lab6

Credit: materials adapted from Patrick Chester, with some examples taken from Ken Benoit's NYU Dept. of Politics short course Fall 2014

1 Setting up

```
# Clear Global Environment
rm(list = ls())
setwd("/Users/Lingyi/TAD/lab/Text-as-Data-Lab-Spr2018/W6_02_27_18")

# Libraries
#devtools::install_github("kbenoit/readtext")

library(quanteda)
```

```
## quanteda version 1.0.0
```

```
## Using 3 of 4 threads for parallel computing
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

```
library(quanteda.corpora)
library(readtext)
```

1 Supervised Learning: Naive Bayes

```
# Example: Replication of 13.1 from IIR textbook

trainingset <- matrix(0,ncol=6,nrow=5)
trainingset[1,] <- c(1, 2, 0, 0, 0, 0)
trainingset[2,] <- c(0, 2, 0, 0, 1, 0)
trainingset[3,] <- c(0, 1, 0, 1, 0, 0)
trainingset[4,] <- c(0, 1, 1, 0, 0, 1)
trainingset[5,] <- c(0, 3, 1, 0, 0, 1)
colnames(trainingset) <- c("Beijing", "Chinese",  "Japan", "Macao", "Shanghai", "Tokyo")
rownames(trainingset) <- paste("d", 1:5, sep="")

trainingset <- as.dfm(trainingset) # training data

trainingclass <- factor(c("Y", "Y", "Y", "N", NA), ordered = TRUE)
# training/test classes -- last document is unknown
```

```
# replicate IIR p261 prediction for test set (document 5)
nb.p261 <- textmodel_nb(x = trainingset, y = trainingclass,
                        smooth = 1, prior="docfreq")
# Smooth gives values of 1 for new words; NB wouldn't work very well
pr.p261 <- predict(nb.p261)
pr.p261
```

```
## $log.posterior.lik
##            Y         N
## d1 -3.928188 -6.591674
## d2 -3.928188 -6.591674
## d3 -3.080890 -5.087596
## d4 -6.413095 -5.898527
## d5 -8.107690 -8.906681
##
## $posterior.prob
##            Y          N
## d1 0.9348373 0.06516267
## d2 0.9348373 0.06516267
## d3 0.8814994 0.11850060
## d4 0.3741233 0.62587672
## d5 0.6897586 0.31024139
##
## $nb.predicted
## [1] "Y" "Y" "Y" "N" "Y"
##
## $Pc
##    Y    N
## 0.75 0.25
##
## $classlabels
## [1] "Y" "N"
##
## $call
## predict.textmodel_nb(object = nb.p261)
```

2 Classification using Word Scores

```
# Read in conservative and labour manifestos
filenames <- list.files(path = "cons_labour_manifestos")

# Party name and year are in the filename -- we can use regex to extract these to use as our docvars
party <- unlist(regmatches(unlist(filenames), gregexpr("^[[:alpha:]]{3}", unlist(filenames))))
year <- unlist(regmatches(unlist(filenames), gregexpr("[[:digit:]]+", unlist(filenames))))

## get the data directory
#DATA_DIR <- system.file("extdata/", package = "readtext")

# This is how you would make a corpus with docvars from this data
cons_labour_manifestos <- corpus(readtext("cons_labour_manifestos/*.txt"))
docvars(cons_labour_manifestos, field = c("party", "year") ) <- data.frame(
  cbind(party, year))
```

```r
# We're going to use a dataframe
cons_labour_df <- data.frame(text = texts(cons_labour_manifestos),
                             party = party,
                             year = year,
                             stringsAsFactors = FALSE)

# Identifying test speech: Labor
test_speech <- cons_labour_df[46,]

# Setting training speeches: The remaining 45 Labor and Conservative speeches

training_df <- cons_labour_df[1:45, ]

# Create DFMs
training_dfm <- dfm(
  corpus(training_df$text,
         docvars = training_df[, c("party", "year")]))

test_dfm <- dfm(corpus(test_speech$text, docvars = test_speech[, c("party", "year")]))

# Train Word Score model

ws_base <- textmodel_wordscores(training_dfm,
                    y = (2 * as.numeric(training_df$party == "Lab")) - 1
# Y variable must be coded on a binary x in {-1,1} scale,
# so -1 = Conservative and 1 = Labour
                    )

