Supervised Learning methods with RTexTools

Loren Collingwood

Supervised learning for text data is a machine learning method where the user develops and trains a model, then uses that model to predict onto out of sample text. If done rigorously, it can save users countless hours of otherwise manual hand-coding of text documents.

The first step is to take a pre-existing corpus of text documents that have been manually classified into distinct categories or labels. In this case we will use a New York Times dataset.

The text are the columns Title and Subject, and the manually coded labels is Topic.Code.

```r
options(scipen = 999, digits = 4)

########################
#       Packages       #
########################
#install.packages("RTextTools")
library(RTextTools)
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

Step 1

Load NYT Media Data from within the RTextTools package.

```r
data(NYTimes)
head(NYTimes)
```

```
##   Article_ID     Date
## 1      41246 1-Jan-96
## 2      41257 2-Jan-96
## 3      41268 3-Jan-96
## 4      41279 4-Jan-96
## 5      41290 5-Jan-96
## 6      41302 7-Jan-96
##                                                                        Title
## 1                      Nation's Smaller Jails Struggle To Cope With Surge in Inmates
## 2                                     FEDERAL IMPASSE SADDLING STATES WITH INDECISION
## 3                    Long, Costly Prelude Does Little To Alter Plot of Presidential Race
## 4                            Top Leader of the Bosnian Serbs Now Under Attack From Within
## 5 BATTLE OVER THE BUDGET: THE OVERVIEW; LEADERS IN HOUSE DROP G.O.P. PLAN ON U.S. WORKERS
## 6                                         South African Democracy Stumbles on Old Rivalry
##                                                                                   Subject
## 1                                                  Jails overwhelmed with hardened criminals
## 2                                                    Federal budget impasse affect on states
## 3                                                 Contenders for 1996 Presedential elections
## 4                                                 Bosnian Serb leader criticized from within
## 5 Battle over budget: Republican leaders abandon strategy of using closed Government offices
## 6                                                          political violence in south africa
##   Topic.Code
## 1         12
## 2         20
## 3         20
## 4         19
## 5          1
## 6         19
```

Step 2

Then you want to look at the distribution of the topic code/label. Sometimes you might have a binary (1, 0), or like a -1, 0, 1. Here we have a wide distribution but that some topics, like code 15, 19, and 20, make up a larger share. Usually, machine learning errors will benefit these topics since there are more examples of their code in the data. Also, you need to have well defined and theorized topics, otherwise the classification often will not work well. Garbage in garbage out.

```
table(NYTimes$Topic.Code)
```

```
##
##   1   2   3   4   5   6   7   8  10  12  13  14  15  16  17  18  19  20  21  24
##  71  88 185  19  83  87  34  33  50 163  22  40 172 444  81  16 662 394  20  76
##  26  27  28  29  30  31  99
##  47  11  75 141  29  41  20
```

```
# Randomize the sample to reduce any chance of time dependency.

data <- NYTimes[sample(1:3100,size=3100,replace=FALSE),]
n <- nrow(data)
```

Step 3

Create a matrix object where we break this up eventually into a sparse matrix, etc. for memory purposes. Note we also use TfIdf weighting, stem words, and remove numbers. This automatically creates a document term matrix.

```
matrix <- create_matrix(cbind(data["Title"],data["Subject"]),
                        language="english",
                        removeNumbers=TRUE,
                        stemWords=FALSE,
                        weighting=tm::weightTfIdf)
```

Step 4

Create a container object of training set and testing set. The training set you will develop your model on, and your test set you will pretend is virgin and predict the codes/labels onto. However, because you know the true value of these texts, you can compare the guess versus the truth. You will use this to evaluate your model accuracy and whether you can then feel comfortable proceeding to real prediction onto documents with unknown labels.

```
container <- create_container(matrix,data$Topic.Code,
                              trainSize=1:2100,
                              testSize=2101:n,
                              virgin=FALSE)
```

Step 5

Now you will train the models using in this case the glmnet and svm machine learning algoriths. RTextTools has 8 algorithms to choose from – previously 9 but maximum entropy has been cut due to dependency issues. Some algorithms are older and do take quite a bit of time. So for now we will use GLMNET and SVM. The advantage of using multiple algorithms is ensemble methods. This can greatly improve your accuracy for a subset of documents. Finally, for now, we will use a one n-gram bag-of-words approach.

```
print_algorithms()
```

```
## [1] "BAGGING"  "BOOSTING" "GLMNET"   "NNET"     "RF"       "SLDA"     "SVM"
## [8] "TREE"
```

```
# Train your model this can take some time
models <- train_models(container, algorithms=c("GLMNET","SVM"))
```

```
## Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground

## Warning: from glmnet Fortran code (error code -34); Convergence for 34th lambda
## value not reached after maxit=100000 iterations; solutions for larger lambdas
## returned
```

Step 6

Classification. This is where you apply the results of your trained model (i.e., the word dog gets a high coefficient value for topic 19, etc.) and predict onto text with supposedly unknown text labels (even though secretly WE DO KNOW!)

```
results <- classify_models(container, models)
```

Step 7

Now look at the precision and recall overall and then by different topics. Recall is the number of relevant documents retrieved by a search divided by the total number of existing relevant documents. In other words, what proportion of actual positives was identified correctly?

Precision is the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search. In other words, what proportion of positive identifications was actually correct?

```r
# Create analytics out of results #
analytics <- create_analytics(container, results)
summary(analytics)
```

```
## ENSEMBLE SUMMARY
##
##        n-ENSEMBLE COVERAGE n-ENSEMBLE RECALL
## n >= 1                1.0              0.58
## n >= 2                0.6              0.69
##
##
## ALGORITHM PERFORMANCE
##
##    SVM_PRECISION       SVM_RECALL       SVM_FSCORE GLMNET_PRECISION
##           0.5707           0.4867           0.5063           0.5130
##    GLMNET_RECALL    GLMNET_FSCORE
##           0.2352           0.2841
```