Assigment 2

Pavel Linder, Nikita Brancatisano

12/26/2019

0. Read input

```
train = read.table(file = 'train.tsv', sep = '\t', header = TRUE, stringsAsFactors = FALSE)
test = read.table(file = 'test.tsv', sep = '\t', header = TRUE)
length(which(!complete.cases(train)))
```

```
## [1] 0
```

```
head(train$text_a)
```

```
## [1] "Xanax was her death blow. \\\\xc2\\\\xa0That stuff is totally dangerous because you
## [2] "you are both morons and that is never happening"
## [3] "you are just an idiot blabbermouth that is gonna get stopped HARD one day! You W
## [4] "how do the towers connect to the bottom pentagon?  Since it's not flat..."
## [5] "I love Cam Newton's upside, and think he'll be an All-Pro caliber QB, but 21 TD's
## [6] "Eat shit and die Andrew"
```

1. Cleaning data

Remove punctuation and stopwords

```
train$text_a = as.character(train$text_a)
train$text_a = tm::removePunctuation(train$text_a)
train$text_a = tm::removeWords(x = train$text_a, stopwords(kind = "SMART"))
train$text_a = tm::stripWhitespace(train$text_a)

#train$text_a = tolower(train$text_a)
word_count <- lapply(train$text_a, wordcount)
length(which(word_count > 100))
```

```
## [1] 23
```

```
length(which(word_count < 100))
```

```
## [1] 802
```

```
length(which(word_count == 0))
```

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

```
length(train$text_a)
```

```
## [1] 825
```

```
train <- train[which(word_count < 100),]
word_count <- lapply(train$text_a, wordcount)
length(train$text_a)
```

```
## [1] 802
```

```
train <- train[which(word_count > 0),]
length(train$text_a)
```

```
## [1] 801
```

```
head(train$text_a)
```

```
## [1] "Xanax death blow xc2xa0That stuff totally dangerous build tolerance quickly stop abruptly xc2xa0
## [2] " morons happening"
## [3] " idiot blabbermouth gonna stopped HARD day You WILL NOT saved"
## [4] " towers connect bottom pentagon Since flat"
## [5] "I love Cam Newtons upside hell AllPro caliber QB 21 TDs 17 Interceptions NFL Network 10th Heisma
## [6] "Eat shit die Andrew"
```

Anonymize proper nouns

```
n <- length(train$text_a)
word_ann <- Maxent_Word_Token_Annotator()
sent_ann <- Maxent_Sent_Token_Annotator()
pos_ann = Maxent_POS_Tag_Annotator()

for (i in 1:n) {
  while(1) {
    doc <- as.String(train$text_a[[i]])
    wordAnnotation <- annotate(doc, list(sent_ann, word_ann))
    POSAnnotation <- annotate(doc, pos_ann, wordAnnotation)
    POSWords <- subset(POSAnnotation, type == "word")
    POSTags <- vector()
    for (j in 1:length(POSWords$features))
      POSTags <- c(POSTags, POSWords$features[[j]]$POS)
    tokenPOS <- cbind(doc[POSWords], POSTags)
    ppn_idx <- which(tokenPOS[,2] == "NNP", 1)
    if (length(ppn_idx) == 0) {
      break;
    }
    words <- subset(wordAnnotation, type == "word")
    hashed <- digest(tokenPOS[ppn_idx, 1], "xxhash32")
    ppn <- words[ppn_idx]
    train$text_a[[i]] <- gsub(doc[ppn$start,ppn$end], hashed, doc)
  }
}

head(train$text_a)
```

```
## [1] "5dcac30f death blow xc2xa0That stuff totally dangerous build tolerance quickly stop abruptly xc
## [2] " morons happening"
## [3] " idiot blabbermouth gonna stopped e8a1d6c8 day You WILL 91b0cb01 saved"
## [4] " towers connect bottom pentagon Since flat"
## [5] "I love a9350e16 cee1217a upside hell ea737b57 caliber 9c894fe8 21 1a620f48 17 Interceptions 914
## [6] "d84ee5df shit die 2703f309"
```

Remove unknown symbols (non UTF-8 characters)

```
train$text_a <- iconv(train$text_a, to='UTF-8', sub='byte')
length(train$text_a)
```

```
## [1] 801
```

```
head(train$text_a)
```

```
## [1] "5dcac30f death blow xc2xa0That stuff totally dangerous build tolerance quickly stop abruptly xc
## [2] " morons happening"
## [3] " idiot blabbermouth gonna stopped e8a1d6c8 day You WILL 91b0cb01 saved"
## [4] " towers connect bottom pentagon Since flat"
## [5] "I love a9350e16 cee1217a upside hell ea737b57 caliber 9c894fe8 21 1a620f48 17 Interceptions 914
## [6] "d84ee5df shit die 2703f309"
```

2. Exploration

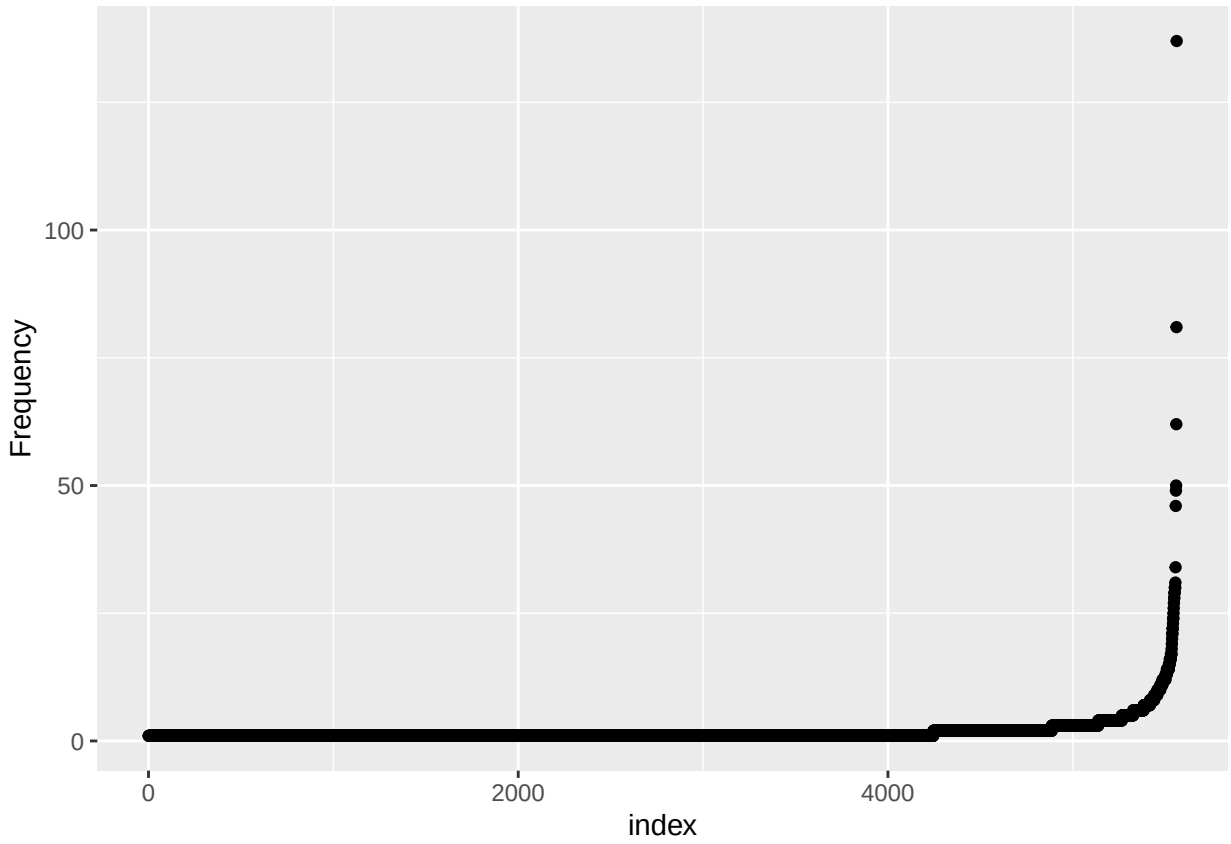
We are computing the TDM matrix from 2 corpuses: one with stemmization, one without. From the results we can see that the character of the corpus remains. Thus, we will use the stemmed corpus for the future evaluation. ### I. Plot the frequency of words (without stemmization)

