**FILTERING OF MESSAGES USING NAÏVE BAYES**

This project uses Naïve Bayes to create a classification algorithm that filters spam (unwanted) messages from ham messages. The dataset was obtained from <http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/>.

All the code for the project can be found at my GitHub page <https://github.com/chrisliti/Naive-Bayes-Text-Classifier>. R software was used to clean and process the data then training the model

The dataset is a compilation of messages categorized with a label of whether spam or ham. There are 5574 observations in this dataset.

The categorization variable was converted into a factor variable in order to find the proportion of spam and ham messages. Below is a tabulation of the proportions

## ham spam   
## 0.8659849 0.1340151

From the table we can see that approximately 87% of the messages are ham and 14% are spam.

A package named tm is then called upon from the R repository for cleaning and processing the text data. Messages are texts composed of words, spaces, numbers and punctuation characters. The goal here is to omit words and other components that are not key in helping us classify the messages.

The first step is to convert our dataset into a corpus (collection of text documents). Then the following transformations are carried out on the corpus.

* The corpus is transformed to lower case since R is case sensitize
* Numbers are omitted from the corpus.
* Stop words are removed from the corpus.
* Punctuation characters are then omitted.
* Reduction of words into their root form (stemming)
* Stripping of white spaces that previously separated the now missing pieces.

All the above transformations can be carried out using packages from the tm package except stemming function contained in the snowballc package.

The Messages are then split into individual components through a process known as tokenization. A token can be defined as a single element of a text string (word). This process is carried out by the DocumentTermMatrix in the tm package. This function takes a corpus and converts it into a data structure called Document Term Matrix in which rows indicate documents (messages) and columns indicate terms (words).

After cleaning and processing the text data we split it into 75% training set and 25% validation set. We then tabulate the proportions of the sets of data to confirm if they are distributed similar to the original dataset.

#Test sample proportions  
prop.table(table(sms\_train\_labels))

## sms\_train\_labels  
## ham spam   
## 0.8648325 0.1351675

prop.table(table(sms\_test\_labels))

## sms\_test\_labels  
## ham spam   
## 0.8694405 0.1305595

From the above code snippet we can see the proportions are similar.

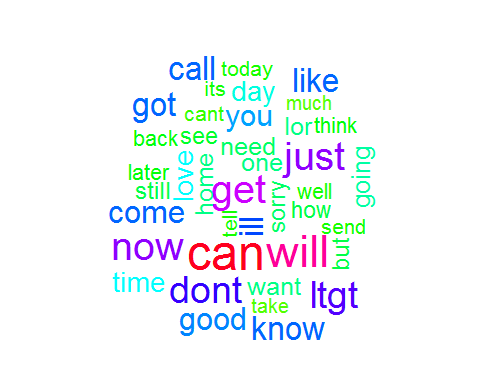
Texts can be visualized by use of word clouds. Word clouds depict the frequency at which words appear in text data. Words appearing more often are shown in larger fonts, while less often words are shown in smaller fonts.

We compare the word clouds from spam and ham messages to gauge whether our classifier would have a chance of success. Below are the code snippets and word clouds from ham and spam messages from the dataset generated by the wordcloud function from the wordcloud package.

#Visualizing both spam and ham texts  
spam <- subset(sms\_raw,type=="spam")  
ham <- subset(sms\_raw,type=="ham")  
  
wordcloud(spam$text,scale=c(3,0.5),max.words = 40,colors = rainbow(50))



wordcloud(ham$text,scale=c(3,0.5),max.words = 40,colors = rainbow(50))



From the above word clouds we can easily tell that he first cloud comes from spam messages and the second from ham. Spam messages most frequent words are quite distinguishable from those of ham.

Next we transform the sparse matrix into a structure than can be used in training the Naïve Bayes classifier. Every feature that appears in the sparse matrix is a representation of every word that appears in our text messages. Some of these features are not helpful in our classification process so we reduce the features by selecting only those that appear in more than five messages which is about 0.1% of our messages. Our Document Term Matrix will now only contain those words that appear in more than five messages.

The Naïve Bayes classifier is typically trained on data containing categorical features but our Document Term Matrix contains counts of words, it uses the presence or absence of words to estimate the probability that a given message is spam or ham.

The counts are converted into either yes or no depending on the presence of a word or feature in a message.

Below is a snippet of the function used for the conversion.

#Convert the counts into categorical variables since Naive Bayes classifier works well on categorical variables  
#Create a fxn  
  
convert\_counts <- function(x){  
 x <- ifelse(x>0,"Yes","No")  
}

Now that the matrix is in a form that can be utilized by our classifier we train the model.

The Naïve Bayes algorithm can be obtained from the e1071 package from R repository.

Below is a snippet for training the filtering model and making predictions

#Training model   
library(e1071)

sms\_classifier <- naiveBayes(sms\_train,sms\_train\_labels)  
  
#Make Predictions  
sms\_test\_pred <- predict(sms\_classifier,sms\_test)

After training the model and making predictions on our test dataset we can use the confusion matrix to see how the model performed

## Cell Contents  
## |-------------------------|  
## | N |  
## | N / Col Total |  
## |-------------------------|  
##   
##   
## Total Observations in Table: 1394   
##   
##   
## | Actual   
## Predicted | ham | spam | Row Total |   
## -------------|-----------|-----------|-----------|  
## ham | 1203 | 20 | 1223 |   
## | 0.993 | 0.110 | |   
## -------------|-----------|-----------|-----------|  
## spam | 9 | 162 | 171 |   
## | 0.007 | 0.890 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 1212 | 182 | 1394 |   
## | 0.869 | 0.131 | |   
## -------------|-----------|-----------|-----------|  
##   
##

mean(sms\_test\_pred==sms\_test\_labels)

## [1] 0.9791966

From the confusion matrix only nine messages were predicted as spam while they were actually ham and 20 messages were predicted as ham while they were actually spam. The model had a general accuracy of approximately 98%.

The performance of the model might be enhanced by the use of the Laplace estimator. This enables words that didn’t appear in the spam or ham messages in our training set to have an indisputable say in the classification process.

Below is a snippet of the code tuning the model by placing laplace estimator =1

#Improve Model  
#The Laplace estimator essentially adds a small number to each of the counts in the frequency  
#table, which ensures that each feature has a nonzero probability of occurring with each class  
sms\_classifier2 <- naiveBayes(sms\_train,sms\_train\_labels,laplace = 1)  
#Make Predictions  
sms\_test\_pred2 <- predict(sms\_classifier2,sms\_test)

The resulting confusion matrix is shown below

#cross tabulate  
library(gmodels)  
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: 1394   
##   
##   
## | Actual   
## Predicted | ham | spam | Row Total |   
## -------------|-----------|-----------|-----------|  
## ham | 1205 | 28 | 1233 |   
## | 0.994 | 0.154 | |   
## -------------|-----------|-----------|-----------|  
## spam | 7 | 154 | 161 |   
## | 0.006 | 0.846 | |   
## -------------|-----------|-----------|-----------|  
## Column Total | 1212 | 182 | 1394 |   
## | 0.869 | 0.131 | |   
## -------------|-----------|-----------|-----------|  
##   
##

mean(sms\_test\_pred2==sms\_test\_labels)

## [1] 0.9748924

After tuning the model only seven messages were classified as spam while they were actually ham. There was an increase in messages classified as ham while they were actually spam. The models general accuracy also slightly decreased.

It is preferable to have small numbers of spam messages slip through our filter than filtering ham messages more aggressively.