terms (by kappa) for a particular sentiment class.

6 Effectiveness of the Different Learning Algorithms

Some observations:

1. **Choosing a classifier is dataset and task specific exercise.** SVM performed better in general for topic classification due to high precision, but Naïve Bayes gave higher recall. Differences for sentiment analysis did not seem significant.
2. **Naïve Bayes works well on shorter documents and less data.** Reviews datasets, and some classes like housing in Reuters had comparatively shorter documents. NB performed very similar or in some cases better than SVM in these cases.
3. Naïve Bayes is insensitive to irrelevant features.
4. **SVM is more insensitive to addition of more features.** After a certain number of features, accuracy of SVM classifier is largely stable.
5. **Efficiency:** Naïve Bayes is a lot faster than SVM. Since NB assigns probabilities, it's computations are much more simplistic than SVM which tends to find a multi-dimensional hyperplane linearly separating the classes. With more features, this complexity increases but NB still performs very fast with lot of features.

   Training time for SVM increases sharply but stabilizes when there are a lot of features. This is an important consideration when choosing a classifier especially in Experiment 4 where differences in accuracy are not that significant. Below is a comparison of training times for NB and SVM on Reuters for Experiment 2:



6. **Feature selection** affects the performance of classification task to a large degree. Optimal feature representation varies by task and classifier, but I found that using unigram terms worked as well or better than using other entities for both classifiers and on all datasets in these experiments.