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
corpus <- Corpus(VectorSource(train$text_a)) # turn into corpus
tdm <- TermDocumentMatrix(corpus)

wordFreq <- sort(rowSums(as.matrix(tdm)), decreasing=TRUE)
qplot(seq(length(wordFreq)),sort(wordFreq), xlab = "index", ylab = "Frequency")
```




```
findFreqTerms(tdm, lowfreq=50)
```

```
## [1] "dont"   "you"    "the"    "people"
```

```
mostFreq <- subset(wordFreq, wordFreq >= 50)
head(mostFreq, 10)
```

```
##    you   dont    the people
##    137     81     62     50
```

```
length(wordFreq)
```

```
## [1] 5562
```

```
length(wordFreq[wordFreq<10])
```

```
## [1] 5457
```

```
length(wordFreq[wordFreq<5])
```

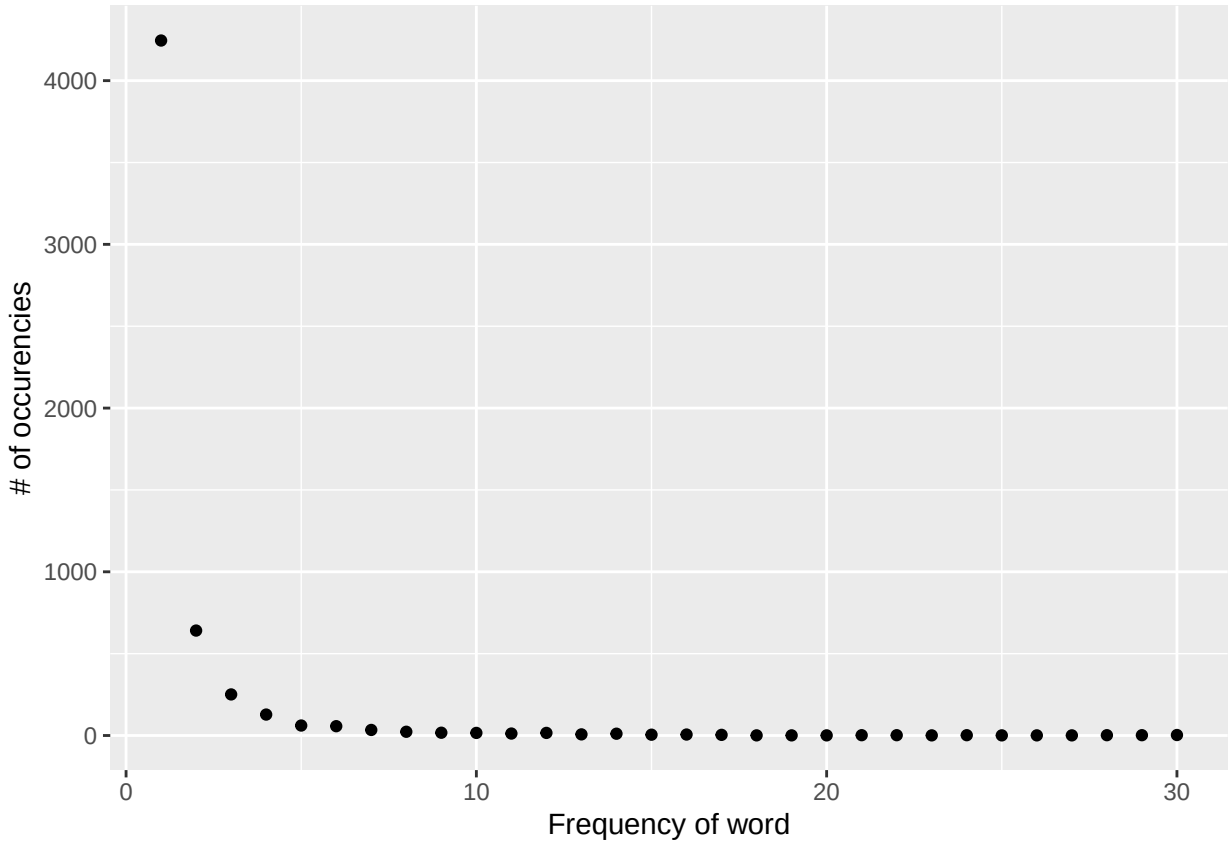
```
## [1] 5265
```

```
length(wordFreq[wordFreq==1])
```

```
## [1] 4245
```

```
freq <- sort(unique(wordFreq), decreasing=FALSE)
occ <- vector()
for (i in 1:length(freq)) {
  occ[i] <- length(wordFreq[wordFreq == freq[i]])
}
```

```
qplot(freq[1:30], occ[1:30], xlab = "Frequency of word", ylab = "# of occurencies")
```



I. Plot the frequency of words (with stemmization)

```
stemmed <- stemDocument(train$text_a, language = "english")
corpus2 <- Corpus(VectorSource(stemmed)) # turn into corpus
```

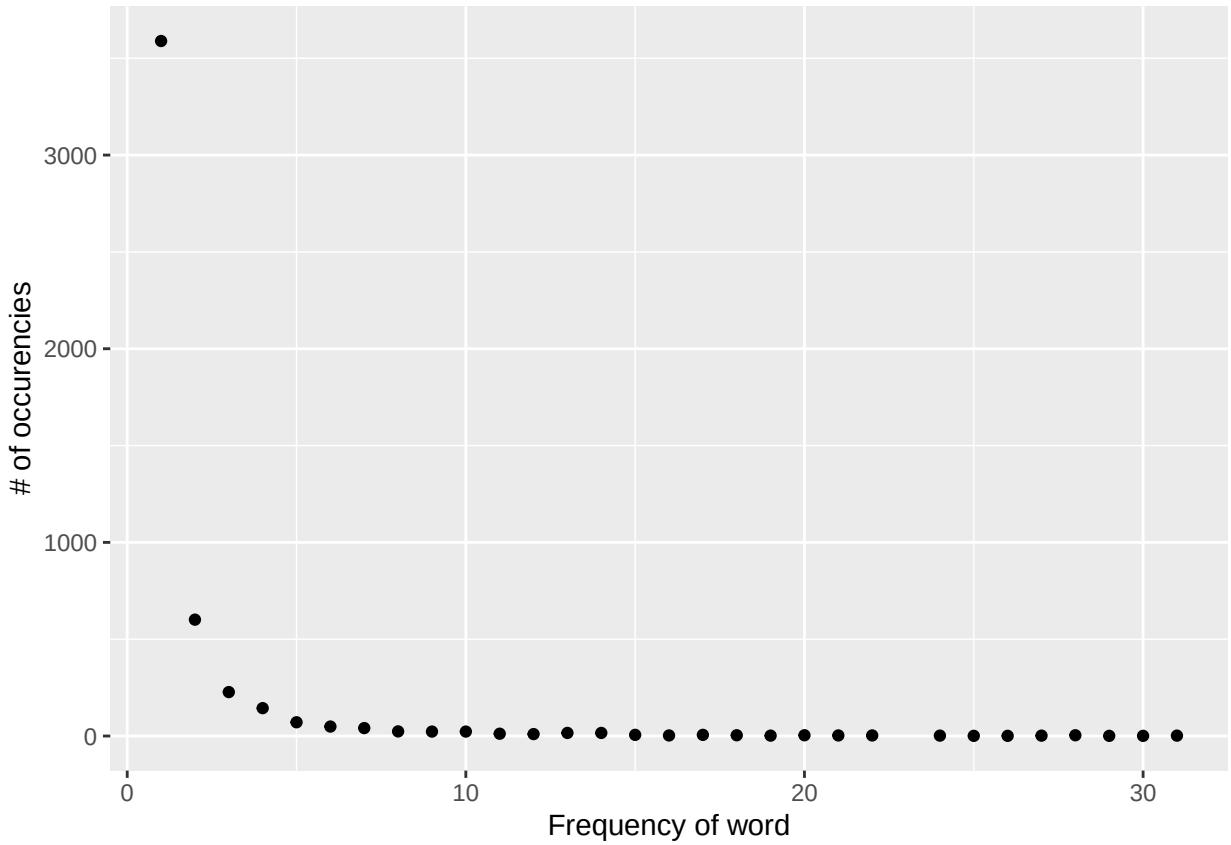
```
##  you dont fuck  the
##  137   81   79   62
```

