Extending upon Text Categorization using Naive Bayes, CS276 PE2

Rio Akasaka (rakasaka), Joseph Lau (jhlau)

In this project we evaluate and expand upon Naive Bayes classifers using the 20 Newsgroups data set of nearly 20,0000 documents from MIT. In addition to comparing multivariate with multinomial classification, we perform Complement Multinomial Naive Bayes (CNB), Weight-Normalized Complement Multinomial Naive Bayes (WCNB), and Transformed Weight-Normalized Complement Multinomial Naive Bayes (TWCNB) classification.

[Note: this was turned in on May 30th, within 4 days after the deadline of midnight on the night of the 26th, using 4 late days from both Rio and Joseph]

1, 2: Multinomial and Multivariate Naive Bayes (w/ Chi-squared)

We implemented a multivariate Naive Bayes classifier using log probabilities to prevent underflow and Laplace smoothing for overfitting. We noticed that the lowest accuracy result is in the `talk.religion.misc` newsgroup, which we can explain by the possibility that the discussions that take place here are more varied. The `talks.politics.misc` also exhibits poor classification accuracy for similar reasons.

	MV	X^2 MV	MNB	X^2 MNB
alt.atheism	18	20	20	19
comp.graphics	16	16	20	19
comp.os	19	17	19	16
comp.sys.ibm.pc.hardware	20	15	18	17
comp.sys.mac.hardware	19	19	19	20
comp.windows.x	18	18	20	19
misc.forsale	19	17	18	17
rec.autos	18	15	19	18
rec.motorcycles	19	19	19	19
rec.sport.baseball	17	17	20	20
rec.sport.hockey	20	19	20	20
sci.crypt	15	18	19	19
sci.electronics	18	17	19	16
sci.med	17	18	20	18
sci.space	13	15	19	19
soc.religion.christian	19	20	20	20
talk.politics.guns	18	18	20	20

It is also worthwhile noting the differences in speed taken for the entire process of training and testing. We recorded the time required for the processes using by a somewhat rudimentary measure using the system time, but it nonetheless highlights the amount of overhead needed to reduce the number of desirable features using X^2.

We see that there is an obvious cost-benefit trade-off of selecting features. Selecting X^2 values takes time, meaning training time as a whole is longer for both MVB and MNB. However, we see that MVB spends a significantly shorter amount of time classifying, and actually saves time in the end. This makes sense because MVB originally summed up values for each word in the dictionary, which is many more than the number of features selected. MNB, on the other hand, takes longer in total but spends about the same amount of time classifying. MNB saw no benefit in feature selection because the difference in number of terms it summed up was too small.

talk.politics.mideast	17	19	20	20
talk.politics.misc	14	17	20	18
talk.religion.misc	5	14	17	13
Total classification accuracy	339 **84.75%**	348 **87%**	386 **96.5%**	367 **91.75%**
Time taken in seconds (training + classification) = total	15.3 + 4.13 = 19.43	16.5 + 1.88 = 18.38	15.8 + 2.03 = 17.83	16.5 + 2.1 = 18.6

Table 1: *Classification accuracy and times by newsgroup and classifier*

3: Chi-squared

Chi-squared feature selection helps identify the most salient features for a class. This helps the classifier focus on features that are more indicative of different classes, and has the added bonus of dramatically reducing the time it takes to classify for Naive Bayes classifiers and also particularly for SVMs. Below we show results of varying the number of features chosen for both Multivariate and Multinomial Bayes.

