BBC news articles: LSA versus PEC

Preprocessing the data

Download and preprocess the BBC news data from http://mlg.ucd.ie/files/datasets/bbc-fulltext.zip Preprocessing of the BBC News articles consists of taking all articles from the categories business and politics, and keeping only a small number of articles from the tech topic.

```
rm(list=ls())

#download the data
temp <- tempfile()
download.file("http://mlg.ucd.ie/files/datasets/bbc-fulltext.zip",temp,quiet = T)
con<-unzip(temp)

#category: business
busi<-con[grepl("business",con)]
docs<-character(0)
for(i in 1:length(busi)){
  doc<-paste0(readLines(busi[i]),collapse=' ')
  docs<-c(docs,doc)
}
#category: politics
pol<-con[grepl("politics",con)]
for(i in 1:length(pol)){
  doc<-paste0(readLines(pol[i]),collapse=' ')
  docs<-c(docs,doc)
}
#category: tech
tech<-con[grepl("tech",con)]
nout<-50 # how many articles to include
# set.seed(123)
# tech<-sample(tech,nout,replace=F)
for(i in 1:nout){
  doc<-paste0(readLines(tech[i]),collapse=' ')
  docs<-c(docs,doc)
}
#keep label for reference
label<-numeric(length(docs)); label[(length(docs)-nout+1):length(docs)]<-1
#shuffle
set.seed(1234)
ind<-sample.int(length(docs))
docs<-docs[ind]; label<-label[ind]

unlink(temp)
```

Text Corpus

Use R-Package Quanteda (Benoit K, Watanabe K, Wang H, Nulty P, Obeng A, Müller S, Matsuo A (2018). "quanteda: An R package for the quantitative analysis of textual data." Journal of Open Source Software, 3(30), 774. doi: 10.21105/joss.00774, https://quanteda.io.) to set up a corpus, to split in tokens, and compute the relative frequencies.

```
library(quanteda)
```

```
## Warning: package 'quanteda' was built under R version 4.0.4
```

```
mycorpus<-corpus(docs)
print(paste("Number of docs is",ndoc(mycorpus)))
```

```
## [1] "Number of docs is 977"
```

```
#tokenize
tk<-tokens(mycorpus, remove_punct=TRUE, remove_numbers = TRUE, remove_symbols = TRUE,
           split_hyphens=T)
#remove stopwords
tk<-tokens_select(tk, pattern = stopwords('english'),
                  case_insensitive = TRUE,selection='remove', min_nchar=3)
#dfm
mydfm<-dfm(tk, verbose = FALSE,stem=T,tolower=TRUE)
#keep only terms with more than two chars
mydfm<-dfm_select(mydfm,min_nchar = 3)
#keep only words with frequency >=10
mydfm<-dfm_trim(mydfm, min_termfreq = 10)
#take relative frequencies as entries
mydfm_w<-dfm_weight(mydfm, scheme = "prop")
mydfm_w
```

```
## Document-feature matrix of: 977 documents, 2,804 features (95.6% sparse).
##        features
## docs         iraqi       voter        turn      econom        issu      beyond
##   text1 0.03100775 0.005167959 0.002583979 0.005167959 0.007751938 0.005167959
##   text2 0          0.004291845 0           0.004291845 0.008583691 0
##   text3 0          0.003184713 0           0           0           0
##   text4 0          0           0           0.005181347 0           0
##   text5 0          0           0           0           0           0
##   text6 0          0           0           0.008547009 0.008547009 0
##        features
## docs         desper       secur      situat        iraq
##   text1 0.002583979 0.007751938 0.005167959 0.031007752
##   text2 0           0           0           0
##   text3 0           0           0           0
##   text4 0           0           0           0
##   text5 0           0           0           0
##   text6 0           0           0           0.004273504
## [ reached max_ndoc ... 971 more documents, reached max_nfeat ... 2,794 more features ]
```

Document-term matrix

Create the document term matrix from the preprocessed data.

```
#export as matrix
dtm<-as.matrix(mydfm_w)
print(paste("Document-term matrix of dimension",dim(dtm)[1],"x",dim(dtm)[2]))
```

```
## [1] "Document-term matrix of dimension 977 x 2804"
```

Latent semantic analysis (LSA)

Do the dimension reduction of tdm - LSA - to create document embeddings/ vectors. LSA corresponds to the uncenterred principal component analysis (PCA).

```
#LSA (corresponds to the uncenterred PCA)
lsa<-prcomp(dtm,center=F,scale=F)
nk<-5 # this is the dimension of the document embeddings/ number of the components
dirs<-lsa$rotation[,1:nk] # these are the weights for all the words in tdm to form the topics contained
```

Principal expectile components (PEC)

Now we apply asymmetric weighting (PEC algorithm maximizing the τ-variance of the projection) to the documents in dtm in order to underweight politics and business, and overweight the tech. Note, the algorithm is unsupervised: it does not know which articles are which category; it has to figure out which documents are to overweight to filter out the tech topic. Due to the highly unbalanced data, we take a high weight $\tau = 0.9999$.

```
#functions for PEC
source('functions_PCAAN.R')
#tau
tau<- 0.9999
#pec
set.seed(123)
output.pec0 = pec.k.ini(t(dtm), alpha = tau-0.5, nk=nk, lab.ini=dirs[,1:nk])
```

