**Objective:**

The goal is to build a prediction model that will classify text messages/sms as spam or ham(non-spam)

**Dataset description**

The dataset consists of one message per line. Each line is prefixed with a ham/spam label separated by a “,”. There are 5,574 messages. The summary is given below.

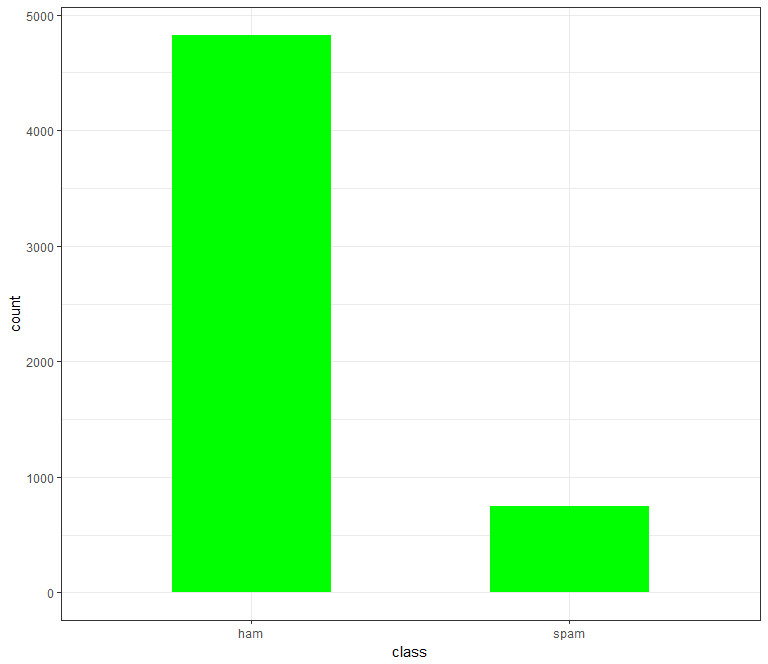
|  |  |  |
| --- | --- | --- |
| Message | Amount | % |
| Hams | 4,827 | 86.60 |
| Spams | 747 | 13.40 |
| Total | 5,574 | 100 |

**File Format**

The files contain one message per line. Each line is composed by two columns: one with label (ham or spam) and other with the raw text. Here are some examples:

|  |  |
| --- | --- |
| class | text |
| ham | Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat... |
| ham | Ok lar... Joking wif u oni... |
| spam | Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's |
| ham | U dun say so early hor... U c already then say... |

The below bar graph shows comparison between two categories (ham & spam).



The below table provides statistics related to the tokens extracted from the corpus. The dataset has a total of 84,579 tokens and spam has in average 8 tokens more than legitimate messages.

|  |  |
| --- | --- |
| Hams | 68,327 |
| Spams | 16,252 |
| Total | 84,579 |
| Avg token per Msg | 15.17 |
| Avg token in Hams | 14.15 |
| Avg token in Spams | 21.75 |

The code snippet is given below

#########################################

# Read the file

#####################################

setwd("D:\\Rajib\\XLRI\\Textmining\\Assignment")

corp <- read.csv("smsspam.csv",header = TRUE,stringsAsFactors = FALSE)

str(corp)

set.seed(1234)

corp<-corp[sample(nrow(corp)),]

# Check the number of spam and ham messages

table(corp$class)

round(prop.table(table(corp$class))\*100, digits = 1)

# ############################

# # Generte Barplot

# ############################

theme\_set(theme\_bw())

ggplot(aes(x=class),data=corp) +

geom\_bar(fill="green",width=0.5)

corphamspam <- corpus(corp$text)

docvars(corphamspam) <-corp$class

#######################

# Get total token in the corpous

######################

corpoustoken<-corphamspam

sum(ntoken(corpoustoken,

removeNumbers=TRUE,

remove\_punct = TRUE,

remove\_url=TRUE,

remove\_hyphens =TRUE,

remove\_symbols = TRUE,

remove\_separators = TRUE,

remove\_numbers = TRUE)

##############################

## No of tokens in Spam

token.spam.plot<-spam.plot

sum(ntoken(token.spam.plot,

remove\_punct = TRUE,

removeNumbers=TRUE,

remove\_url=TRUE,

remove\_hyphens =TRUE,

remove\_symbols = TRUE,

remove\_separators = TRUE,

remove\_numbers = TRUE))

###################################

ham.plot <- corpus\_subset(corphamspam, docvar1 == "ham")

##############################

# No of token in ham

##########################

token.ham.plot<-ham.plot

sum(ntoken(token.ham.plot,remove\_punct = TRUE,

removeNumbers=TRUE,

remove\_url=TRUE,

remove\_hyphens =TRUE,

remove\_symbols = TRUE,

remove\_separators = TRUE,

remove\_numbers = TRUE))

**Pre-processing**

The following steps are performed during data pre-processing

* We use quanteda's corpus() command to construct a corpus from the Text field of raw data
* Attach label field as a document variable to the corpus using the docvars() command. We attach label as a variable directly to our corpus so that we can associate SMS messages with their respective ham/spam label later in the analysis.
* Use set.seed() is to ensure reproducible results.