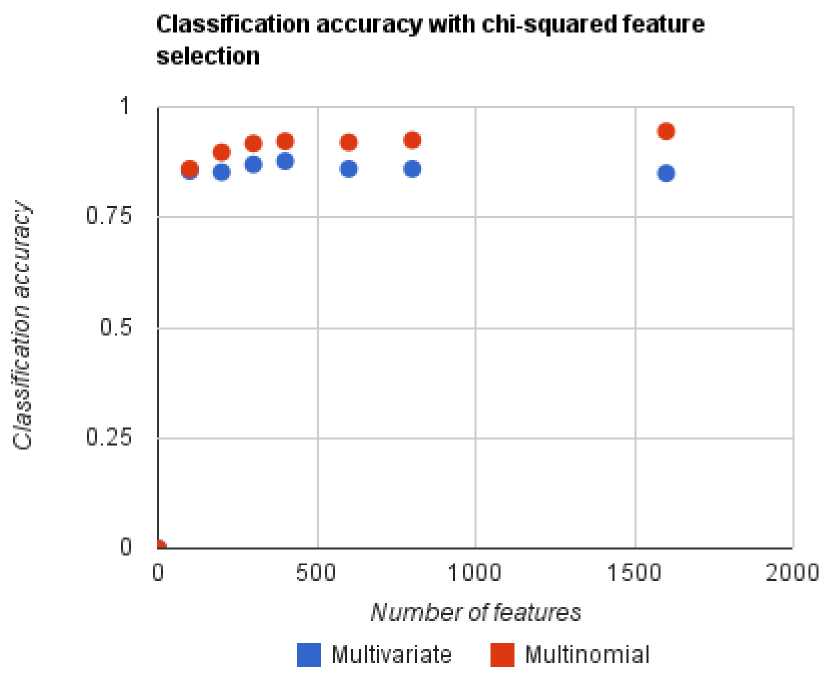


Figure 1. *Classification accuracy with chi-squared feature selection for multivariate and multinomial Naive Bayes*

We see that the accuracy of Multivariate Bayes improves with the selection of a moderate number of features (~300-400), but falls off with both fewer and more features. This makes sense because Multivariate Bayes counts only occurrences, and certain words have much more significance and therefore should have much more weigh; so picking the right features is necessary to strengthen the signal. Picking too few resulted in overspecialization, picking too many resulted in too much noise.

On the other hand, Multinomial becomes more accurate as the number of features increase (reaching maximum accuracy when all words are features). Selecting a small fraction of the dictionary as features degrades

Multinomial performance because the model considers term frequencies, already accounting for the significance of certain words (whereas Multivariate didn't). Limiting the features in a Multinomial Bayes classifier is akin to limiting the amount of information the classifier has: the fewer the features, the less the information and the worse the accuracy. The **Appendix** lists some of the top features selected for newsgroups.

We also noticed that there was an obvious cost-benefit trade-off in terms of time and accuracy involved in chi-squared feature selection. We mention this in detail in **Part 1: Multinomial and Multivariate Naive Bayes**, but it's worth restating the conclusions here. Basically, for Multivariate Bayes, spending time to select features ends up in a drastically reduced classification time (and more accuracy, as explained above), since Multivariate Bayes sums a term for each feature. On the other hand, Multinomial Bayes sums a term for each position in the test document and so does not see the same reduction in classification time.

Implementation notes:
Chi-squared values were calculated differently for Multivariate than they were for Multinomial. Calculations for Multivariate were centered around the fraction of documents with occurrences in a class, and the total number of documents. Calculations for Multinomial were centered around the summed frequency of a term in a class, and the total number of terms. These calculations followed the guidelines laid out in lecture notes and on Piazzza (@239).

4: K-fold

k-fold classification is convenient to divide one single data set into training and testing. In the table below, we can see that the larger k is, the more accurate the classifier becomes. This makes intuitive sense: as the training set becomes larger compared to the test set, the classifier becomes more accurate. We also observe the effect of diminishing returns based on different values of *k* plotted against classification accuracy in the plot below. The suggested value of 10 is sufficient to obtain good results without spending too much time in unnecessary overhead.

