# PeerReview_4

Minxin Cheng

0. Load packages

```
#install.packages("tm")
#install.packages("SnowballC")
#install.packages("wordcloud")
#install.packages("e1071")
#install.packages("klaR")

library(tm)
```

```
## Loading required package: NLP
```

```
library(SnowballC)
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(e1071)
library(gmodels)
library(klaR)
```

```
## Loading required package: MASS
```

# Problem 1

1. Read in data file

```
sms_raw <- read.csv("da5030.spammsgdataset.csv", stringsAsFactors = FALSE)
```

2. Get the overall information of the data

```
# check the structure of the data
str(sms_raw)
```

```
## 'data.frame':    5574 obs. of  2 variables:
##  $ type: chr  "ham" "ham" "spam" "ham" ...
##  $ text: chr  "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... C
```

```r
# convert type column to factor
sms_raw$type <- factor(sms_raw$type)
# check the type column and count the number in each type
str(sms_raw$type)
```

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

```r
table(sms_raw$type)
```

```
##
##  ham spam
## 4827  747
```

3. Create corpus

```r
# create sms corpus
sms_corpus <- VCorpus(VectorSource(sms_raw$text))
# print the corpus to check
print(sms_corpus)
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5574
```

4. Clean the dataset

```r
# convert text string to all lower case
sms_corpus_clean <- tm_map(sms_corpus, content_transformer(tolower))
# remove numbers from text
sms_corpus_clean <- tm_map(sms_corpus_clean, removeNumbers)
# remove stop words such as "to", "and", and "or"
sms_corpus_clean <- tm_map(sms_corpus_clean, removeWords, stopwords())
# remove punctuation
sms_corpus_clean <- tm_map(sms_corpus_clean, removePunctuation)
# only keep the stems of the word
sms_corpus_clean <- tm_map(sms_corpus_clean, stemDocument)
# remove the blank spaces due to the cleaning
sms_corpus_clean <- tm_map(sms_corpus_clean, stripWhitespace)
```

5. Splitting text documents into words, create a sparse matrix

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

6. Create training data and testing data

```r
# split the data by row number
sms_dtm_train <- sms_dtm[1:4169, ]
sms_dtm_test <- sms_dtm[4170:5559, ]
# get the type information from the original dataset
sms_train_labels <- sms_raw[1:4169, ]$type
sms_test_labels <- sms_raw[4170:5559, ]$type
# check the portion of ham and span in training and testing data
prop.table(table(sms_train_labels))
```

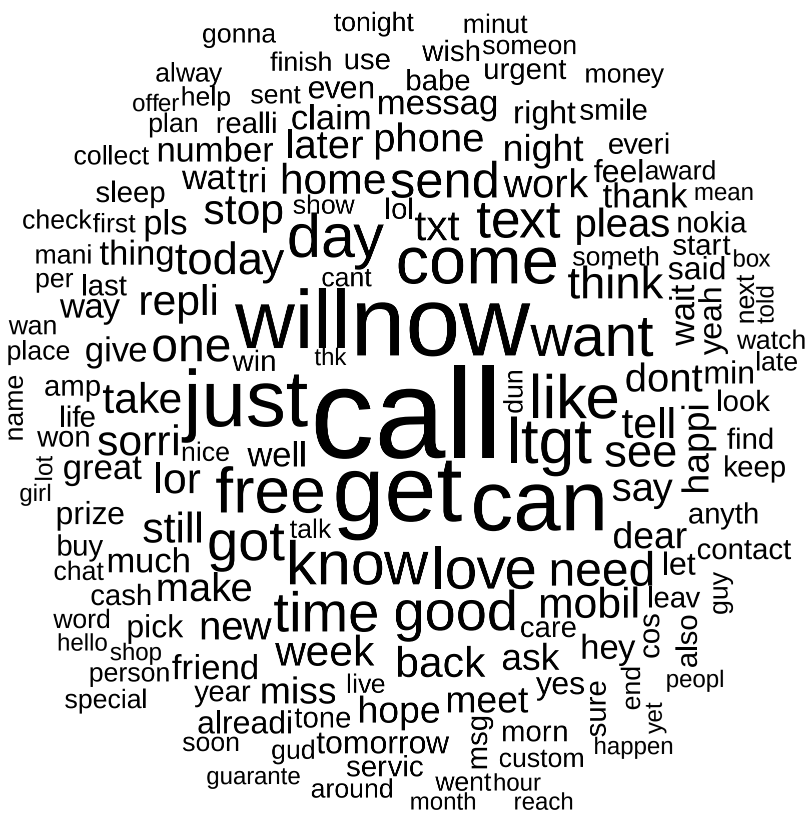
```
## sms_train_labels
##       ham      spam
## 0.8647158 0.1352842
```

```r
prop.table(table(sms_test_labels))
```

```
## sms_test_labels
##       ham      spam
## 0.8697842 0.1302158
```

