Naive Bayes Project

Matthew Lueder

February 9, 2017

Load Packages

```
require('Rcpp')

## Loading required package: Rcpp

## Warning: package 'Rcpp' was built under R version 3.2.4

require('inline')

## Loading required package: inline

## Warning: package 'inline' was built under R version 3.2.5

##
## Attaching package: 'inline'

## The following object is masked from 'package:Rcpp':
##
##     registerPlugin

require('microbenchmark')

## Loading required package: microbenchmark

## Warning: package 'microbenchmark' was built under R version 3.2.5

require('caret')

## Loading required package: caret

## Warning: package 'caret' was built under R version 3.2.4

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.4
```

Read in training data

Training data will be loaded into a list, with each element being a list of words for that training example. The first word for each line of input data is the label. This is removed and stared in the lbls vector. A vector containing each class is created (classes).

```
# Set working directory to project folder
setwd("C:/Users/Matthew/Dropbox/Machine Learning/Projects/Project2")

training = scan("T_S.data", what = character(), fill = T, sep = '\n')
training = lapply(training, strsplit, split ="[ ]")
lbls = unlist(lapply(training, function(l) unlist(l)[1]))
classes = unique(lbls)
training = lapply(training, function(l) unlist(l)[-1])    # Remove first
element (the label)
```

Build Vocabulary

The vocabulary is the set of words which have appeared at least once in the training data

```
vocab = unique(unlist(training))
```

Calculate P(Class) for each class

We create a vector of the probabilities of seeing each class based on the frequency of appearance of that class in the training data.

$$P(Class_i) = \frac{Num.\ of\ training\ examples\ of\ in\ class\ i}{Total\ number\ of\ training\ examples}$$

```
P.Class = sapply(classes, function(class) length(lbls[lbls==class])) /
length(lbls)
```

Combine training examples of same class

Create a list with one index for each class and assign each index all words contained in all training examples of the respective class

```
Texts = sapply(classes, function(class) unlist(training[lbls == class]))
```

Count words for each class

1) Create an empty matrix for counting the words of each class
2) Create a C++ function to words quickly
3) Apply the C++ function A C++ function was used because counting words with an R loop took an extraordinary amount of time. We compare the performance of this C++ function to R in the next code chunk.

```
# 1)
counts = vector(mode = "integer", length = length(vocab))
names(counts) = vocab
counts = t(sapply(classes, function(class) counts))

# 2)
funcSrc <- '
#include <map>
```

```
IntegerMatrix counts(counts_in);
CharacterVector classes(classes_in);
CharacterVector vocab(vocab_in);
List texts(texts_in);

// Create map to look up position of words/class indices
std::map<std::string, int> wordIndex;
std::map<std::string, int> classIndex;
for (int i=0; i != vocab.size(); ++i)
{
  wordIndex[as<std::string>(vocab[i])] = i;
}
for (int i=0; i != classes.size(); ++i)
{
  classIndex[as<std::string>(classes[i])] = i;
}

// Count words
for (int i=0; i != classes.size(); ++i)
{
  CharacterVector text =
Rcpp::as<CharacterVector>(texts[as<std::string>(classes[i])]);

  for (int j=0; j != text.size(); ++j)
  {
    counts( classIndex[as<std::string>(classes[i])],
wordIndex[as<std::string>(text[j])] )++;
  }
}
return counts;
'
count.words <- cxxfunction(sig = signature(counts_in="integer",
classes_in="character", texts_in="list", vocab_in="character"), funcSrc,
plugin = "Rcpp")

# 3)
counts = count.words(counts, classes, Texts, vocab)

rm(funcSrc)
```