```
## [1] 4904
```

```
## [1] 4769
```

```
## [1] 4561
```

```
## [1] 3589
```



II. Perform a clustering on the vectorized document space

We will use Weighted TF-IDF as a way to represent the document space:

```
tdm <- tm::DocumentTermMatrix(corpus2)
tdm.tfidf <- tm::weightTfIdf(tdm)
tdm.tfidf <- tm::removeSparseTerms(tdm.tfidf, 0.999)  # sparsity being not well handled overall in R
tfidf.matrix <- as.matrix(tdm.tfidf)
```

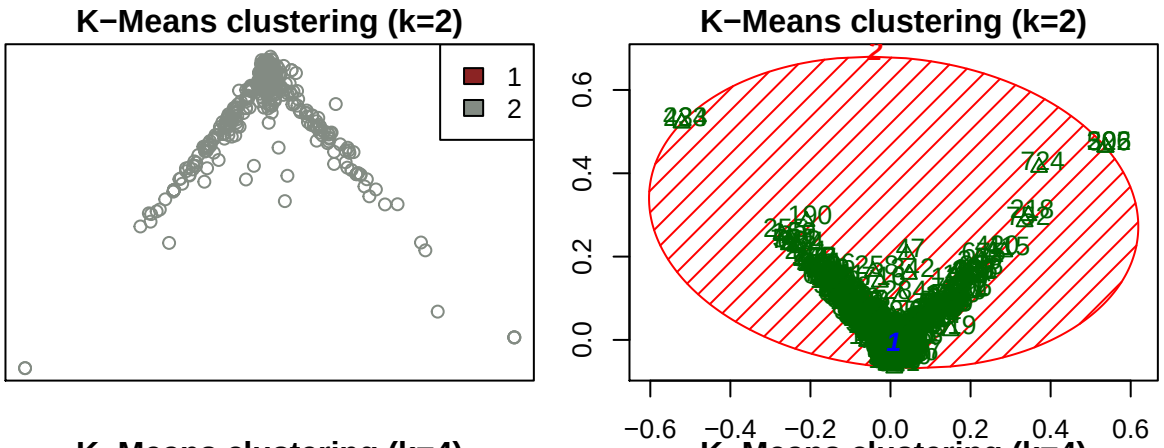
Afterwards, we perform k-means algorithm to cluster in {2,4,8,16} classes.

```
cluster2 <- kmeans(tfidf.matrix, centers=2)
cluster4 <- kmeans(tfidf.matrix, centers=4)
cluster8 <- kmeans(tfidf.matrix, centers=8)
cluster16 <- kmeans(tfidf.matrix, centers=16)

cluster2.master <- cluster2$cluster
cluster4.master <- cluster4$cluster
cluster8.master <- cluster8$cluster
cluster16.master <- cluster16$cluster
```

We perform Classical multidimensional scaling (SMC) to map the data (distance matrix) into 2D dimension and then visualize it.

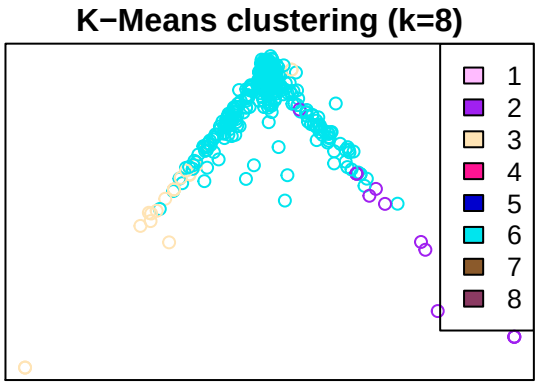
```
dist.matrix = proxy::dist(tfidf.matrix, method = "cosine")
points <- cmdscale(dist.matrix, k = 2)
previous.par <- par(mfrow=c(2,2), mar = rep(1.5, 4))
color <- grDevices::colors()[grep('gr(a|e)y', grDevices::colors(), invert = T)]
```

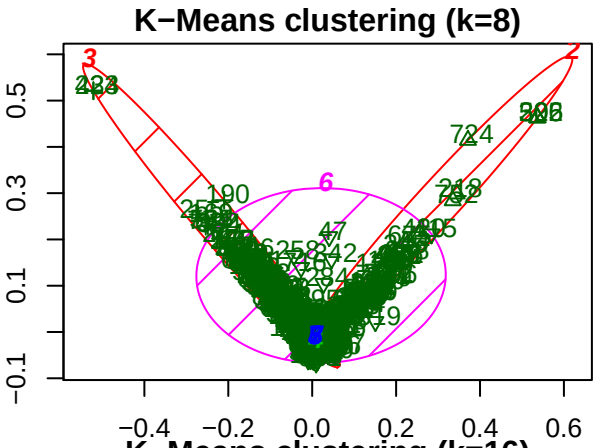
```
my_palette = sample(color, 2)
plot(points, main = 'K-Means clustering (k=2)', col = my_palette[as.factor(cluster2.master)],
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
legend("topright", sprintf("%s",seq(1,2)), fill = my_palette[1:2])
clusplot(points, cluster2.master, main='K-Means clustering (k=2)', color=TRUE, shade=TRUE, labels=2, li

my_palette = sample(color, 4)
plot(points, main = 'K-Means clustering (k=4)', col = my_palette[as.factor(cluster4.master)],
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
legend("topright", sprintf("%s",seq(1,4)), fill = my_palette[1:4])
clusplot(points, cluster4.master, main = 'K-Means clustering (k=4)', color=TRUE, shade=TRUE, labels=2,
```

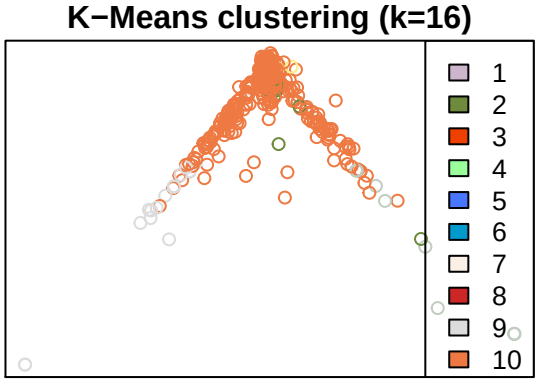


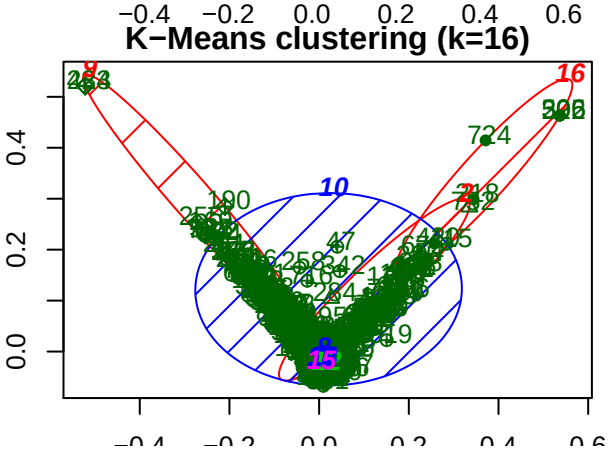

```
my_palette = sample(color, 8)
plot(points, main = 'K-Means clustering (k=8)', col = my_palette[as.factor(cluster8.master)],
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
legend("topright", sprintf("%s",seq(1,8)), fill = my_palette[1:8])
clusplot(points, cluster8.master, main = 'K-Means clustering (k=8)', color=TRUE, shade=TRUE, labels=2,

my_palette = sample(color, 16)
plot(points, main = 'K-Means clustering (k=16)', col = my_palette[as.factor(cluster16.master)],
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
legend("topright", sprintf("%s",seq(1,16)), fill = my_palette[1:16])
```

