test3

Vicky

August 18, 2015

```
options(warn=-1)
library(knitr)
library(doBy)
```

```
## Loading required package: survival
```

```
library(ggplot2)
library(plyr)
library(XML)
library(foreach)
# Some helper functions
library(tm)
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
readerPlain = function(fname){
                readPlain(elem=list(content=readLines(fname)),
                          id=fname, language='en') }
# # get list of authors
author_dirs = Sys.glob('../data/ReutersC50/C50train/*')
author_dirs = lapply(author_dirs, function(x){substring(x, first=29)})


### get test articles from both test and training directory, do all the pre processing steps
setwd('/Users/vickyzhang/Documents/MSBA/predictive2/STA380/R') ## spend lots of time fixing a bug here l
test_dirs = Sys.glob(c('../data/ReutersC50/C50tests/*', '../data/ReutersC50/C50train/*'))
file_list = NULL
labels = NULL
for(author in test_dirs) {
    author_name = substring(author, first=29)
    files_to_add = Sys.glob(paste0(author, '/*.txt'))
    file_list = append(file_list, files_to_add)
    labels = append(labels, rep(author_name, length(files_to_add)))
}
head(labels)
```

```
## [1] "AaronPressman" "AaronPressman" "AaronPressman" "AaronPressman"
## [5] "AaronPressman" "AaronPressman"
```

```r
# Need a more clever regex to get better names here
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list

# Preprocessing, tokenization, data cleaning
my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everything lowercase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove numbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) # remove punctuation
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## remove excess white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords("SMART"))

DTM = DocumentTermMatrix(my_corpus)
DTM # some basic summary statistics
```

```
## <<DocumentTermMatrix (documents: 5000, terms: 44235)>>
## Non-/sparse entries: 858721/220316279
## Sparsity           : 100%
## Maximal term length: 45
## Weighting          : term frequency (tf)
```

```r
class(DTM)  # a special kind of sparse matrix format
```

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

```r
## You can inspect its entries...
#inspect(DTM[1:10,1:20])
DTM = removeSparseTerms(DTM, 0.975) # remove those that are 0 in 97.5% of the docs or more
DTM
```

```
## <<DocumentTermMatrix (documents: 5000, terms: 1386)>>
## Non-/sparse entries: 494509/6435491
## Sparsity           : 93%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```r
# Now a dense matrix
Z = as.matrix(DTM) # Y is test data matrix
#print(dim(Z))

# each row is the PCA of an author
X = Z[1:2500,]
Y = Z[2501:4000,]
pca_all_author = foreach(j=1:50, .combine='rbind') %do% {

  corpus = X[j:(j+49),] # don't forget the bracket around j+49
  corpus = corpus/rowSums(corpus)
  # all prepared. run PCA!
```

```
  pca_author = prcomp(corpus, scale=FALSE)
  pca_author = pca_author$rotation[order(abs(pca_author$rotation[,1]),decreasing=TRUE),1]
}
pca_all_author[,pca_all_author=0] = 0.00000001
rownames(pca_all_author) = author_dirs
```

```
prediction_pca = foreach(j=1:1000, .combine='rbind') %do% {
  y_test = Y[j,]

  # for each article in test set, get the inner products, get the predicted author, put into a list
  logprob = foreach(i = 1:50, .combine='rbind') %do% {
    product = y_test*log(pca_all_author[i,])
    product[product == -Inf] = 0.0000001
    sum(product, na.rm = TRUE)
  }

  # set the list of authors as row names of logprob
  rownames(logprob) = author_dirs
  logprob = t(logprob)
  # get the predicted author
  y_predict = names(logprob[,logprob == max(logprob)])
}
```

```
# labels_test = labels[2501:5000]
# labels_test[1]
# prediction_pca[1]
# accuracy_pca = foreach(j=1:1000, .combine='c') %do% {
#   if (prediction_pca[j] == labels_test[j]) {
#     result = TRUE
#   }
#   else {
#     result = FALSE
#   }
#   result
# }
# accuracy[accuracy == TRUE] = 1
# accuracy[accuracy == FALSE] = 0
#
# sum(accuracy_pca) # 2
```