Compare performance of C++ and R

Shows why C++ was nessesary, and how much it improved performace. First I reduce the dataset to 1/1000th of the size to allow R function to complete in reasonable time

```
Texts.short = lapply(Texts, function(words)
words[0:as.integer(length(words)/1000)])

count.words.inR <- function(counts, classes, Texts) {
   for(class in classes)
```

```
  {
    for(word in unlist(Texts[class]))
    {
      counts[class, word] = counts[class, word] + 1
    }
  }
}

microbenchmark( count.words(counts, classes, Texts.short, vocab),
                count.words.inR(counts, classes, Texts.short),
                times = 10 )

## Unit: milliseconds
##                                                expr        min          lq
##  count.words(counts, classes, Texts.short, vocab)   57.52659   58.92327
##     count.words.inR(counts, classes, Texts.short) 7770.37234 8155.74572
##        mean     median         uq        max neval cld
##    59.93083   60.13951   60.46035    63.7283    10  a
##  8495.30335 8359.92956 8499.68593 10362.9070    10   b

rm(count.words.inR, Texts.short)
```

Calculate word probabilities for each class

$$P(Word_k|Class_i) = (n_k + 1)/(n + |Vocabulary|)$$

n = total number of word positions in class i n_k = number of times Word k occurs in class

```
n = apply(counts, 1, sum)
n.v = n + length(vocab)   # = n + |Vocabulary|
P.words = (counts + 1) / n.v
Log.P.words = log(P.words)

rm(n, n.v, P.words)
```

Read in validation data

```
validation = scan("V_S.data", what = character(), fill = T, sep = '\n')
validation = lapply(validation, strsplit, split ="[ ]")
v.lbls = unlist(lapply(validation, function(l) unlist(l)[1]))
validation = lapply(validation, function(l) unlist(l)[-1])  # Remove first
element (the label)

# Remove words not in vocabulary
validation = lapply(validation, function(input) input[unlist(input) %in%
vocab])
```

Perform classification

```
classify <- function(example) {
  which.max(rowSums(cbind(Log.P.words[,unlist(example)], log(P.Class))))
}
```

```
PredictedClasses = factor(names(sapply(validation, classify)))
```