```
clusplot(points, cluster16.master, main = 'K-Means clustering (k=16)', color=TRUE, shade=TRUE, labels=2
```



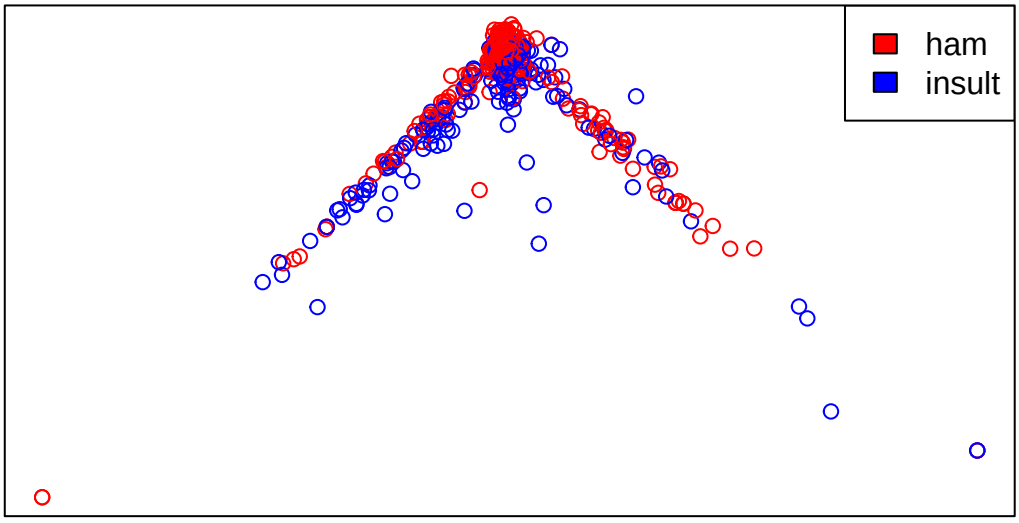







```
par(previous.par) # recovering the original plot space parameters
```

```
points <- cmdscale(dist.matrix, k = 2)
colors <- c('red', 'blue')

plot(points, main = 'Documents with class labels', col = colors[as.factor(train$label)],
     mai = c(0, 0, 0, 0), mar = c(0, 0, 0, 0),
     xaxt = 'n', yaxt = 'n', xlab = '', ylab = '')
legend("topright", c('ham', 'insult'), fill = colors[1:2])
```




```
docs <- stemmed
n <- length(docs)
word_ann <- Maxent_Word_Token_Annotator()
sent_ann <- Maxent_Sent_Token_Annotator()
pos_ann = Maxent_POS_Tag_Annotator()

docsPOS <- list()
for (i in 1:n) {
  doc <-  as.String(docs[[i]])
  wordAnnotation <- annotate(doc, list(sent_ann, word_ann))
  POSAnnotation <- annotate(doc, pos_ann, wordAnnotation)
  POSWords <- subset(POSAnnotation, type == "word")
  POSTags <- vector()
  for (j in 1:length(POSWords$features))
    POSTags <- c(POSTags, POSWords$features[[j]]$POS)
  docsPOS[[i]] <- list(POSTags)
}

head(docsPOS)
```

```
## [[1]]
## [[1]][[1]]
##  [1] "IN"  "NN"  "NN"  "WP"  "NN"  "JJ"  "NN"  "VB"  "RB"  "JJ"  "NN"
## [12] "JJ"  "NN"  "RB"  "VB"  "VB"  "FW"  "NN"  "VB"  "PRP" "VBP" "JJ"
## [23] "NN"  "FW"  "VB"  "NN"  "NNS" "VBP" "RP"  "IN"  "NNP" "NNS" "NN"
## [34] "NN"  "JJ"  "JJ"  "NN"  "NN"  "IN"  "RB"  "VB"  "NN"  "JJ"  "NN"
## [45] "NN"  "JJ"  "NN"  "NN"  "NN"  "NN"  "NN"  "."
##
##
## [[2]]
## [[2]][[1]]
## [1] "NN" "VB"
##
##
## [[3]]
```

```
## [[3]][[1]]
##  [1] "NN"  "NN"  "VBN" "NN"  "VBG" "NN"  "PRP" "MD"  "VB"  "IN"
##
##
## [[4]]
## [[4]][[1]]
## [1] "NN"  "VBP" "JJ"  "JJ"  "NNP" "NN"
##
##
## [[5]]
## [[5]][[1]]
##  [1] "PRP" "VBP" "DT"  "JJ"  "JJ"  "NN"  "VBD" "NN"  "IN"  "CD"  "CD"
## [12] "CD"  "IN"  "CD"  "CD"  "JJ"  "CD"  "JJ"  "NN"  "DT"  "CD"  "CD"
## [23] "NN"  "CD"  "NN"  "JJ"  "NN"  "JJ"  "NN"  "JJ"  "NN"  "NN"  "NN"
## [34] "JJ"  "NN"  "JJ"  "NN"  "IN"  "CD"  "JJ"  "NN"  "IN"  "JJ"  "NN"
## [45] "NN"  "NN"  "NNS"
##
##
## [[6]]
## [[6]][[1]]
## [1] "JJ"  "NN"  "VBP" "CD"
```

```
head(stemmed)
```

```
## [1] "5dcac30f death blow xc2xa0That stuff total danger build toler quick stop abrupt xc2xa0It insidi
## [2] "moron happen"
## [3] "idiot blabbermouth gonna stop e8a1d6c8 day You WILL 91b0cb01 save"
## [4] "tower connect bottom pentagon Sinc flat"
## [5] "I love a9350e16 cee1217a upsid hell ea737b57 calib 9c894fe8 21 1a620f48 17 Intercept 91465074 4
## [6] "d84ee5df shit die 2703f309"
```

```
length(docsPOS)
```

```
## [1] 801
```

3. Modeling

```
test$text_a = as.character(test$text_a)
test$text_a = tm::removePunctuation(test$text_a)
test$text_a = tm::removeWords(x = test$text_a, stopwords(kind = "SMART"))
test$text_a = tm::stripWhitespace(test$text_a)

# n <- length(test$text_a)
# for (i in 1:n) {
#   while(1) {
#     doc <- as.String(test$text_a[[i]])
#     wordAnnotation <- annotate(doc, list(sent_ann, word_ann))
#     POSAnnotation <- annotate(doc, pos_ann, wordAnnotation)
#     POSWords <- subset(POSAnnotation, type == "word")
#     POSTags <- vector()
#     for (j in 1:length(POSWords$features))
#       POSTags <- c(POSTags, POSWords$features[[j]]$POS)
#     tokenPOS <- cbind(doc[POSWords], POSTags)
#     ppn_idx <- which(tokenPOS[,2] == "NNP", 1)
```

```
#     if (length(ppn_idx) == 0) {
#       break;
#     }
#     words <- subset(wordAnnotation, type == "word")
#     hashed <- digest(tokenPOS[ppn_idx, 1], "xxhash32")
#     ppn <- words[ppn_idx]
#     test$text_a[[i]] <- gsub(doc[ppn$start,ppn$end], hashed, doc)
#   }
# }
test$text_a <- iconv(test$text_a, to='UTF-8', sub='byte')
test$label=ifelse(test$label==0,"No","Yes")
test$label <- as.factor(test$label)
stemmedtest <- stemDocument(test$text_a, language = "english")
corpustest <- Corpus(VectorSource(stemmedtest)) # turn into corpus

tdmtest <- tm::DocumentTermMatrix(corpustest)
tdmtest.tfidf <- tm::weightTfIdf(tdmtest)
```

```
## Warning in tm::weightTfIdf(tdmtest): empty document(s): 173 458
```

```
tdmtest.tfidf <- tm::removeSparseTerms(tdmtest.tfidf, 0.999)
tfidftest.matrix <- as.matrix(tdmtest.tfidf)
```

The train dataset is imbalanced with, where there are more than 2 times as much documents labeled as non-insults than insults. We performed undersampling to get rid of this imbalance:

```
train$text_a = stemmed
table(train$label)
```

```
##
##   0   1
## 572 229
```

```
train2 <- ovun.sample(label ~ ., data = train, method = "over")$data
table(train2$label)
```

```
##
##   0   1
## 572 599
```

```
corpus <- Corpus(VectorSource(train2$text_a)) # turn into corpus
tdm <- tm::DocumentTermMatrix(corpus)
tdm.tfidf <- tm::weightTfIdf(tdm)
tdm.tfidf <- tm::removeSparseTerms(tdm.tfidf, 0.999)  # sparsity being not well handled overall in R
tfidf.matrix <- as.matrix(tdm.tfidf)
```

```
train2$lbl <- train2$label
avector <- as.vector(train2['lbl'])
final <- cbind(tfidf.matrix, avector)
final <- as.data.frame(final)
final$lbl=ifelse(final$lbl==0,"No","Yes")
final$lbl <- as.factor(final$lbl) #Adding a vector of labels to the tfidf matrix changing the 0s to No
dat <- twoClassSim(200) #A custom f1 funtion for the metric
f1 <- function(data, lev = NULL, model = NULL) {
  f1_val <- F1_Score(y_pred = data$pred, y_true = data$obs, positive = lev[1])
  c(F1 = f1_val)
}
```