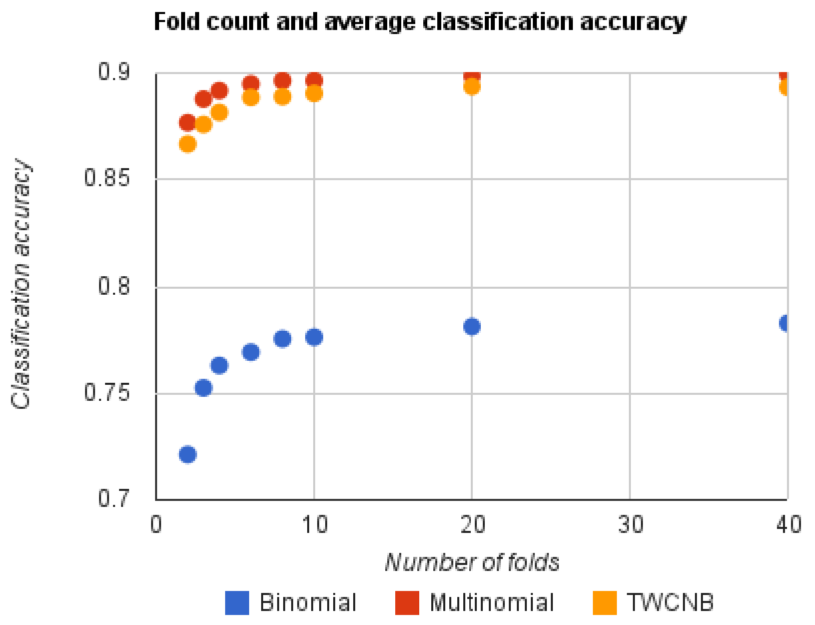


Figure 2. *Classification accuracy varying fold numbers for binomial (multivariate), multinomial, and TWCNB.*

The following table lists the accuracies for 400 docs (first 20 docs from each class) against the average accuracy for cross-validation. We expect validation accuracies to decrease with k-fold cross-validation for two reasons. First, since it reduces the probabilities for overfitting the training data, cross-validation results in a classifications that are less accurate but much more representative of the results for an unknown test set. Secondly, cross-validation, depending on the k-value, tests varying relative sizes of training vs. testing sets. We see that, compared to the 400 docs, we are classifying a much larger test set, relative to the training set. This may also contribute to poorer accuracy.

	400 docs	**With 10-fold**
Multinomial		
baseline	96.5	89.7
chi-squared	91.8	85.5
Multivariate		
baseline	84.8	77.6
chi-squared	87.0	83.1

Table 2: *Classification accuracy comparing use of k-fold and chi-squared*

In general, results were in line with our expectations. Importantly, we note that chi-squared feature selection results in the same increase in improvement for Multivariate Bayes and the same decrease in improvement for Multinomial Bayes.

Implementation notes: With cross-validation (or any other testing where the test group is not a subset of the training group), the classifier may see tokens that have never been seen. Our implementation ignored these tokens because they tell us nothing about correlation with a class. A common alternative, adding a small constant, would be equivalent to adding a constant to the score for every class, having no benefit for classification.

5: Transformed Weight-normalized Complement Naïve Bayes

In order to test further improvements to the Naive Bayes classifier, we implement Complement Naïve Bayes (CNB), Weight-normalized Complement Naïve Bayes (WCNB) and finally Transformed Weight-normalized Complement Naïve Bayes. These steps add to Naive Bayes by incorporating tf-idf as well as estimating the parameters of a class using all other classes (the “complement”). Our accuracies for 10-fold cross validation are shown below.

Classifier	**Accuracy (400 docs total)**	**Accuracy (cross validation k = 10)**
Multinomial NB	386	89.65
CNB	384	89.7
WCNB	386	89.5
WCNB + Transform 1	385	89.7
WCNB + Transform 1, 2	389	88.5
TWCNB (Transforms 1, 2, 3)	386	89.2

Table 3: *Implementation of transforms and weight normalization to standard MNB.*

Surprisingly, we saw very little improvement from the techniques given in the paper. We tested extensively using the top 20 docs from each class and through cross-validation with k=10. We tested less extensively with k=2, 5, and 20, but saw relatively similar results.

We saw negligible improvement from Complement Naive Bayes, but this is presumably because our training set is not very skewed at all. Classes all tend to have between 900-1000 documents.

Additionally, normalizing the weight vectors also seemed to have negligible effect. This must mean that these particular training sets and test sets are together varied enough where the independence assumptions of Naive Bayes do not result in too much accidental overweighting for certain terms, or results in similar accidental overweighting for many classes. Either of these phenomena would result in classification accuracy similar to normal MNB, which is what we saw.