# Look at strongest features
lab_features <- sort(ws_base$wordscores, decreasing = TRUE)

lab_features[1:10]
```

```
##          vested    co-operators           brain             die
##               1               1               1               1
##            tory        disgrace        bankrupt             yes
##               1               1               1               1
## enfranchisement      inequality
##               1               1
```

```r
con_features <- sort(ws_base$wordscores, decreasing = FALSE)

con_features[1:10]
```

```
## conclusive    triumph  displayed  prescribe   imperial  baldwin's
##         -1         -1         -1         -1         -1         -1
##    evident indirectly       rush assurances
##         -1         -1         -1         -1
```

```r
ws_base$wordscores[c("drugs", "minorities", "unemployment")]
```

```
##        drugs   minorities unemployment
##   -0.5337070    0.2757702    0.3393032
```

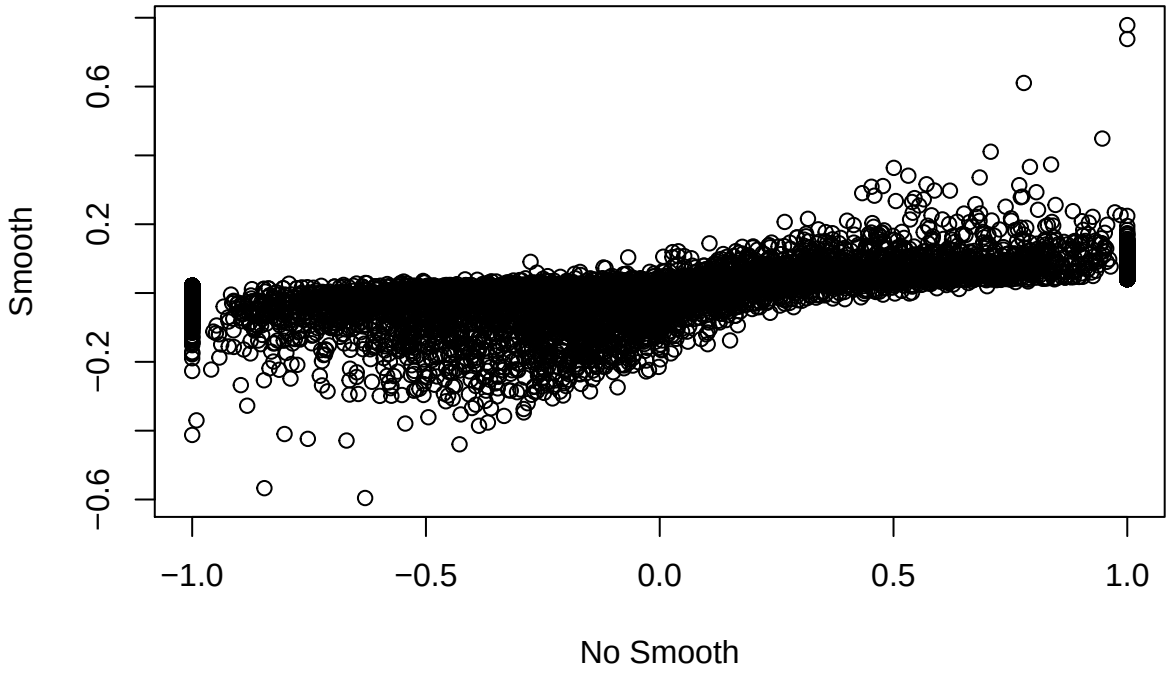
```r
# Trying it again with smoothing
ws_smooth <- textmodel_wordscores(training_dfm,
                          y = (2 * as.numeric(training_df$party == "Lab")) - 1,
                          smooth = 1)

ws_smooth$wordscores[c("drugs", "minorities", "unemployment")]
```

```
##        drugs   minorities unemployment
##  -0.15597370   0.09958915   0.18520351
```

```r
# Smoothing is giving stronger priors, decreasing the impact of new information

plot(ws_base$wordscores, ws_smooth$wordscores, xlim=c(-1, 1),
     xlab="No Smooth", ylab="Smooth")
```




```r
## predict that last speech
predict(ws_base, newdata = test_dfm,
        rescaling = "none", level = 0.95)
```

```
##       text1
## -0.03693925
```

```r
predict(ws_smooth, newdata = test_dfm,
        rescaling = "none", level = 0.95) # It makes the wrong prediction!
```