```
train.control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 3,
                              classProbs = TRUE,
                              #sampling = "smote",
                              summaryFunction = f1,
                              search = "grid")
#fitRf <- train(lbl ~ ., data = final, method = 'rf',tuneLength = 5, metric = "F1",
       #      trControl = train.control)
fitSvm <- caret::train(lbl ~ ., data = final, method = 'svmLinear', scale=F, tuneLength = 5, metric = "
print(fitSvm$results$F1)
```

```
## [1] 0.9144553
```

```
#We check what columns are missing in the test dataframe and we add them setting their values to 0
test.matrix <- as.data.frame(tfidftest.matrix)
cols <- colnames(final)
Missing <- setdiff(cols, names(test.matrix))
test.matrix[Missing] <- 0
test.matrix <- test.matrix[cols]

#We predict the two models on the test matrix
#predRf = predict(fitRf, newdata=test.matrix)
predSvm = predict(fitSvm, newdata=test.matrix)

#And then we factorize the labels for visualization
#test$label <- as.factor(test$label)
#con.matrix.rf<-confusionMatrix(predRf, test$label)
#print(con.matrix.rf)
con.matrix.svm<-confusionMatrix(predSvm, test$label)
print(con.matrix.svm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  514 130
##        Yes  45  79
##
##                Accuracy : 0.7721
##                  95% CI : (0.7408, 0.8014)
##     No Information Rate : 0.7279
##     P-Value [Acc > NIR] : 0.002895
##
##                   Kappa : 0.3409
##
##  Mcnemar's Test P-Value : 2.156e-10
##
##             Sensitivity : 0.9195
##             Specificity : 0.3780
##          Pos Pred Value : 0.7981
##          Neg Pred Value : 0.6371
##              Prevalence : 0.7279
##          Detection Rate : 0.6693
```

```
##    Detection Prevalence : 0.8385
##       Balanced Accuracy : 0.6487
##
##        'Positive' Class : No
##
```

4. Understanding

```
corpus <- Corpus(VectorSource(stemmed)) # turn into corpus
tdm <- tm::DocumentTermMatrix(corpus)
tdm.tfidf <- tm::weightTfIdf(tdm)
tdm.tfidf <- tm::removeSparseTerms(tdm.tfidf, 0.999)  # sparsity being not well handled overall in R
tfidf.matrix <- as.matrix(tdm.tfidf)
```

We will use only terms which occur in more than one document and which correlation with other terms is high:

```
# load the library
library(mlbench)
library(caret)

# get rid of words which are only in 1 document
dim(tfidf.matrix)
```

```
## [1]  801 4904
```

```
tfidf.matrix <- tfidf.matrix[,-which(rowSums(as.matrix(tdm2)) == 1)]
dim(tfidf.matrix)
```

```
## [1]  801 1315
```

```
# calculate correlation matrix
correlationMatrix <- cor(tfidf.matrix)
# find attributes that are highly corrected
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)
# how many attributes stays (we get indices as 'highlyCorrelated')
dim(tfidf.matrix)
```

```
## [1]  801 1315
```

```
tfidf.matrix <- tfidf.matrix[,highlyCorrelated]
dim(tfidf.matrix)
```

```
## [1] 801 344
```

Perform feature ranking

We will use the TF-IDF matrix to calculate information gain by filter method:

```
#install.packages('ggpubr')
library(mlr)
```

```
## Loading required package: ParamHelpers
```

```
##
## Attaching package: 'mlr'
```

```
## The following object is masked from 'package:caret':
##
```

```
##     train
```

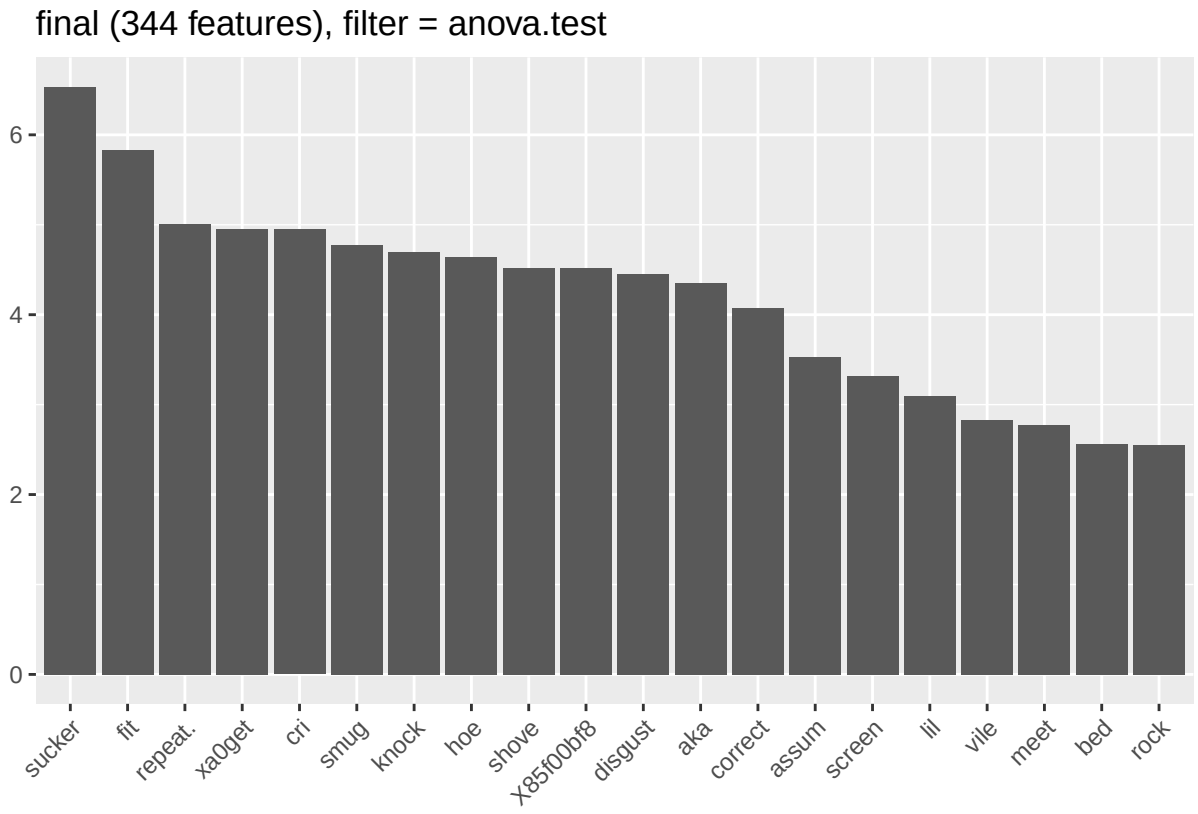
```
library(ggpubr)
```

```
## Loading required package: magrittr
```

```
train$lbl <- train$label
avector <- as.vector(train['lbl'])
final <- cbind(tfidf.matrix, avector)
final <- as.data.frame(final)
final$lbl=ifelse(final$lbl==0,"No","Yes")
final$lbl <- as.factor(final$lbl)
colnames(final) <- make.names(colnames(final),unique = T)
label.task <- makeClassifTask(data=final, target='lbl')

melt <- reshape2::melt
fv = generateFilterValuesData(label.task, method ="anova.test")

fv$data <- fv$data[order(-fv$data$value),]
fv.plot <- fv
fv.plot$data <- head(fv.plot$data, 20)
fv.plot$data
```

```
##          name    type     method    value
## 77     sucker numeric anova.test 6.534402
## 13        fit numeric anova.test 5.827245
## 208   repeat. numeric anova.test 5.006537
## 40     xa0get numeric anova.test 4.949054
## 153       cri numeric anova.test 4.948132
## 25       smug numeric anova.test 4.770762
## 258     knock numeric anova.test 4.698796
## 147       hoe numeric anova.test 4.644575
## 83      shove numeric anova.test 4.520375
## 189 X85f00bf8 numeric anova.test 4.520375
## 12    disgust numeric anova.test 4.453953
## 152       aka numeric anova.test 4.354442
## 207   correct numeric anova.test 4.074235
## 81      assum numeric anova.test 3.530860
## 217    screen numeric anova.test 3.318853
## 45        lil numeric anova.test 3.100535
## 76       vile numeric anova.test 2.828182
## 91       meet numeric anova.test 2.772769
## 309       bed numeric anova.test 2.558271
## 44       rock numeric anova.test 2.547782
```

