R_hw2_qs4

steven

August 18, 2017

Question 2

Naive Bayes method

```
library(tm)
```

```
## Loading required package: NLP
```

```
#use the readerPlain function in the Reuter example
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),id=fname, language='en') }

# load training and test directory
train_dirs = Sys.glob("C:/Users/zjj27/Documents/GitHub/STA380/data/ReutersC50/C50train/*")
test_dirs = Sys.glob("C:/Users/zjj27/Documents/GitHub/STA380/data/ReutersC50/C50test/*")
```

Getting sparse matrix

```
# For training data set
file_list_train = NULL
labels_train = NULL
y_train = NULL
for(author in train_dirs) {
  author_name = tail(strsplit(author,split="/")[[1]],1)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list_train = append(file_list_train, files_to_add)
  labels_train = append(labels_train, rep(author_name, length(files_to_add)))
}

#using the for loop here, we got the author name for each txt file in training data set and al
so the file path toward each text file for each author in training set.

train_docs = lapply(file_list_train, readerPlain)
names(train_docs) = file_list_train
names(train_docs) = sub('.txt', '', names(train_docs))
my_corpus_train = Corpus(VectorSource(train_docs))
#we then combine all the text file in training set make it into corpus by using Corpus()

## for testing data set
file_list_test = NULL
labels_test = NULL
```

```
for(author in test_dirs) {
  author_name = tail(strsplit(author,split="/")[[1]],1)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list_test = append(file_list_test, files_to_add)
  labels_test = append(labels_test, rep(author_name, length(files_to_add)))
}

test_docs = lapply(file_list_test, readerPlain)
names(test_docs) = file_list_test
names(test_docs) = sub('.txt', '', names(test_docs))
my_corpus_test = Corpus(VectorSource(test_docs))
# we repeat what we did for training set and make a corpus for testing data
```

Data Preprocessing

```
# for train data
my_corpus_train = tm_map(my_corpus_train, content_transformer(tolower)) # make everything lowe
rcase
my_corpus_train = tm_map(my_corpus_train, content_transformer(removeNumbers)) # remove numbers
my_corpus_train = tm_map(my_corpus_train, content_transformer(removePunctuation)) # remove pun
ctuation
my_corpus_train = tm_map(my_corpus_train, content_transformer(stripWhitespace)) ## remove exce
ss white-space
my_corpus_train = tm_map(my_corpus_train, content_transformer(removeWords), stopwords("SMART")
)

# for test data
my_corpus_test = tm_map(my_corpus_test, content_transformer(tolower)) # make everything lowerc
ase
my_corpus_test = tm_map(my_corpus_test, content_transformer(removeNumbers)) # remove numbers
my_corpus_test = tm_map(my_corpus_test, content_transformer(removePunctuation)) # remove punct
uation
my_corpus_test = tm_map(my_corpus_test, content_transformer(stripWhitespace)) ## remove excess
 white-space
my_corpus_test = tm_map(my_corpus_test, content_transformer(removeWords), stopwords("SMART"))
```

Model Prediction and Accuracy

```
library('naivebayes')
# make corpus into document term matrix for train and test set
DTM_train = DocumentTermMatrix(my_corpus_train)
DTM_test = DocumentTermMatrix(my_corpus_test)

# they are special kind of sparse matrix format
class(DTM_train)
```

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
class(DTM_test)
```

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
## You can also inspect its entries
inspect(DTM_train[1:10,1:20])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 20)>>
## Non-/sparse entries: 48/152
## Sparsity           : 76%
## Maximal term length: 11
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs access accounts agencies announced bogus business called character
##   1       1        1        1         1     2        2      1         4
##   10      4        0        0         0     0        1      0         4
##   2       0        0        0         1     0        1      0         4
##   3       2        0        0         0     0        0      0         4
##   4       0        0        0         1     0        1      0         4
##   5       0        0        0         1     0        1      0         4
##   6       0        0        0         0     0        0      0         4
##   7       0        0        1         0     0        0      1         4
##   8       0        0        0         0     0        1      0         4
##   9       0        0        1         0     0        1      0         4
##     Terms
## Docs charged commission
##   1        1          2
##   10       0          0
##   2        0          0
##   3        0          0
##   4        0          0
##   5        0          0
##   6        0          0
##   7        0          5
##   8        1          2
##   9        1          2
```

```
inspect(DTM_test[1:10,1:20])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 20)>>
## Non-/sparse entries: 39/161
## Sparsity           : 80%
## Maximal term length: 10
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs aaron abandon accounting added address affect agency agree allowing
##   1      1       1          3     3       1      1      1     1        1
##   10     0       0          0     1       1      1      0     0        0
##   2      0       0          0     0       1      0      0     0        0
##   3      0       0          0     0       0      0      0     0        1
##   4      0       0          0     0       0      0      0     0        0
##   5      1       0          0     1       1      0      0     0        0
##   6      0       0          0     0       0      0      0     1        1
##   7      0       0          0     0       1      1      0     0        0
##   8      1       0          0     0       0      0      2     0        2
##   9      1       0          0     0       0      0      1     0        2
##     Terms
## Docs approaches
##   1           1
##   10          0
##   2           0
##   3           0
##   4           0
##   5           0
##   6           1
##   7           0
##   8           0
##   9           0
```

```
# we go on removing sparse terms in our sparse matrix for train and test set
DTM_train = removeSparseTerms(DTM_train, 0.975)
DTM_test = removeSparseTerms(DTM_test, 0.975)

# After that, we created our word frequency matrix for training and testing as data frame
X_train = as.data.frame(as.matrix(DTM_train))
X_test = as.data.frame(as.matrix(DTM_test))

# Now we try to intersect common columns that presents in both train and test set
# we create a new training set where it only contains common columns from both train and test
sets
common_cols = intersect(names(X_train), names(X_test))
X_train_2 =X_train[,c(common_cols)]

# we then build naive bayes model using the new training set
nb_train = naive_bayes(x=X_train_2, y= as.factor(labels_train),laplace=1)
train.pred = predict(nb_train, X_test)


count=0
for (i in 1:2500){
  if(train.pred[i]==labels_train[i]){
    count=count+1
  }
}
accuracy = count/2500
cat('Prediction accuracy for navie bayes method is ',accuracy)
```

```
## Prediction accuracy for navie bayes method is  0.1828
```

```
# we get an accuracy of 18.28% when we use naive bayes model to predict the author name in tes
t set.
```

Random Forest Method

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
# we use the new training data set that we got from previous section to perform the randomfore
st model with tree number = 100
rfmodel <- randomForest(x=X_train_2,y=factor(labels_train),ntree=100)
rf.pred = predict(rfmodel,newdata=X_test)
conf_matrix = table(rf.pred,labels_train)

# calculating the number of corrected prediction through all text files
count = 0
for(i in 1:dim(conf_matrix)[1]){
  count = count + conf_matrix[i,i]
}

cat('Prediction accuracy for Random Forest method is around', count/2500)
```

```
## Prediction accuracy for Random Forest method is around 0.6012
```

```
# We use 4 different values(50,100,150,200) to figure out the best tree number for random fore
st model and finally come out with our highest prediction with ntree=100, and the correspondin
g accuracy is around 60%.
```