```
##       text1
## -0.08624139
```

3 Applying Naive Bayes and Word Scores to Amicus texts from Evans et al

```r
# Loading data
data("data_corpus_amicus")
```

```
amicus_dfm <- dfm(data_corpus_amicus)

amNBmodel <- textmodel_nb(amicus_dfm, docvars(data_corpus_amicus, "trainclass"))
amNBpredict <- predict(amNBmodel)

# "confusion matrix": Naive Bayes
tab_NB <- table(
  docvars(data_corpus_amicus, "testclass"),
  amNBpredict$nb.predicted)

tab_NB
```

```
##
##       P  R
##   AP 15  4
##   AR  5 74
```

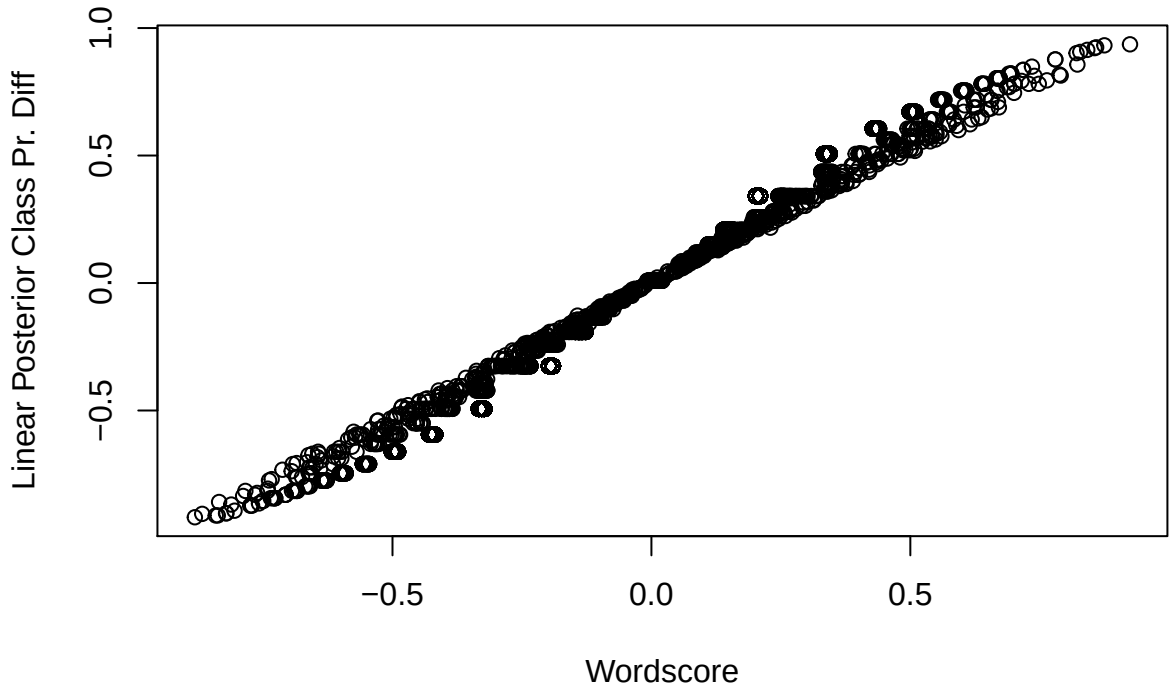
```
# Accuracy: Naive Bayes
(tab_NB[1,1]+tab_NB[2,2])/sum(tab_NB)
```

```
## [1] 0.9081633
```

```
reference <- c(1, 1, -1, -1, rep(NA, 98)) # class labels

amWSmodel <- textmodel_wordscores(amicus_dfm, reference, smooth = 1)

plot(amWSmodel$wordscores,
     c(1, -1) %*% amNBmodel$PcGw,
     xlab="Wordscore",
     ylab="Linear Posterior Class Pr. Diff")
```




```
(amWSpredict <- predict(amWSmodel))
```

```
##       sP1.txt       sP2.txt       sR1.txt       sR2.txt      sAP01.txt
```