```
plotFilterValues(fv.plot)
```



Re-evaluate the models performance for top n features

```
features <- fv$data$name
n_features <- c(head(features, 50), 'lbl')
train.data <- final[,n_features]

fitSvm <- caret::train(lbl ~ ., data = train.data, method = 'svmLinear',scale=F, tuneLength = 5, metric
             trControl = train.control)
fitSvm$results$F1
```

```
## [1] 0.8419072
```

```
#We check what columns are missing in the test dataframe and we add them setting their values to 0
test.matrix <- as.data.frame(tfidftest.matrix)
cols <- colnames(final)
Missing <- setdiff(cols, names(test.matrix))
test.matrix[Missing] <- 0
test.matrix <- test.matrix[cols]

#We predict the two models on the test matrix
predSvm = predict(fitSvm, newdata=test.matrix)

#And then we factorize the labels for visualization
con.matrix.svm<-confusionMatrix(predSvm, test$label)
print(con.matrix.svm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  No Yes
##        No  545 191
##        Yes  14  18
##
##                Accuracy : 0.7331
##                  95% CI : (0.7003, 0.7641)
##     No Information Rate : 0.7279
##     P-Value [Acc > NIR] : 0.3905
##
##                   Kappa : 0.0831
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.97496
##             Specificity : 0.08612
##          Pos Pred Value : 0.74049
##          Neg Pred Value : 0.56250
##              Prevalence : 0.72786
##          Detection Rate : 0.70964
##    Detection Prevalence : 0.95833
##       Balanced Accuracy : 0.53054
##
##        'Positive' Class : No
##
```

```
print(con.matrix.svm$overall[1])
```

```
##  Accuracy
## 0.7330729
```

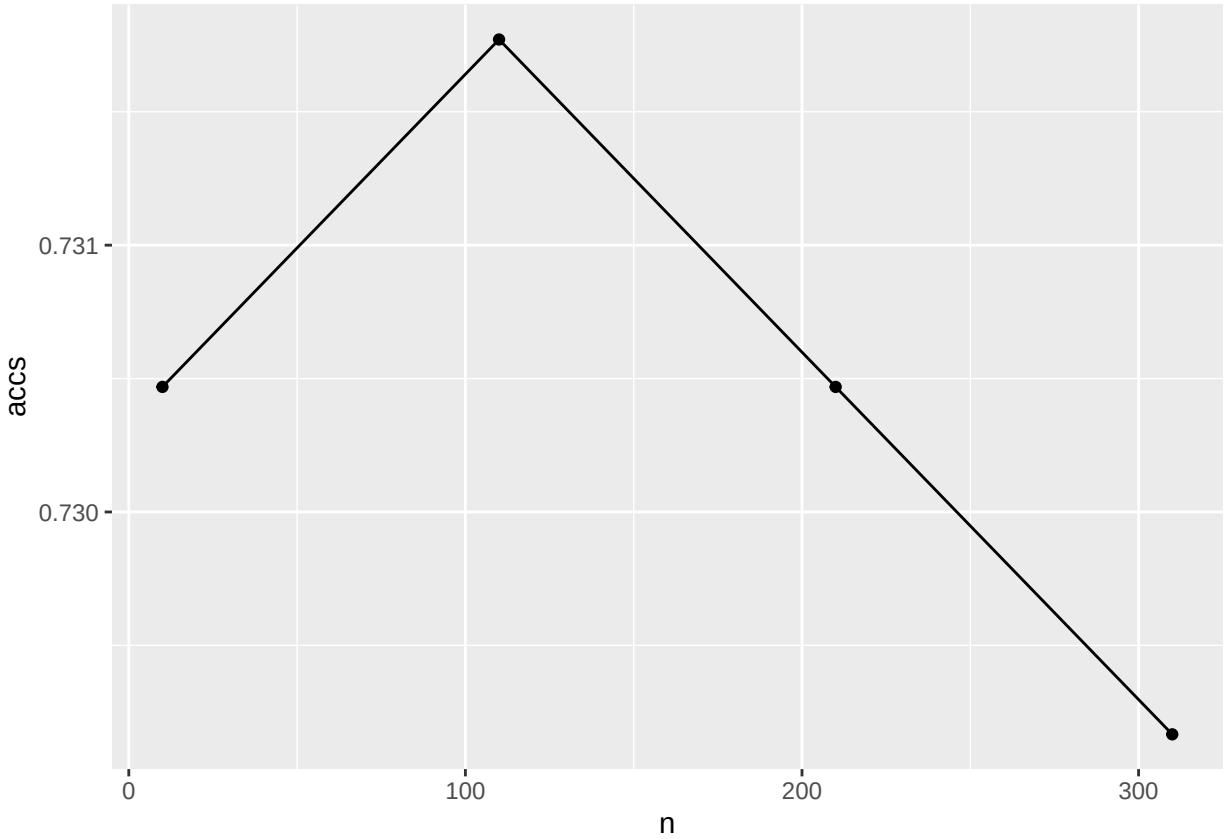
Visualize model performance w.r.t. n by using the selected measure of performance.

```
library(ggplot2)
features <- fv$data$name
#n <- c(seq(10, 100, 10), seq(150, length(features), 100))
n <- seq(10, length(features), 100)
accs <- c()
for (i in 1:length(n)) {
  n_features <- c(head(features, n[i]), 'lbl')
  train.data <- final[,n_features]

  fitSvm <- caret::train(lbl ~ ., data = train.data, method = 'svmLinear',scale=F, tuneLength = 5, metri
              trControl = train.control)

  #We check what columns are missing in the test dataframe and we add them setting their values to 0
  test.matrix <- as.data.frame(tfidftest.matrix)
  cols <- colnames(final)
  Missing <- setdiff(cols, names(test.matrix))
  test.matrix[Missing] <- 0
  test.matrix <- test.matrix[cols]

  #We predict the two models on the test matrix
  predSvm = predict(fitSvm, newdata=test.matrix)
```

```
  #And then we factorize the labels for visualization
  con.matrix.svm<-confusionMatrix(predSvm, test$label)
  print(con.matrix.svm$overall[[1]])
  accs <- c(accs, con.matrix.svm$overall[[1]])
}
```

```
## [1] 0.7304688
## [1] 0.7317708
## [1] 0.7304688
## [1] 0.7291667
```

```
qplot(n, accs, geom=c("point", "line"))
```



Extract feature importances from a wrapper method

```
# training the model
model <- caret::train(lbl ~ ., data = final, method = 'svmLinear', scale=F, tuneLength = 5, metric = "F
importance <- varImp(model, scale=FALSE)$importance
# summarize importance
print(importance)
```

```
##                    No       Yes
## connect       0.5017483 0.5017483
## flat          0.5026224 0.5026224
## sinc          0.5017483 0.5017483
## enough        0.5013131 0.5013131
## flag          0.5013131 0.5013131
## more          0.5013131 0.5013131
## rape          0.5022063 0.5022063
```