Collect metrics / Visualize results

```
confusionMatrix(PredictedClasses, factor(v.lbls))

## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         atheism autos baseball christianity cryptology
##   atheism              242     0        0            8          0
##   autos                  0   369        0            0          1
##   baseball               0     2      364            0          0
##   christianity          40     0        2          376          0
##   cryptology             1     3        2            0        376
##   electronics            0     4        0            0          4
##   forsale                0     2        2            0          0
##   graphics               0     1        0            1          2
##   guns                   4     2        1            2          5
##   hockey                 1     0       17            0          0
##   mac                    0     0        0            0          0
##   medicine               3     0        0            1          1
##   mideastpolitics        8     2        2            2          0
##   motorcycles            1     3        1            0          0
##   mswindows              0     0        0            1          2
##   pc                     0     0        0            0          2
##   politics               6     5        3            3          2
##   religion              10     0        0            3          0
##   space                  3     2        1            1          0
##   xwindows               0     0        2            0          1
##                   Reference
## Prediction         electronics forsale graphics guns hockey mac medicine
##   atheism                    0       0        0    0      0   0        7
##   autos                      9      19        0    0      0   3        3
##   baseball                   0       2        0    0      1   1        0
##   christianity               2       2        1    1      3   0       10
##   cryptology                48       2       14    5      1   6        0
##   electronics              266      10        4    0      0  12        4
##   forsale                    1     261        0    1      0   5        0
##   graphics                  12       5      316    0      0   9        8
##   guns                       0       6        0  335      0   2        5
##   hockey                     0       1        0    0    389   0        0
##   mac                        6      22       10    0      0 293        1
##   medicine                   6       5        2    0      0   4      336
##   mideastpolitics            3       0        1    1      1   0        5
##   motorcycles                5       5        0    1      0   1        0
##   mswindows                  4       2        5    0      0   7        1
##   pc                        28      38       10    0      0  34        1
##   politics                   0       3        0   15      2   2       12
##   religion                   0       0        1    3      0   0        0
```

```
##   space                    3       5         8    2       2   4         3
##   xwindows                 0       2        17    0       0   2         0
##                   Reference
## Prediction        mideastpolitics motorcycles mswindows  pc politics
##   atheism                       7           0         2   0        6
##   autos                         1          14         0   0        0
##   baseball                      1           2         1   0        0
##   christianity                  4           1         2   0        2
##   cryptology                    4           0        19   6        5
##   electronics                   0           1         1  26        0
##   forsale                       0           1         0   5        0
##   graphics                      0           0        34   8        0
##   guns                          6           1         0   0      100
##   hockey                        0           0         0   0        0
##   mac                           0           0         8  30        0
##   medicine                      0           1         1   0        1
##   mideastpolitics             338           0         0   0        3
##   motorcycles                   2         372         0   2        0
##   mswindows                     0           0       245  23        0
##   pc                            0           0        33 290        0
##   politics                     13           4        13   0      184
##   religion                      0           0         1   0        0
##   space                         0           1         3   1        9
##   xwindows                      0           0        30   1        0
##                   Reference
## Prediction        religion space xwindows
##   atheism               43     1        0
##   autos                  0     0        0
##   baseball               0     0        1
##   christianity          68     1        0
##   cryptology             0     1        9
##   electronics            0     5        1
##   forsale                0     0        1
##   graphics               4     8       42
##   guns                  21     1        2
##   hockey                 0     0        0
##   mac                    0     0        6
##   medicine               2     4        1
##   mideastpolitics        3     2        2
##   motorcycles            0     0        1
##   mswindows              0     0        8
##   pc                     0     0        4
##   politics               6    15        2
##   religion              98     1        0
##   space                  6   352        4
##   xwindows               0     3      308
##
## Overall Statistics
##
##                Accuracy : 0.8117
```

```
##                  95% CI : (0.8027, 0.8205)
##     No Information Rate : 0.053
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8017
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: atheism Class: autos Class: baseball
## Sensitivity                 0.75862      0.93418         0.91688
## Specificity                 0.98973      0.99299         0.99846
## Pos Pred Value              0.76582      0.88067         0.97067
## Neg Pred Value              0.98932      0.99634         0.99539
## Prevalence                  0.04238      0.05248         0.05274
## Detection Rate              0.03215      0.04902         0.04836
## Detection Prevalence        0.04198      0.05567         0.04982
## Balanced Accuracy           0.87418      0.96358         0.95767
##                      Class: christianity Class: cryptology
## Sensitivity                      0.94472           0.94949
## Specificity                      0.98050           0.98233
## Pos Pred Value                   0.73010           0.74900
## Neg Pred Value                   0.99686           0.99715
## Prevalence                       0.05288           0.05261
## Detection Rate                   0.04995           0.04995
## Detection Prevalence             0.06842           0.06669
## Balanced Accuracy                0.96261           0.96591
##                      Class: electronics Class: forsale Class: graphics
## Sensitivity                     0.67684        0.66923         0.81234
## Specificity                     0.98991        0.99748         0.98123
## Pos Pred Value                  0.78698        0.93548         0.70222
## Neg Pred Value                  0.98233        0.98220         0.98968
## Prevalence                      0.05221        0.05181         0.05168
## Detection Rate                  0.03534        0.03468         0.04198
## Detection Prevalence            0.04491        0.03707         0.05978
## Balanced Accuracy               0.83338        0.83335         0.89678
##                      Class: guns Class: hockey Class: mac Class: medicine
## Sensitivity              0.92033       0.97494    0.76104         0.84848
## Specificity              0.97794       0.99733    0.98838         0.99551
## Pos Pred Value           0.67951       0.95343    0.77926         0.91304
## Neg Pred Value           0.99588       0.99860    0.98713         0.99162
## Prevalence               0.04836       0.05301    0.05115         0.05261
## Detection Rate           0.04451       0.05168    0.03893         0.04464
## Detection Prevalence     0.06550       0.05420    0.04995         0.04889
## Balanced Accuracy        0.94914       0.98614    0.87471         0.92200
##                      Class: mideastpolitics Class: motorcycles
## Sensitivity                         0.89894            0.93467
## Specificity                         0.99511            0.99691
## Pos Pred Value                      0.90617            0.94416
## Neg Pred Value                      0.99469            0.99635
```

```
## Prevalence                              0.04995            0.05288
## Detection Rate                          0.04491            0.04942
## Detection Prevalence                    0.04955            0.05234
## Balanced Accuracy                       0.94702            0.96579
##                        Class: mswindows Class: pc Class: politics
## Sensitivity                     0.62341   0.73980         0.59355
## Specificity                     0.99257   0.97898         0.98531
## Pos Pred Value                  0.82215   0.65909         0.63448
## Neg Pred Value                  0.97953   0.98561         0.98259
## Prevalence                      0.05221   0.05208         0.04119
## Detection Rate                  0.03255   0.03853         0.02445
## Detection Prevalence            0.03959   0.05846         0.03853
## Balanced Accuracy               0.80799   0.85939         0.78943
##                        Class: religion Class: space Class: xwindows
## Sensitivity                    0.39044      0.89340         0.78571
## Specificity                    0.99739      0.99187         0.99187
## Pos Pred Value                 0.83761      0.85854         0.84153
## Neg Pred Value                 0.97935      0.99410         0.98827
## Prevalence                     0.03335      0.05234         0.05208
## Detection Rate                 0.01302      0.04676         0.04092
## Detection Prevalence           0.01554      0.05447         0.04862
## Balanced Accuracy              0.69391      0.94263         0.88879
```