```
##  0.0463611797  0.0441314298 -0.0578700916 -0.0326225179  0.0072442731
##     sAP02.txt     sAP03.txt     sAP04.txt     sAP05.txt     sAP06.txt
## -0.0038518723 -0.0011623859  0.0090533623 -0.0035788474  0.0048185544
##     sAP07.txt     sAP08.txt     sAP09.txt     sAP10.txt     sAP11.txt
## -0.0039522351 -0.0060324551 -0.0117224916 -0.0078865887 -0.0013982447
##     sAP12.txt     sAP13.txt     sAP14.txt     sAP15.txt     sAP16.txt
## -0.0043969975  0.0095243175  0.0025730719 -0.0035439035  0.0038302271
##     sAP17.txt     sAP18.txt     sAP19.txt     sAR01.txt     sAR02.txt
## -0.0154016733 -0.0090571857  0.0034205422 -0.0102261951  0.0004065718
##     sAR03.txt     sAR04.txt     sAR05.txt     sAR06.txt     sAR07.txt
## -0.0197155515 -0.0224776571 -0.0030425788 -0.0209559481 -0.0198296799
##     sAR08.txt     sAR09.txt     sAR10.txt     sAR11.txt     sAR12.txt
## -0.0177242249 -0.0150669289 -0.0220056887 -0.0086157898 -0.0132528041
##     sAR13.txt     sAR14.txt     sAR15.txt     sAR16.txt     sAR17.txt
## -0.0177605446 -0.0178823872 -0.0243155804 -0.0156863139 -0.0074407500
##     sAR18.txt     sAR19.txt     sAR20.txt     sAR21.txt     sAR22.txt
## -0.0184157603 -0.0050704101 -0.0066476276 -0.0089441748 -0.0210664296
##     sAR23.txt     sAR24.txt     sAR25.txt     sAR26.txt     sAR27.txt
## -0.0241688294 -0.0326156611 -0.0233249279 -0.0288108311 -0.0291769226
##     sAR28.txt     sAR29.txt     sAR30.txt     sAR31.txt     sAR32.txt
## -0.0182034750 -0.0141591268 -0.0189065008 -0.0188973090 -0.0074482941
##     sAR33.txt     sAR34.txt     sAR35.txt     sAR36.txt     sAR37.txt
## -0.0088058089 -0.0195370733 -0.0158132680 -0.0155265389  0.0003312207
##     sAR38.txt     sAR39.txt     sAR40.txt     sAR41.txt     sAR42.txt
## -0.0242922067 -0.0221411703 -0.0215422647 -0.0071186889 -0.0075155849
##     sAR43.txt     sAR44.txt     sAR45.txt     sAR46.txt     sAR47.txt
## -0.0305184700 -0.0309595632 -0.0136601662 -0.0235866315 -0.0089229987
##     sAR48.txt     sAR49.txt     sAR50.txt     sAR51.txt     sAR52.txt
## -0.0218971898 -0.0183941284 -0.0176363588 -0.0165273544 -0.0194470335
##     sAR53.txt     sAR54.txt     sAR55.txt     sAR56.txt     sAR58.txt
## -0.0052255693 -0.0179436107 -0.0077110690 -0.0143581765 -0.0086522559
##     sAR59.txt     sAR60.txt     sAR61.txt     sAR62.txt     sAR63.txt
## -0.0117094118 -0.0182767345 -0.0090923917 -0.0141087989 -0.0163563567
##     sAR64.txt     sAR65.txt     sAR66.txt     sAR67.txt     sAR68.txt
## -0.0211294835 -0.0211436529 -0.0108646755 -0.0179647946 -0.0107338512
##     sAR71.txt     sAR72.txt     sAR73.txt     sAR74.txt     sAR75.txt
## -0.0158756649 -0.0212776004 -0.0197323430 -0.0052725608 -0.0163203561
##     sAR76.txt     sAR77.txt     sAR78.txt     sAR79.txt     sAR80.txt
## -0.0184632105 -0.0154536036 -0.0114869806 -0.0002606598 -0.0196881229
##     sAR81.txt     sAR83.txt
## -0.0112007928 -0.0083545581
```

```
amWSresults <- ifelse(amWSpredict > 0, "P", "R")

# "confusion matrix": WordScores
(tab_WS <- table(docvars(data_corpus_amicus, "testclass"), amWSresults) )
```

```
##     amWSresults
##       P  R
##   AP  7 12
##   AR  2 77
```

```
# Accuracy: WordScores
(tab_WS[1,1]+tab_WS[2,2])/sum(tab_WS)
```

```
## [1] 0.8571429
```