R_hw2_qs4

steven

August 18, 2017

Question 2

Naive Bayes method

```
library(tm)
```

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

We defined a function called readerPlain to read the content of text files.

```
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),id=fname, language='en') }
```

At first, we loaded the training and test directory.

```
train_dirs = Sys.glob("~/Documents/STA380/ReutersC50/C50train/*")
test_dirs = Sys.glob("~/Documents/STA380/ReutersC50/C50test/*")
```

Getiing the Sparse matrix

For training 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)))
}
```

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

We then combined all the text file in training set make it into corpus by using Corpus()

```
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))
```

For testing set, we repeated what we did for training set and make a corpus for testing data.

```
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'))
```

```r
  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))
```

Data Preprocessing

For train data

```r
my_corpus_train = tm_map(my_corpus_train, content_transformer(tolower)) # make everything lowercase
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 punctuation
my_corpus_train = tm_map(my_corpus_train, content_transformer(stripWhitespace)) ## remove excess white-s
my_corpus_train = tm_map(my_corpus_train, content_transformer(removeWords), stopwords("SMART"))
```

After we removed numbers, punctuation and excess and stopwords, we got our new corpus for training set.

```r
# for test data
my_corpus_test = tm_map(my_corpus_test, content_transformer(tolower)) # make everything lowercase
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 punctuation
my_corpus_test = tm_map(my_corpus_test, content_transformer(stripWhitespace)) ## remove excess white-spa
my_corpus_test = tm_map(my_corpus_test, content_transformer(removeWords), stopwords("SMART"))
```

After we removed numbers, punctuation and excess and stopwords, we got our new corpus for testing set.

Model Prediction and Accuracy

```r
library('naivebayes')
```

Then We made our training and testing corpus into document term matrices

```r
DTM_train = DocumentTermMatrix(my_corpus_train)
DTM_test = DocumentTermMatrix(my_corpus_test)
```

By using class functions, we finally got our sparse matrix for the training and testing sets respectively. In addition, inspect function were used to access the values inside the sparse matrix.

```r
class(DTM_train)
```

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

```r
class(DTM_test)
```

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

```r
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 then went on removing some sparse terms in our sparse matrix for train and test set. After that, we converted our sparse matrices to word frequency matrices for training and testing in data frame type.

```
DTM_train = removeSparseTerms(DTM_train, 0.975)
DTM_test = removeSparseTerms(DTM_test, 0.975)

X_train = as.data.frame(as.matrix(DTM_train))
X_test = as.data.frame(as.matrix(DTM_test))
```

Now we tried to intersect common columns that present in both train and test set. We created 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 built naive bayes model using the new training set and then found out its model accuracy.

```
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
```

Finally, We got an accuracy of 18.28% when we use naive bayes model to predict the author names in test set.

Random Forest Method

We used the new training data set that we got from previous section to perform the randomforest model with tree number = 100.

```
library(randomForest)
```

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

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

```
set.seed(1)
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 used 4 different values(50,100,150,200) to figure out the best tree number for random forest model and finally came out with our highest prediction with ntree=100, and the corresponding accuracy is around 60%.