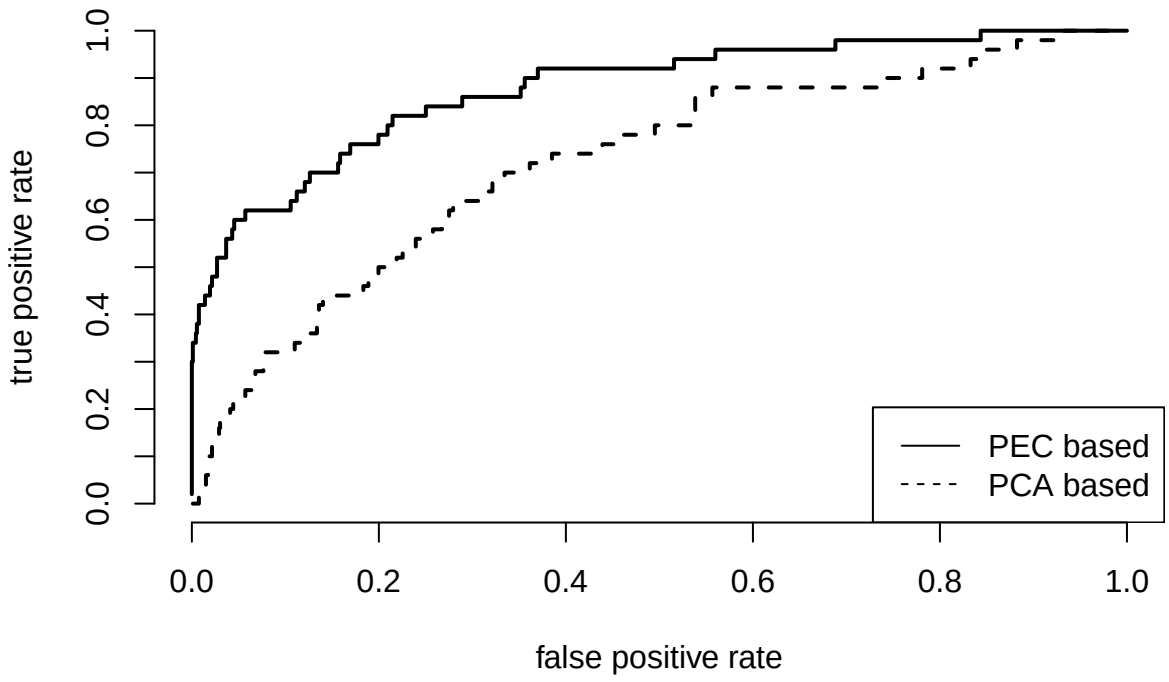
```
## [1] "computing for k =  1"
## [1] "computing for k =  2"
## [1] "computing for k =  3"
## [1] "computing for k =  4"
## [1] "computing for k =  5"
```

```
pecdirs<-output.pec0[[2]]
rownames(pecdirs)<-colnames(dtm)
```

ROC curves

Construct and plot the best ROC curves among the first five components as returned PEC and LSA procedures. The best ROC curves are chosen based on the approximation of the area under the curve.

```r
#roc function
roc <- function(scores, labels){
  labels <- labels[order(scores, decreasing=TRUE)]
  data.frame(fpr=cumsum(!labels)/sum(!labels),tpr=cumsum(labels)/sum(labels))
}
#ROC curves for LSA and PEC
scores_pec<-scores_pc<-roc_pec<-roc_pc<-list()
ar_pec<-ar_pc<-numeric(nk)
for(i in 1:nk){
  #PEC
  scores_pec[[i]]<-c(pecdirs[,i]%*%t(dtm))
  roc_pec[[i]]<-roc(scores_pec[[i]],label)
  ar_pec[i]<-mean(sample(scores_pec[[i]][label==1],10^6,replace=T) >
                    sample(scores_pec[[i]][label==0],10^6,replace=T)) # proxi for the area
  #LSA
  scores_pc[[i]]<-c(dirs[,i]%*%t(dtm))
  roc_pc[[i]]<-roc(scores_pc[[i]],label)
  ar_pc[i]<-mean(sample(scores_pc[[i]][label==1],10^6,replace=T) >
                    sample(scores_pc[[i]][label==0],10^6,replace=T))
}
#plot
plot(roc_pec[[which.max(ar_pec)]],type='l',col=1,lwd=2,xlab='false positive rate',ylab='true positive ra
axis(1); axis(2,at=seq(0,1,0.1))
lines(roc_pc[[which.max(ar_pc)]],col=1,lty=2,lwd=2)
legend('bottomright',legend=c('PEC based','PCA based'),lty=c(1,2),col=c(1,1),cex=1,border=NA)
```



Words from tech

```
print(paste("words with the highest weights by PEC are:",paste(rev(names(tail(sort(abs(pecdirs[,5])),20
```

```
## [1] "words with the highest weights by PEC are: microsoft; program; softwar; spywar; anti; user; viru
```