7. Create word cloud 7.1 Create a word cloud for the entire dataset

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




7.2 Create word cloud for spam data and ham data respectively

```r
# separate span and ham data
spam <- subset(sms_raw, type == "spam")
ham <- subset(sms_raw, type == "ham")
# create word cloud for each
wordcloud(spam$text, max.words = 40, scale = c(3, 0.5))
```

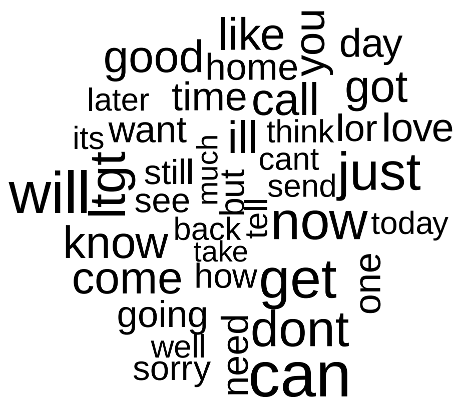
```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```


```r
wordcloud(ham$text, max.words = 40, scale = c(2, 0.5))
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```




8. Get the most frequent words

```r
# only keep the words that appeared at least 5 times
sms_freq_words <- findFreqTerms(sms_dtm_train, 5)
str(sms_freq_words)
```

```
##  chr [1:1157] "£wk" "abiola" "abl" "abt" "accept" "access" "account" ...
```

```r
# only keep the columns that have the frequent words
sms_dtm_freq_train <- sms_dtm_train[ , sms_freq_words]
sms_dtm_freq_test <- sms_dtm_test[ , sms_freq_words]
```

9. Convert counts to yes/no for Naive Bayes

```r
# create the function
convert_counts <- function(x){
  x <- ifelse(x > 0, "Yes", "No")
}
# apply the function to both training and testing data
sms_train <- apply(sms_dtm_freq_train, MARGIN = 2, convert_counts)
sms_test <- apply(sms_dtm_freq_test, MARGIN = 2, convert_counts)
```

10. Apply the Naive Bayes

```r
sms_classifier <- naiveBayes(sms_train, sms_train_labels)
sms_test_pred <- predict(sms_classifier, sms_test)
```

11. Evaluate the prediction

```r
CrossTable(sms_test_pred, sms_test_labels, prop.chisq = FALSE, prop.t = FALSE, dnn = c("predicted", "ac
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  1390
##
##
##              | actual
##    predicted |       ham |      spam | Row Total |
## -------------|-----------|-----------|-----------|
##          ham |      1200 |        20 |      1220 |
##              |     0.984 |     0.016 |     0.878 |
##              |     0.993 |     0.110 |           |
## -------------|-----------|-----------|-----------|
##         spam |         9 |       161 |       170 |
##              |     0.053 |     0.947 |     0.122 |
##              |     0.007 |     0.890 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      1209 |       181 |      1390 |
##              |     0.870 |     0.130 |           |
## -------------|-----------|-----------|-----------|
##
##
```

As the table above, 1390 predictions were made, the accuracy is (1200 + 161) / 1390 = 97.9%

12. Improve the model

```r
# set laplace as 1
sms_classifier2 <- naiveBayes(sms_train, sms_train_labels, laplace = 1)
sms_test_pred2 <- predict(sms_classifier2, sms_test)
# evaluate the prediction
CrossTable(sms_test_pred2, sms_test_labels, prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE, dnn = c
```

```
##
##
##    Cell Contents
## |-------------------------|
```

```
## |                        N |
## |            N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  1390
##
##
##              | actual
##    predicted |       ham |      spam | Row Total |
## -------------|-----------|-----------|-----------|
##          ham |      1182 |        10 |      1192 |
##              |     0.978 |     0.055 |           |
## -------------|-----------|-----------|-----------|
##         spam |        27 |       171 |       198 |
##              |     0.022 |     0.945 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      1209 |       181 |      1390 |
##              |     0.870 |     0.130 |           |
## -------------|-----------|-----------|-----------|
##
##
```

As the table above, the accuracy is (1182 + 171) / 1390

# Problem 2

1. Load and check dataset

```r
# load data
data(iris)
# get an overview of the data
nrow(iris)
```

```
## [1] 150
```

```r
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800  
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500  
##        Species  
##  setosa    :50  
##  versicolor:50  
##  virginica :50  
##
##
##
```

```r
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

2. split the dataset

```r
# get all the row number that can be devided by 5
testidx <- which(1:length(iris[, 1]) %% 5 == 0)
# split the data
iristrain <- iris[-testidx, ]
iristest <- iris[testidx, ]
```

3. Apply Naive Bayes and make the prediction

```r
nbmodel <- NaiveBayes(Species~., data = iristrain)
prediction <- predict(nbmodel, iristest[ , -5])
```

4. Evaluate the prediction

```r
table(prediction$class, iristest[ , 5])
```

```
##             
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         2
##   virginica       0          0         8
```

As the table above, the predict accuracy is (10 + 10 + 8) / (10 + 10 + 2 + 8) = 0.93