We saw that transforming for term frequency improved the classification accuracy slightly, and we expect that this is due to the fact that some terms appear very often and are correlated very strongly with different classes. This effect is what makes our chi-squared feature selection effective. Transforming for document frequency, or terms that occur in many documents, did not seem to help at all. We suspect this is due to the fact that we are already employing stopwords; much of the improvement that this transform would've yielded has already been similarly affected by using stopwords. Finally, transforming for document length seemed to help a little bit. In a brief analysis of the corpus, we saw that there were wildly varying document lengths; thus it makes sense that the third transform is somewhat effective.

6: Domain Specific features

Tested Factor	Accuracy (k=10)
None	89.65
Upweighting subject (5x)	90.0
Upweighting subject (10x)	90.3
Upweighting subject (20x)	90.5
Upweighting subject (50x)	90.5
Upweighting subject (200x)	90.0
Removing signature from emails	89.65
Removing “from” field from emails	89.7
No stemming	90.0
No lowercasing	90.0
Overall top features (1000)	79.5
Overall top features (10000)	86.6

Table 4: *Trials on different methods for improving features.*

We consider several options given the particular nature of the newsgroups text. Since these are emails, we consider downweighting the `From:` line and upweighting the `Subject` line, following the idea of document zones. We found upweighting to some extent to be yield a great improvement in accuracy. Since subject headers are written to be extremely relevant, this improvement makes sense.

We then follow up with removing anything that comes after two dashes (--) as they likely pertain to signatures, and completely removing the “from” field. We consider lastly the idea that email sections that are quoted with ‘>’ are likely to be more pertinent/ relevant to the discussion. Removing the signature did not result in improvement, and upon further study of the corpus we realized the signature can contain many significant tokens (such as university department affiliation). However, removing the “from” field helped a little. We expect this is due to the fact that it lowered the weight of certain email addresses that were before weighted too highly.

We also tried removing lowercasing and stemming. Lowercasing and stemming are used to decrease the number of features trained on. However, as we found previously, Multinomial Bayes (and also TWCNB) seem to classify more accurately with more features. Our findings above support this conclusion, as removing lowercase and stemming both result in similar classification accuracy. Multivariate, on the other hand, seemed to classify less accurately. Another reason lowercasing and stemming might help is in the same way that lowercasing and stemming can be useful for spell-correction or for a corpus with lots of spelling variation: it reduces variation. Again, we found this to be relatively insignificant, since the corpus is of emails from an academic setting and the amount of mispelling is relatively minimal.

We reached the results above by testing using Multinomial Naive Bayes and cross-validation (k=10). We also tested on other fold values (k=2, 5, 20) and found similar results. In conclusion, we found that in general, very few considerations actually produced large results. We found the most single most helpful factor was upweighting.

7, 8: LibSVM and Weka implementation

We implemented a variety of classifiers using Weka, a Java-based data mining software. Weka wrappers for support vector machines with LibSVM and SMO (sequential minimal optimization) were also used. SMO is a popular SVM that uses a linear kernel - we maintained the complexity constant C as 1. Table 5 shows 10-fold (stratified) cross validation using Weka with different filters applied. For consistency as well as efficiency, we apply a Chi-squared feature selection filter (using **ChiSquaredAttributeEval**) that extracts 300 best ranked terms as features. (Note: we modified our MNB, MV, and TWCNB to use the same method for this part of the paper) This is a compromise, but it helps identify performance consistently.

Method	Accuracy	Time (sec)
MNB (Weka)	69.8375	0.09
MNB (ours)	58.50	2.309
MV (Weka)	65.6628	5.71
MV (ours)	65.50	3.097
TWCNB (ours)	68.95	2.383
3NN	61.939	0.03
LibSVM	59.289	60.47
SMO	68.951	140.32
DMNB	71.171	0.45

Table 5: *Classification results for LibSVM and other classifiers using Weka.*

High variance nonlinear classifiers like k-NN show that they work well at least for this data set, but may be overfitting at the same time. We see that the accuracy for SVM is fairly low, which is to be expected given the nonlinear nature of the data set and the feature selection limit. A separate attempt was made to use LibSVM and linear SVMs (using LibLinear) natively with Sally to convert the raw data, but the effort did not prove successful. We explain our attempts in the **Appendix**. In terms of accuracy to time ratio, we note that k-nearest neighbors and MNB seems to perform best.

