Keosotra Veng
Stat 6620 Sect 2

Homework 3

Perform the SMS spam filtering analysis.

Step 1: Exploring and preparing the data ----

- We got the data from UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/

Step 2: Exploring and preparing the data ----

- We import sms_spam.csv data set into R

```
sms_raw <- read.csv("sms_spam.csv", stringsAsFactors = FALSE)
```

- Now we convert the sms type into factor Ham = 1 and Spam = 2

```
sms_raw$type <- factor(sms_raw$type)
```

- We have R print the structure of column "type" in sms_raw. We also use table function to count how many are classified as spam and how many are classified as ham.

```
str(sms_raw$type)

##  Factor w/ 2 levels "ham","spam": 1 1 1 2 2 1 1 1 2 1 ...

table(sms_raw$type)

##
##  ham spam
## 4812  747
```

- We use vectorsource and Vcorpus from tm to break the string text into single string text and put into in a corpus.

```
library(tm)

## Loading required package: NLP

sms_corpus <- VCorpus(VectorSource(sms_raw$text))

print(sms_corpus)

## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5559

inspect(sms_corpus[1:2])

## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 2
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 49
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 23
```

```
as.character(sms_corpus[[1]])
```

```
## [1] "Hope you are having a good week. Just checking in"
```

```
lapply(sms_corpus[1:2], as.character)
```

```
## $`1`
## [1] "Hope you are having a good week. Just checking in"
##
## $`2`
## [1] "K..give back my thanks."
```

- We use tm_map to convert all the text from the corpus to lowercase and put it in sms_corpus_clean

```
sms_corpus_clean <- tm_map(sms_corpus, content_transformer(tolower))
```

- We then remove numbers, stop words (meaningless words), and punctuation from the sms_corpus_clean variable.

```
sms_corpus_clean <- tm_map(sms_corpus_clean, removeNumbers)
sms_corpus_clean <- tm_map(sms_corpus_clean, removeWords, stopwords())
sms_corpus_clean <- tm_map(sms_corpus_clean, removePunctuation)
```

- We use stemDocument function from SnowballC to compute the stems of each words in the corpus and then we get rid of the unneeded whitespace in the corpus like double spaces.

```
library(SnowballC)

sms_corpus_clean <- tm_map(sms_corpus_clean, stemDocument)

sms_corpus_clean <- tm_map(sms_corpus_clean, stripWhitespace)
```

- We create a document term matrix from sms_corpus_clean. Basically we have each unique words and put it into columns and rows would be to see how many time these words occur.

Keosotra Veng
Stat 6620 Sect 2

```
sms_dtm <- DocumentTermMatrix(sms_corpus_clean)
```

- We set 4169 observations to train data set and the rest to test data set.

```
sms_dtm_train <- sms_dtm[1:4169, ]
sms_dtm_test  <- sms_dtm[4170:5559, ]
```

- We are creating labels for traning and test data set

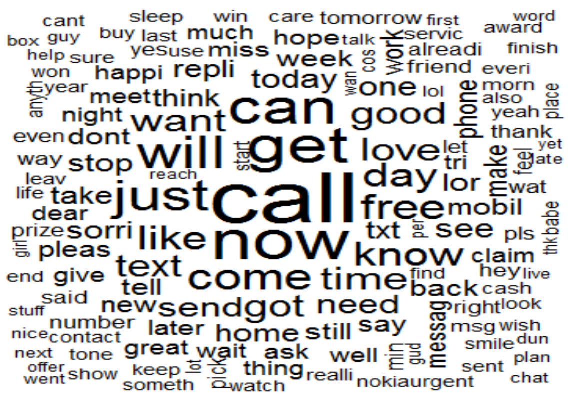
```
sms_train_labels <- sms_raw[1:4169, ]$type
sms_test_labels  <- sms_raw[4170:5559, ]$type
```

- We creat a probability table for Ham and Spam. in the train data set we have 86% ham and 14% spam and in our test data set we have 87% ham and 13% spam

```
prop.table(table(sms_train_labels))

## sms_train_labels
##       ham      spam
## 0.8647158 0.1352842

prop.table(table(sms_test_labels))

## sms_test_labels
##       ham      spam
## 0.8683453 0.1316547
```

- We create the word cloud using wordcloud library to see which word occur the most.