```
## remot          0.5013055 0.5013055
## these          0.5013131 0.5013131
## offici         0.5004275 0.5004275
## candi          0.5017483 0.5017483
## disgust        0.5043668 0.5043668
## fit            0.5065502 0.5065502
## control        0.5034965 0.5034965
## starter        0.5017483 0.5017483
## wet            0.5013055 0.5013055
## class          0.5004428 0.5004428
## wage           0.5013055 0.5013055
## bulli          0.5004352 0.5004352
## winner         0.5017483 0.5017483
## heat           0.5017483 0.5017483
## cool           0.5004275 0.5004275
## shes           0.5026224 0.5026224
## russian        0.5017483 0.5017483
## smug           0.5043668 0.5043668
## answer         0.5026224 0.5026224
## prior          0.5017483 0.5017483
## field          0.5004275 0.5004275
## theater        0.5013055 0.5013055
## catch          0.5004390 0.5004390
## gun            0.5034965 0.5034965
## avoid          0.5013055 0.5013055
## brink          0.5017483 0.5017483
## urg            0.5017483 0.5017483
## int            0.5017483 0.5017483
## kommer         0.5017483 0.5017483
## och            0.5017483 0.5017483
## pxe5           0.5017483 0.5017483
## xe4r           0.5026224 0.5026224
## xa0get         0.5043668 0.5043668
## system         0.5034965 0.5034965
## f9a82e4ds      0.5021834 0.5021834
## network        0.5021834 0.5021834
## rock           0.5034927 0.5034927
## lil            0.5043668 0.5043668
## crimin         0.5004504 0.5004504
## encourag       0.5017483 0.5017483
## late           0.5017483 0.5017483
## lectur         0.5017483 0.5017483
## cant           0.5034927 0.5034927
## holder         0.5017483 0.5017483
## scum           0.5034851 0.5034851
## cathol         0.5017483 0.5017483
## social         0.5043706 0.5043706
## len            0.5017483 0.5017483
## medic          0.5017483 0.5017483
## with           0.5013207 0.5013207
## week           0.5004466 0.5004466
## jag            0.5017483 0.5017483
## again          0.5004352 0.5004352
## prison         0.5017483 0.5017483
```

```
## situat         0.5004275 0.5004275
## abort          0.5017483 0.5017483
## ban            0.5017483 0.5017483
## slide          0.5017483 0.5017483
## compani        0.5017483 0.5017483
## parent         0.5013131 0.5013131
## shoot          0.5017483 0.5017483
## hide           0.5004428 0.5004428
## order          0.5013131 0.5013131
## sign           0.5034965 0.5034965
## number         0.5034965 0.5034965
## believ         0.5017483 0.5017483
## replac         0.5017483 0.5017483
## mine           0.5013055 0.5013055
## vile           0.5034927 0.5034927
## sucker         0.5065502 0.5065502
## feder          0.5017483 0.5017483
## even           0.5034965 0.5034965
## result         0.5017483 0.5017483
## assum          0.5035003 0.5035003
## suit           0.5013131 0.5013131
## shove          0.5043668 0.5043668
## sure           0.5017483 0.5017483
## foolish        0.5013131 0.5013131
## bin            0.5017483 0.5017483
## childish       0.5013055 0.5013055
## pre            0.5017483 0.5017483
## she            0.5026186 0.5026186
## green          0.5004275 0.5004275
## meet           0.5034927 0.5034927
## walk           0.5043706 0.5043706
## fastbal        0.5017483 0.5017483
## pig            0.5017483 0.5017483
## turd           0.5034851 0.5034851
## intent         0.5026224 0.5026224
## close          0.5026224 0.5026224
## cream          0.5013131 0.5013131
## swallow        0.5013131 0.5013131
## virgin         0.5013131 0.5013131
## red            0.5004275 0.5004275
## stamp          0.5017483 0.5017483
## rude           0.5013055 0.5013055
## fall           0.5017483 0.5017483
## count          0.5017483 0.5017483
## ask            0.5013055 0.5013055
## taxpay         0.5017483 0.5017483
## nasti          0.5013131 0.5013131
## moder          0.5017483 0.5017483
## western        0.5013055 0.5013055
## recal          0.5017483 0.5017483
## despit         0.5017483 0.5017483
## tail           0.5017483 0.5017483
## fund           0.5017483 0.5017483
## rider          0.5013131 0.5013131
```

```
## claim          0.5017483 0.5017483
## sweeti         0.5017483 0.5017483
## eventu         0.5017483 0.5017483
## oversea        0.5017483 0.5017483
## debt           0.5017483 0.5017483
## energi         0.5017483 0.5017483
## X5dcac30f      0.5008741 0.5008741
## dose           0.5008741 0.5008741
## X91465074      0.5008741 0.5008741
## cb740efa       0.5008741 0.5008741
## influenc       0.5026224 0.5026224
## X0308ccbd      0.5008741 0.5008741
## X30e1f3e0      0.5008741 0.5008741
## X7633fb30      0.5008741 0.5008741
## X90d7d4e1      0.5008741 0.5008741
## X977a1422      0.5008741 0.5008741
## X9be256e2      0.5008741 0.5008741
## mention        0.5043706 0.5043706
## X17a1172a      0.5021834 0.5021834
## b4f35d1eing    0.5021834 0.5021834
## mental         0.5017215 0.5017215
## X3ce93de9      0.5008741 0.5008741
## X8b89ec53      0.5008741 0.5008741
## sum            0.5013055 0.5013055
## site           0.5021834 0.5021834
## color          0.5026186 0.5026186
## attach         0.5013131 0.5013131
## victim         0.5026224 0.5026224
## X2189a5ad      0.5008741 0.5008741
## ppl            0.5013055 0.5013055
## breath         0.5034851 0.5034851
## hoe            0.5043668 0.5043668
## book           0.5004352 0.5004352
## letto          0.5017483 0.5017483
## classforumitem 0.5017483 0.5017483
## div            0.5017483 0.5017483
## aka            0.5043668 0.5043668
## cri            0.5043668 0.5043668
## slaveri        0.5021834 0.5021834
## mom            0.5026186 0.5026186
## wors           0.5004428 0.5004428
## burn           0.5004275 0.5004275
## hole           0.5034851 0.5034851
## effect         0.5034965 0.5034965
## drive          0.5017483 0.5017483
## deep           0.5013055 0.5013055
## alla           0.5008741 0.5008741
## fix            0.5013131 0.5013131
## X62de19d5      0.5021834 0.5021834
## X9278acac      0.5056723 0.5056723
## deepli         0.5017483 0.5017483
## X040da673      0.5008741 0.5008741
## X28408123      0.5008741 0.5008741
## dream          0.5013131 0.5013131
```

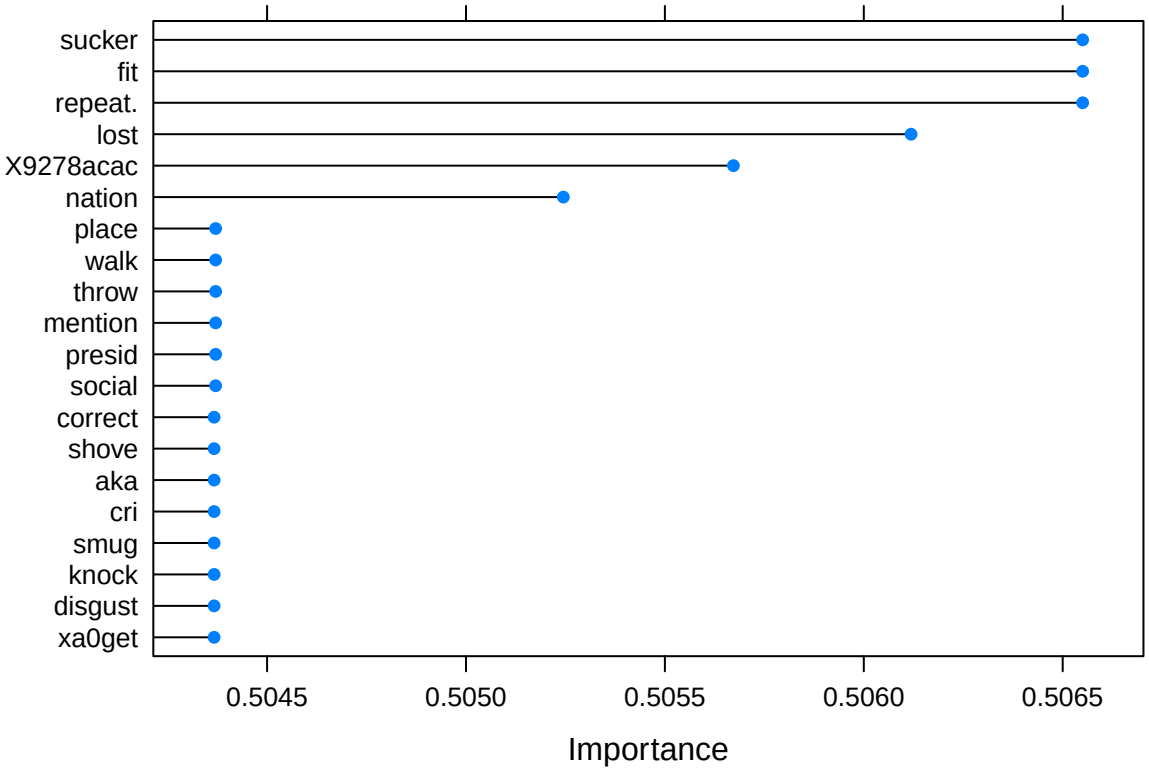
```
## hmm           0.5017483 0.5017483
## then          0.5021910 0.5021910
## bigot         0.5017483 0.5017483
## X075aca06     0.5008741 0.5008741
## professor     0.5008741 0.5008741
## worthless     0.5034851 0.5034851
## X2c2d7d7b     0.5008741 0.5008741
## xc2xa0h       0.5008741 0.5008741
## japan         0.5008741 0.5008741
## xa0that       0.5017483 0.5017483
## hold          0.5004428 0.5004428
## write         0.5004275 0.5004275
## earth         0.5013055 0.5013055
## xa0i          0.5013055 0.5013055
## warm          0.5013055 0.5013055
## repub         0.5017483 0.5017483
## food          0.5021758 0.5021758
## stuff         0.5004428 0.5004428
## anyway        0.5013055 0.5013055
## X85f00bf8     0.5043668 0.5043668
## X21578f2c     0.5021834 0.5021834
## a777b4a8      0.5021834 0.5021834
## photograph    0.5017483 0.5017483
## lot           0.5026415 0.5026415
## busi          0.5008703 0.5008703
## sensit        0.5013055 0.5013055
## respect       0.5004428 0.5004428
## bollen        0.5008741 0.5008741
## dxe4r         0.5008741 0.5008741
## dxe5          0.5008741 0.5008741
## fxe5r         0.5008741 0.5008741
## jxe4vla       0.5008741 0.5008741
## mxe5l         0.5008741 0.5008741
## nation        0.5052448 0.5052448
## enforc        0.5008741 0.5008741
## enjoy         0.5017483 0.5017483
## children      0.5034354 0.5034354
## correct       0.5043668 0.5043668
## repeat.       0.5065502 0.5065502
## basic         0.5004428 0.5004428
## X0301c4b8     0.5008741 0.5008741
## X1de6b054     0.5008741 0.5008741
## X249e492a     0.5008741 0.5008741
## gas           0.5026224 0.5026224
## crime         0.5017483 0.5017483
## grow          0.5013207 0.5013207
## ac797eb       0.5021834 0.5021834
## screen        0.5043668 0.5043668
## toler         0.5017483 0.5017483
## divis         0.5017483 0.5017483
## attack        0.5021987 0.5021987
## super         0.5017483 0.5017483
## borrow        0.5008741 0.5008741
## db494fdc      0.5008741 0.5008741
```