Overall accuracy = 81.04% Let's see if we can do better.

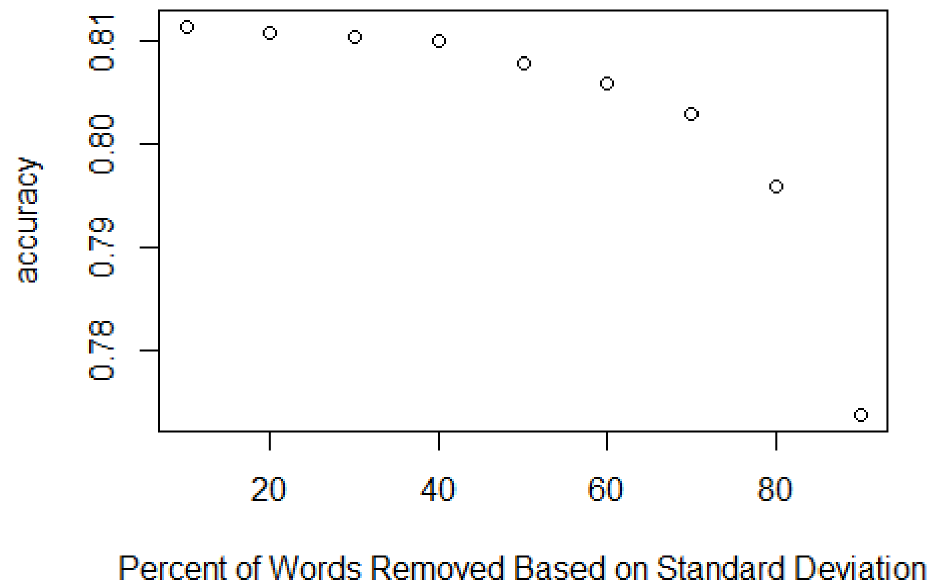
Eliminate words which have a probability with a standard deviation lower than a certain amount

```
Log.P.words.store = Log.P.words
accuracy = c()

for (sdp in seq(.1,.9,.1))
{
  Log.P.words = Log.P.words.store
  sdForEachWord = apply(exp(Log.P.words), 2, sd)
  wordsToElim = which(sdForEachWord < quantile(sdForEachWord, sdp))
  Log.P.words = Log.P.words[,-wordsToElim]
  vs.vocab = vocab[-wordsToElim]
  vs.validation = lapply(validation, function(input) input[unlist(input) %in%
vs.vocab])
  PredictedClasses = factor(names(sapply(vs.validation, classify)))
  accuracy = c( accuracy, confusionMatrix(PredictedClasses,
factor(v.lbls))$overall["Accuracy"])
}

rm(sdp, sdForEachWord, wordsToElim, vs.validation, vs.vocab)
```