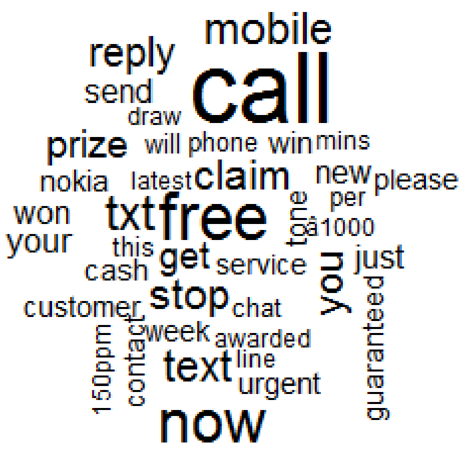
```
library(wordcloud)

## Loading required package: RColorBrewer

wordcloud(sms_corpus_clean, min.freq = 50, random.order = FALSE)
```



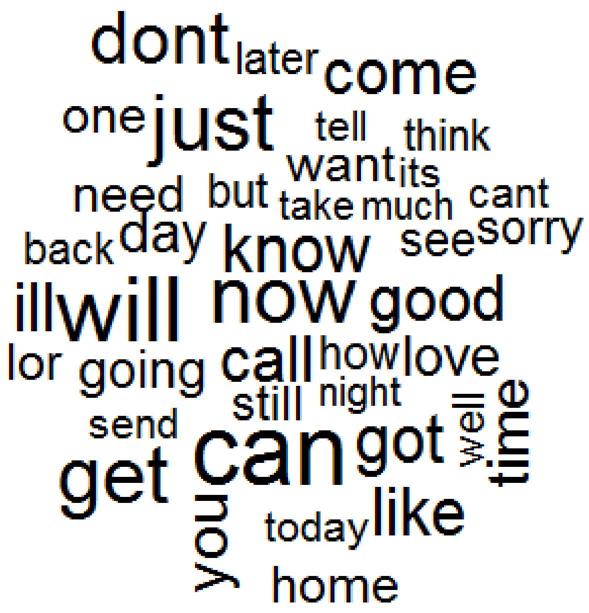
- Now we seperate spam and ham and then look at the most occurred words.

Keosotra Veng
Stat 6620 Sect 2

```r
spam <- subset(sms_raw, type == "spam")
ham  <- subset(sms_raw, type == "ham")

wordcloud(spam$text, max.words = 40, scale = c(3, 0.5))
```



```r
wordcloud(ham$text, max.words = 40, scale = c(3, 0.5))
```



- We set sparsity to 0.999 meaning that if the frequency of the words are more than 0.999 then we remove them.

```r
sms_dtm_freq_train <- removeSparseTerms(sms_dtm_train, 0.999)
sms_dtm_freq_train
```

```
## <<DocumentTermMatrix (documents: 4169, terms: 1104)>>
## Non-/sparse entries: 24827/4577749
## Sparsity           : 99%
## Maximal term length: 19
## Weighting          : term frequency (tf)
```

- We only keep the words that have a frequency of 5 or more.

```
sms_freq_words <- findFreqTerms(sms_dtm_train, 5)
str(sms_freq_words)

##  chr [1:1139] "â<U+0082>â<U+0080><U+009C>""| __truncated__ "abiola" "abl"
"abt" "accept" ...
```

- We create DTMS with only the frequent term

```
sms_dtm_freq_train <- sms_dtm_train[ , sms_freq_words]
sms_dtm_freq_test <- sms_dtm_test[ , sms_freq_words]
```

- Convert count into factor if it's 0 then it's "No", otherwise it's "Yes"

```
convert_counts <- function(x) {
  x <- ifelse(x > 0, "Yes", "No")
}
```

- We apply the convert_counts function to sms_dtm_freq_train data set

```
sms_train <- apply(sms_dtm_freq_train, MARGIN = 2, convert_counts)
sms_test  <- apply(sms_dtm_freq_test, MARGIN = 2, convert_counts)
```

Step 3: Training a model on the data ----

- We applied naiveBayes algorithm to our train data set

```
library(e1071)
sms_classifier <- naiveBayes(sms_train, sms_train_labels)
```

Step 4: Evaluating model performance ----

- We predict there are 1207 actual hams and we predicted 1201 correctly and only 6 wrong. There are 183 spams and we predicted 153 correctly and 30 wrong that we predicted ham, but it was spam.

```
sms_test_pred <- predict(sms_classifier, sms_test)

library(gmodels)
CrossTable(sms_test_pred, sms_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  1390
##
##
##              | actual
##    predicted |       ham |      spam | Row Total |
## -------------|-----------|-----------|-----------|
##          ham |      1201 |        30 |      1231 |
##              |     0.995 |     0.164 |           |
## -------------|-----------|-----------|-----------|
##         spam |         6 |       153 |       159 |
##              |     0.005 |     0.836 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      1207 |       183 |      1390 |
##              |     0.868 |     0.132 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Step 5: Improving model performance ----

- We can improve our model with laplace whic is a tuning parameter in naiveBayes algorithm. Now we set laplace to 1.

The result is better. Now we predicted 1202 Ham out of 1207 and only 5 wrong. And we predicted 155 spams out of 183 spams and only 28 wrong.