```
## downgrad       0.5008741 0.5008741
## tape           0.5013055 0.5013055
## presid         0.5043706 0.5043706
## band           0.5008741 0.5008741
## cover          0.5008741 0.5008741
## babi           0.5004352 0.5004352
## feet           0.5026224 0.5026224
## throw          0.5043706 0.5043706
## destroy        0.5034965 0.5034965
## anyon          0.5004352 0.5004352
## tripl          0.5017483 0.5017483
## X0a2bedd4      0.5021834 0.5021834
## X2894ec62      0.5021834 0.5021834
## X99e1ce42      0.5021834 0.5021834
## wont           0.5017444 0.5017444
## rest           0.5034812 0.5034812
## coat           0.5008741 0.5008741
## lord           0.5008741 0.5008741
## ou2019er       0.5008741 0.5008741
## rain           0.5008741 0.5008741
## cultur         0.5013131 0.5013131
## push           0.5017483 0.5017483
## lost           0.5061189 0.5061189
## repli          0.5034965 0.5034965
## but            0.5008627 0.5008627
## d7c6320e       0.5008741 0.5008741
## pure           0.5017483 0.5017483
## dealnit        0.5008741 0.5008741
## who            0.5034851 0.5034851
## societi        0.5004352 0.5004352
## movi           0.5004275 0.5004275
## rememb         0.5030843 0.5030843
## issu           0.5039546 0.5039546
## crazi          0.5034965 0.5034965
## knock          0.5043668 0.5043668
## won            0.5026224 0.5026224
## X00af952       0.5008741 0.5008741
## X86ab4d30      0.5008741 0.5008741
## X88f422fd      0.5008741 0.5008741
## X0b63caf5      0.5008741 0.5008741
## pack           0.5026224 0.5026224
## money          0.5039469 0.5039469
## honor          0.5017483 0.5017483
## punch          0.5004275 0.5004275
## haha           0.5017483 0.5017483
## those          0.5034965 0.5034965
## puke           0.5013055 0.5013055
## becaus         0.5017483 0.5017483
## ant            0.5008741 0.5008741
## men            0.5004504 0.5004504
## amaz           0.5004352 0.5004352
## contribut      0.5017483 0.5017483
## brought        0.5013131 0.5013131
## X3081a0c6      0.5008741 0.5008741
```

```
## del           0.5008741 0.5008741
## ea9baec       0.5008741 0.5008741
## fe8206a2      0.5008741 0.5008741
## justifi       0.5017483 0.5017483
## god           0.5004352 0.5004352
## edh           0.5008741 0.5008741
## kan           0.5008741 0.5008741
## trail         0.5017483 0.5017483
## b27311f3      0.5008741 0.5008741
## X314a5d0b     0.5021834 0.5021834
## suppos        0.5013131 0.5013131
## X046967a8     0.5021834 0.5021834
## cd9293c0      0.5021834 0.5021834
## power         0.5026224 0.5026224
## X7988eaa5     0.5008741 0.5008741
## materi        0.5013131 0.5013131
## bunch         0.5013055 0.5013055
## X05714d31     0.5008741 0.5008741
## ff4b6c8b      0.5008741 0.5008741
## poor          0.5000115 0.5000115
## arrog         0.5004428 0.5004428
## X3f580b72     0.5008741 0.5008741
## X465dd2ec     0.5008741 0.5008741
## X71ff7d52     0.5008741 0.5008741
## place         0.5043706 0.5043706
## degener       0.5021834 0.5021834
## dyke          0.5021834 0.5021834
## feminist      0.5021834 0.5021834
## reflect       0.5013055 0.5013055
## land          0.5013055 0.5013055
## mate          0.5017483 0.5017483
## bed           0.5034927 0.5034927
## domest        0.5008741 0.5008741
## referendum    0.5008741 0.5008741
## hous          0.5004275 0.5004275
## compar        0.5004275 0.5004275
## longer        0.5013131 0.5013131
## allergi       0.5021834 0.5021834
## dye           0.5021834 0.5021834
## pistachio     0.5021834 0.5021834
## determin      0.5017483 0.5017483
## sex           0.5017483 0.5017483
## left          0.5021261 0.5021261
## wit           0.5017483 0.5017483
## som           0.5017483 0.5017483
## X554356d0     0.5008741 0.5008741
## X2951f27f     0.5008741 0.5008741
## xc2xa0i       0.5017483 0.5017483
## daili         0.5017483 0.5017483
## X14071d47     0.5008741 0.5008741
## X5594625c     0.5008741 0.5008741
## X6e602b90     0.5008741 0.5008741
## X0a70a953     0.5008741 0.5008741
## X0f64c5fd     0.5008741 0.5008741
```

```
## X8ff416dd      0.5008741 0.5008741
## interests      0.5008741 0.5008741
## profit         0.5017483 0.5017483
## fine           0.5017483 0.5017483
## X6502c311      0.5008741 0.5008741
## X9c5cfce7      0.5008741 0.5008741
## giant          0.5013131 0.5013131
## horribl        0.5017483 0.5017483
## exchang        0.5017483 0.5017483
## foxnew         0.5008741 0.5008741
## newt           0.5008741 0.5008741
## rick           0.5008741 0.5008741
## upsid          0.5013131 0.5013131
```

```
# plot importance
plot(varImp(model, scale=FALSE), top = 20)
```



Compare the two feature rankings

```
library(ggplot2)
get_jaccard <- function(A, B) {
  return(length(intersect(A,B)) / length(union(A,B)))
}

A <- fv$data$name
B <- rownames(importance[order(-importance$Yes),])
length(A) == length(B)
```

```
## [1] TRUE
```

```
Jaccard.score <- c()
for (i in 1:length(A)) {
  Jaccard.score[i] <- get_jaccard(A[1:i], B[1:i])
}
n <- seq(1, length(A), 1)
qplot(n, Jaccard.score)
```