See if we did any better

```
accuracy
```

```
##   Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8113458 0.8108144 0.8102830 0.8100173 0.8078916 0.8057659 0.8028431
##   Accuracy  Accuracy
## 0.7959346 0.7737478

plot(seq(10,90,10), accuracy, xlab = "Percent of Words Removed Based on
Standard Deviation")
```

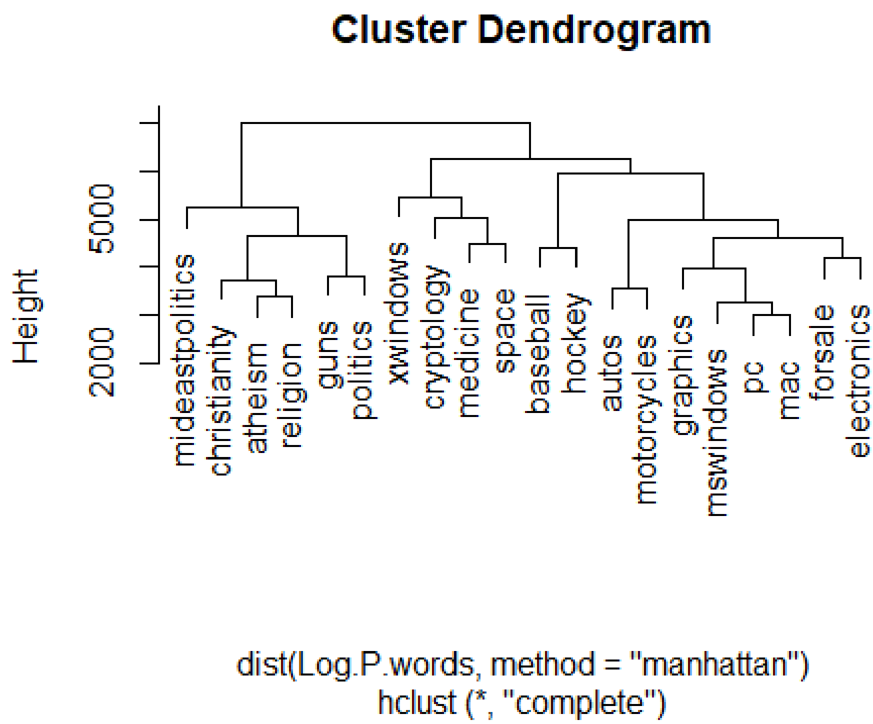


Removing features based on standard deviation hurts the performance of the classifier after removing more than 20% of the words. Removing 10% helped the performance of the classifier but only an extremely small, insignificant amount.

The classes seemed to be related, ex. Christianity, atheism, religion. Maybe it makes sense to group together certain classes. Let's see what hierarchical clustering tells us.

Perform hierarchical clustering

```
clusters = hclust(dist(Log.P.words, method = "manhattan"))
plot(clusters)
```




```
Log.P.words = Log.P.words.store
```

Looking at the results we see that a lot of our clusters make sense. We can use these results to guide us in how we group classes. Christianity, atheism, and religion are closely clustered so we will group them into a single class, religion. xwindows, graphics, mswindows, pc, and mac are closely clustered, so we will group them into a computing class. Cryptology, medicine and space are clustered together, so we will group them into a class named science. Baseball and hockey are clustered together, we will group them in a class named sports. Autos and motorcycles are clustered together, so we will group them in a class called vehicles.