```
sms_classifier2 <- naiveBayes(sms_train, sms_train_labels, laplace = 1)
sms_test_pred2 <- predict(sms_classifier2, sms_test)
CrossTable(sms_test_pred2, sms_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c('predicted', 'actual'))

##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  1390
##
##
##              | actual
##    predicted |       ham |      spam | Row Total |
## -------------|-----------|-----------|-----------|
##          ham |      1202 |        28 |      1230 |
##              |     0.996 |     0.153 |           |
## -------------|-----------|-----------|-----------|
##         spam |         5 |       155 |       160 |
##              |     0.004 |     0.847 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      1207 |       183 |      1390 |
##              |     0.868 |     0.132 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Perform HouseVote84 Analysis

Step 1: Exploring and preparing the data ----

- We got the data from UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/

Step 2: Exploring and preparing the data ----

- We load HouseVotes84 data set into R

```
load("~/R/HouseVotes84.rda")
```

- We have R print the structure of type column. We also use table function to count how many are classified as republicans and how many are classified as democrats. We have 267 democrats and 168 republicans

```
str(HouseVotes84$Class)

##  Factor w/ 2 levels "democrat","republican": 2 2 1 1 1 1 1 2 2 1 ...

table(HouseVotes84$Class)

##
##   democrat republican
##        267        168
```

Step 3: Training a model on the data ----

- We applied naiveBayes algorithm to our data set

```
library(e1071)
model <- naiveBayes(Class ~ ., data = HouseVotes84)
```

```
predict(model, HouseVotes84[1:10, -1])

##  [1] republican republican republican democrat   democrat   democrat
##  [7] republican republican republican democrat
## Levels: democrat republican

predict(model, HouseVotes84[1:10,-1], type = "raw")

##           democrat   republican
##  [1,] 1.029209e-07 9.999999e-01
##  [2,] 5.820415e-08 9.999999e-01
##  [3,] 5.684937e-03 9.943151e-01
##  [4,] 9.985798e-01 1.420152e-03
##  [5,] 9.666720e-01 3.332802e-02
##  [6,] 8.121430e-01 1.878570e-01
##  [7,] 1.751512e-04 9.998248e-01
##  [8,] 8.300100e-06 9.999917e-01
##  [9,] 8.277705e-08 9.999999e-01
## [10,] 1.000000e+00 5.029425e-11
```

Step 4: Evaluating model performance ----

- We predicted 238 democrats out of 267 democrats and 29 wrongly predicted. And we predicted 155 republicans out of 168 republicans and 13 wrongly predicted.

```
pred <- predict(model, HouseVotes84[,-1])
table(pred, HouseVotes84$Class)

## 
## pred         democrat republican
##   democrat        238         13
##   republican       29        155
```

Step 5: Improving model performance ----

- We can improve our model with laplace which is a tuning parameter in naiveBayes algorithm. Now we set laplace to 1.

The result is the same. Now we predicted 238 democrats out of 267 democrats and 29 wrongly predicted. And we predicted 155 republicans out of 168 republicans and 13 wrongly predicted.

```
model1 <- naiveBayes(Class ~ ., data = HouseVotes84, laplace = 1)
pred1 <- predict(model1, HouseVotes84[,-1])
table(pred1, HouseVotes84$Class)

## 
## pred1        democrat republican
##   democrat        238         13
##   republican       29        155
```

Keosotra Veng
Stat 6620 Sect 2

Google Sheet Add-on app (Gavagai Sentiment Analysis)

- I found this add-on on google sheet. It's a text sentiment anlaysis where you enter texts in a cell, then it would have another column that indicate whether the content is positive, negative, skepticism, hate, violence, love, fear, and desire. Those are their only catagories. I tried to look at their website but they didn't disclose their algorithm in doing this. But it seems like the algorithnm would be something similiar to what we learned like knn or naive bayes. It basically looking at certain word then score it based on the occurance of those words and then predict whether which of the catagories above they belong to. I think it's similiar to spam or ham sms but instead of two, they have 8 factors.