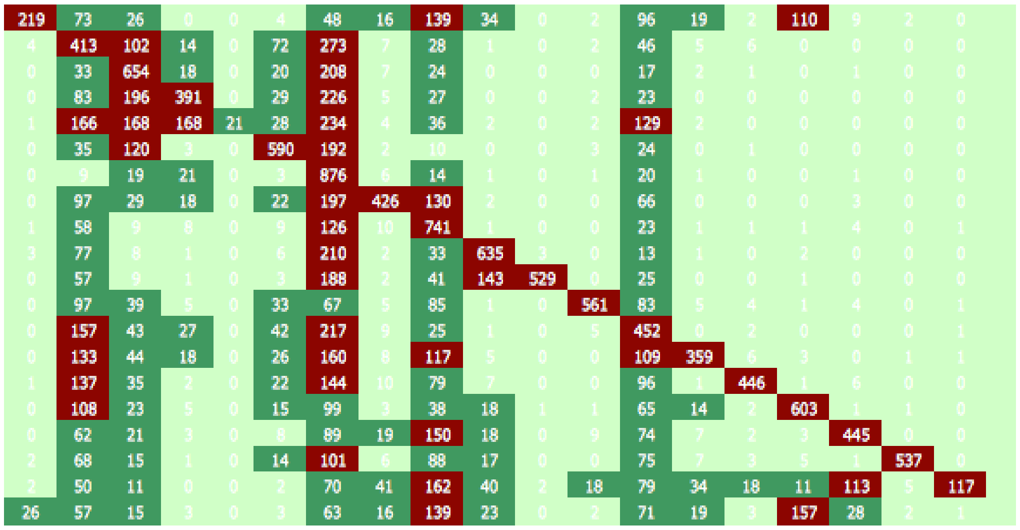
One of the surprises was the high performance of Discriminative Multinominal Naive Bayes, which is explained in [5]. DMNB essentially replaces the frequency estimation in Naive Bayes with a discriminative variant which updates token frequencies with the difference between true and predicted probabilities. DNMB ends up being significantly more accurate than our MNB, but similar in accuracy to TWCNB. Notably, the accuracy of our (Multivariate) Naive Bayes closely mirrors those of Weka. Also, the accuracy of our TWCNB is almost exactly the same as their MNB. This corroboration validates our implementations.

When run on the entire data set, we observe the following:

Method	Correctly classified	Time to train + test (sec)
Multinomial Naive Bayes	87.1043% (91.6773%)	37.17 + 7.95
Sequential minimal optimization[1]	83.8857%	249.96 + 147.38
Naive Bayes	74.9788%	361.48 + 1327.62
3NN	49.0971%	48.85 + 1665.76
LibSVM	47.8808%	1219.24

[1] This is based on John Platt’s algorithm where SVMs are trained using polynomial kernels.

As a purely pedagogic exercise we create a simple heat map using the Weka confusion matrix output with the rows corresponding to newsgroups as ordered in the first table. The columns indicate what each document was classified as with k-fold. We notice that there is a certain amount of misclassification taking place across certain similar groups (comp, sci) - for example, the third column indicates that many documents in `comp.graphics comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x` were being classified as comp.graphics.