Restructure training data

```
reclassify <- function(class) {
  if (class %in% c("atheism", "christianity")) {
    return("religion")
  }
  if (class %in% c("xwindows", "graphics", "mswindows", "pc", "mac")) {
    return("computing")
  }
  if (class %in% c("cryptology", "medicine", "space")) {
    return("science")
  }
  if (class %in% c("baseball", "hockey")) {
    return("sports")
  }
  if (class %in% c("autos", "motorcycles")) {
```

```r
    return("vehicles")
  }
  return(class)
}

lbls = sapply(lbls, reclassify)
classes = unique(lbls)

P.Class = sapply(classes, function(class) length(lbls[lbls==class])) /
length(lbls)
Texts = sapply(classes, function(class) unlist(training[lbls == class]))

counts = vector(mode = "integer", length = length(vocab))
names(counts) = vocab
counts = t(sapply(classes, function(class) counts))
counts = count.words(counts, classes, Texts, vocab)

n = apply(counts, 1, sum)
n.v = n + length(vocab)   # = n + |Vocabulary|
P.words = (counts + 1) / n.v
Log.P.words = log(P.words)
rm(n, n.v, P.words)
```

Restructure validation data

```r
v.lbls = sapply(v.lbls, reclassify)
```

Perform classification with restructured data

And print results.

```r
PredictedClasses = factor(names(sapply(validation, classify)))
confusionMatrix(PredictedClasses, factor(v.lbls))

## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          computing electronics forsale guns mideastpolitics
##   computing              1846         138     132    1               1
##   electronics              14         175       5    0               0
##   forsale                   6           1     188    0               0
##   guns                      1           0       1  323               3
##   mideastpolitics           0           0       0    1             312
##   politics                  3           0       0   10              10
##   religion                  9           3       3   12              40
##   science                  63          48      14   16               5
##   sports                    1           0       3    0               2
##   vehicles                  8          28      44    1               3
##                    Reference
## Prediction          politics religion science sports vehicles
##   computing                0        9      28      5        4
```

```
##   electronics            0        0       6      0        1
##   forsale                0        0       0      0        1
##   guns                  94       16       5      0        0
##   mideastpolitics        2        7       3      1        1
##   politics             166        4       4      1        1
##   religion              26      906      44      5        4
##   science               22       21    1086      8        5
##   sports                 0        1       1    772        2
##   vehicles               0        4       9      4      774
##
## Overall Statistics
##
##                Accuracy : 0.8699
##                  95% CI : (0.8621, 0.8775)
##     No Information Rate : 0.2592
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8467
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: computing Class: electronics Class: forsale
## Sensitivity                    0.9462            0.44529        0.48205
## Specificity                    0.9430            0.99636        0.99888
## Pos Pred Value                 0.8530            0.87065        0.95918
## Neg Pred Value                 0.9804            0.97024        0.97245
## Prevalence                     0.2592            0.05221        0.05181
## Detection Rate                 0.2453            0.02325        0.02498
## Detection Prevalence           0.2875            0.02670        0.02604
## Balanced Accuracy              0.9446            0.72082        0.74047
##                      Class: guns Class: mideastpolitics Class: politics
## Sensitivity              0.88736                0.82979         0.53548
## Specificity              0.98325                0.99790         0.99543
## Pos Pred Value           0.72912                0.95413         0.83417
## Neg Pred Value           0.99421                0.99111         0.98035
## Prevalence               0.04836                0.04995         0.04119
## Detection Rate           0.04291                0.04145         0.02205
## Detection Prevalence     0.05885                0.04344         0.02644
## Balanced Accuracy        0.93530                0.91384         0.76546
##                      Class: religion Class: science Class: sports
## Sensitivity                   0.9360         0.9157        0.9698
## Specificity                   0.9777         0.9681        0.9985
## Pos Pred Value                0.8612         0.8432        0.9872
## Neg Pred Value                0.9904         0.9840        0.9964
## Prevalence                    0.1286         0.1576        0.1058
## Detection Rate                0.1204         0.1443        0.1026
## Detection Prevalence          0.1398         0.1711        0.1039
## Balanced Accuracy             0.9568         0.9419        0.9842
##                      Class: vehicles
```

```
## Sensitivity                       0.9760
## Specificity                       0.9850
## Pos Pred Value                    0.8846
## Neg Pred Value                    0.9971
## Prevalence                        0.1054
## Detection Rate                    0.1028
## Detection Prevalence              0.1162
## Balanced Accuracy                 0.9805
```

Overall accuracy = 0.8699, an improvement. Forsale and electronics were difficult to classify.