219	73	26	0	0	4	48	16	139	34	0	2	96	19	2	110	9	2	0
4	413	102	14	0	72	273	7	28	1	0	2	46	5	6	0	0	0	0
0	33	654	18	0	20	208	7	24	0	0	0	17	2	1	0	1	0	0
0	83	196	391	0	29	226	5	27	0	0	2	23	0	0	0	0	0	0
1	166	168	168	21	28	234	4	36	2	0	2	129	2	0	0	0	0	0
0	35	120	3	0	590	192	2	10	0	0	3	24	0	1	0	0	0	0
0	9	19	21	0	3	876	6	14	1	0	1	20	1	0	0	1	0	0
0	97	29	18	0	22	197	426	130	2	0	0	66	0	0	0	3	0	0
1	58	9	8	0	9	126	10	741	1	0	0	23	1	1	1	4	0	1
3	77	8	1	0	6	210	2	33	635	3	0	13	1	0	2	0	0	0
0	57	9	1	0	3	188	2	41	143	529	0	25	0	0	1	0	0	0
0	97	39	5	0	33	67	5	85	1	0	561	83	5	4	1	4	0	1
0	157	43	27	0	42	217	9	25	1	0	5	452	0	2	0	0	0	1
0	133	44	18	0	26	160	8	117	5	0	0	109	359	6	3	0	1	1
1	137	35	2	0	22	144	10	79	7	0	0	96	1	446	1	6	0	0
0	108	23	5	0	15	99	3	38	18	1	1	65	14	2	603	1	1	0
0	62	21	3	0	8	89	19	150	18	0	9	74	7	2	3	445	0	0
2	68	15	1	0	14	101	6	88	17	0	0	75	7	3	5	1	537	0
2	50	11	0	0	2	70	41	162	40	2	18	79	34	18	11	113	5	117
26	57	15	3	0	3	63	16	139	23	0	2	71	19	3	157	28	2	1

Conclusion

In this exercise we were able to show the different improvements that could be applied to Naive Bayes classification, using algorithms such as TWCNB and domain-specific information such as zones. We compared these performance with SVMs and found in general that MNB-based algorithms were more efficient and accurate.

Resources

[1] Sally: A Tool for Embedding Strings in Vector Spaces http://www.mlsec.org/sally

[2] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[3] LibSVM: A library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm

[4] Rennie et. al. *Tackling the Poor Assumptions of Naive Bayes Text Classifiers.* Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

[5] Discriminative Multinominal Naive Bayes for Text Classification http://www.site.uottawa.ca/~stan/csi5387/DMNB-paper.pdf

Appendix

In order to save space here we list some of the features for some of the newsgroups. The implementation used in the project used the top 300 features.

Newsgroup	Top Features
alt.atheism	atheism islam moral livesey@solntze.wpd.sgi.com keith wpd jaeger jaeger@buphy.bu.edu
comp.graphics	imag tiff pov format viewer textur raytrac cview algorithm iff convert dxf vga rgb bezier
comp.os	microsoft file driver directori lourai panayiotaki ftp 3 zip win grp progman
comp.sys.ibm.pc.hardware	card scsi vlb gatewai drive eisa cach cmo disk dma port seagat maxtor rri ibm
comp.sys.mac.hardware	appl quadra powerbook duo simm nubu monitor adb vram upgrad scsi macus iivx

comp.windows.x	widget server window client suno applic sparc compil olwm lib usr twm
misc.forsale	ship offer condit obo cod stereo mint cassett mutant kou includ sega wolverin shatterstar

For reference the following lines perform MNB, converting the dataset into ARFF format requires the following command. ARFF describes data in the form of a list of documents sharing similar attributes. The `TextDirectoryLoader` class performs that transformation using a directory structure with folder names as labels.

```
java -cp weka.jar weka.core.converters.TextDirectoryLoader -dir /afs/
ir.stanford.edu/data/linguistic-data/TextCat/20Newsgroups/20news-18828 >
newstext.arff
```

Since Weka cannot handle string attributes we use a StringToWordVector filter to convert the strings to feature vectors.

```
java -cp weka.jar weka.classifiers.meta.FilteredClassifier -t newstext.arff
-F "weka.filters.unsupervised.attribute.StringToWordVector -S" -W
weka.classifiers.bayes.NaiveBayesMultinomial
```

With Sally we perform a similar transfom using delimiters on space and punctuation. We create vectors using tf-idf without any normalization.

```
sally --input_format dir --chunk_size 128 --ngram_len 1 --
ngram_delim "%0a%0d%20%22.,:;?" --vect_embed tfidf --vect_norm none --input_format
dir --output_format libsvm data data.libsvm
```

Finally, we perform training using LibLinear

```
train -v 5 -c 100 data.libsvm
```