It is suggested in literature to remove class prior probabilities when classifying text documents. Let's see what happens when we do this.

Perform classification again, but drop P(Class) from the final equation

```
zeros = rep(0, length(classes))
classify2 <- function(example) {
  which.max(rowSums(cbind(Log.P.words[,unlist(example)], zeros)))
}

PredictedClasses = factor(names(sapply(validation, classify2)))
confusionMatrix(PredictedClasses, factor(v.lbls))
```

```
## Confusion Matrix and Statistics
##
##                  Reference
## Prediction        computing electronics forsale guns mideastpolitics
##   computing            1841         126     121    1               0
##   electronics            15         190       8    0               0
##   forsale                 7           1     197    0               0
##   guns                    2           0       2  324               3
##   mideastpolitics         0           0       0    1             314
##   politics                4           0       0   10              13
##   religion                9           3       3   11              37
##   science                63          46      12   16               4
##   sports                  2           0       3    0               2
##   vehicles                8          27      44    1               3
##                  Reference
## Prediction        politics religion science sports vehicles
##   computing              0        8      28      5        4
##   electronics            0        0       6      0        1
##   forsale                0        0       0      0        2
##   guns                  93       17       8      0        0
##   mideastpolitics        3        7       5      1        1
##   politics             170        4       5      1        2
##   religion              24      906      42      6        4
##   science               20       21    1082      7        4
##   sports                 0        1       1    772        2
##   vehicles               0        4       9      4      773
```

```
##
## Overall Statistics
##
##                Accuracy : 0.8727
##                  95% CI : (0.865, 0.8802)
##     No Information Rate : 0.2592
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8502
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: computing Class: electronics Class: forsale
## Sensitivity                    0.9436            0.48346        0.50513
## Specificity                    0.9475            0.99579        0.99860
## Pos Pred Value                 0.8627            0.86364        0.95169
## Neg Pred Value                 0.9796            0.97222        0.97363
## Prevalence                     0.2592            0.05221        0.05181
## Detection Rate                 0.2446            0.02524        0.02617
## Detection Prevalence           0.2835            0.02923        0.02750
## Balanced Accuracy              0.9455            0.73963        0.75186
##                      Class: guns Class: mideastpolitics Class: politics
## Sensitivity              0.89011                0.83511         0.54839
## Specificity              0.98255                0.99748         0.99460
## Pos Pred Value           0.72160                0.94578         0.81340
## Neg Pred Value           0.99435                0.99138         0.98087
## Prevalence               0.04836                0.04995         0.04119
## Detection Rate           0.04305                0.04172         0.02259
## Detection Prevalence     0.05965                0.04411         0.02777
## Balanced Accuracy        0.93633                0.91629         0.77149
##                      Class: religion Class: science Class: sports
## Sensitivity                   0.9360         0.9123        0.9698
## Specificity                   0.9788         0.9696        0.9984
## Pos Pred Value                0.8670         0.8486        0.9860
## Neg Pred Value                0.9904         0.9834        0.9964
## Prevalence                    0.1286         0.1576        0.1058
## Detection Rate                0.1204         0.1437        0.1026
## Detection Prevalence          0.1388         0.1694        0.1040
## Balanced Accuracy             0.9574         0.9409        0.9841
##                      Class: vehicles
## Sensitivity                   0.9748
## Specificity                   0.9851
## Pos Pred Value                0.8855
## Neg Pred Value                0.9970
## Prevalence                    0.1054
## Detection Rate                0.1027
## Detection Prevalence          0.1160
## Balanced Accuracy             0.9800
```

Overall accuracy = 0.8727, a slight improvement.