The code snippet is given below

corphamspam <- corpus(corp$text)

docvars(corphamspam) <-corp$class

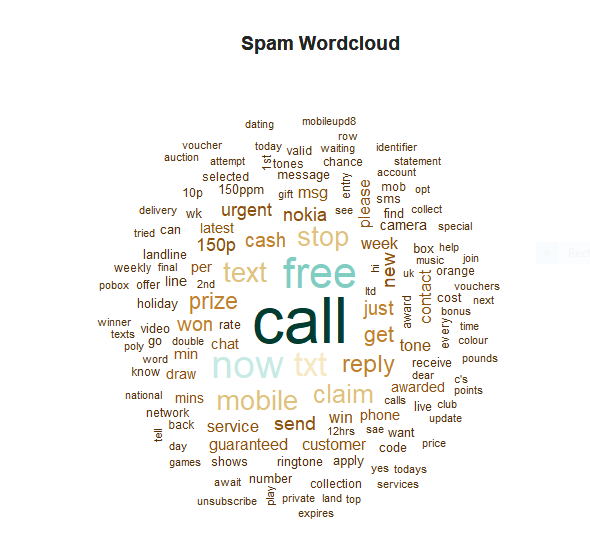
#Add metadata to the Corpus object

metadoc(corphamspam, 'language') <- "english"

**Building the Word clouds for spam**

The following steps are performed

* Subset and filter all the spam text messages from the message corpus
* Generate a document feature matrix which is a sparse matrix consisting of the frequency of words that occur in a document
* Transformed raw text with normalization techniques such as tokenization & case folding
* Only the alphabetical & alphanumerical tokens are stored. Tokens with punctuation marks, strip whitespaces, special characters, url & numbers, words like lt ,gt , ü, po are ignored
* Perform language-specific pre-processing techniques such as stop word removal.
* No word stemming is done here.
* The effect of abbreviations and misspellings in the messages are ignored,
* Frequency count of each token per message is stored as Bag of Words
* Picks and selects the most commonly occurring words in the sentences i.e the words having the highest frequencies and plots them, the more the frequency of a particular word the greater is the size of the word in the word-cloud.

Word cloud for Spam messages–

The group of tokens (send, ringtone, text, tone, free, reply, mobile) related to mobile/ringtone associated with message appear to be most prominent in spam message.

Similarly token such as (txt, win, cash, 150p, prize, guaranteed, urgent, claim, draw, cash) related to completions & prize associated with message appear to be most prominent in spam message.

The below table shows twenty tokens that most appeared in spam messages

|  |  |
| --- | --- |
| Token | Token Number of spam Msg |
| Call | 355 |
| Free | 223 |
| Now | 199 |
| Txt | 160 |
| mobile | 127 |
| Text | 124 |
| Stop | 123 |
| Claim | 113 |
| reply | 104 |
| prize | 92 |
| Get | 86 |
| Just | 79 |
| Cash | 76 |
| Won | 73 |
| Send | 71 |
| New | 69 |
| Nokia | 67 |
| 150p | 65 |
| Win | 60 |
| Tone | 59 |

The code snippet is given below

###################################

### spam world cloud

###################################

spam.plot <- corpus\_subset(corphamspam, docvar1 == "spam")

spam.plot <- dfm(spam.plot, tolower = TRUE,

removeNumbers=TRUE,

remove\_punct = TRUE,

remove\_url=TRUE,

removeNumbers=TRUE,

remove\_hyphens =TRUE,

remove\_symbols = TRUE,

remove\_separators = TRUE,

remove\_numbers = TRUE,

remove=c("lt","gt","ur","ü","po","cs",letters,stopwords("english")),

valuetype="fixed",

verbose=TRUE)

topfeatures(spam.plot, 50)

spam.col <- brewer.pal(10, "BrBG")

spam.cloud <- textplot\_wordcloud(spam.plot, min\_count = 16,

random\_order = FALSE,random\_color = FALSE,

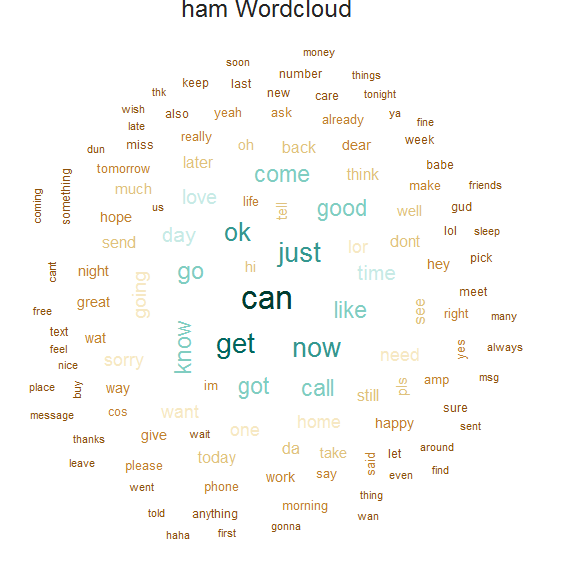
rotation = 0.1,

min\_size=0.5,color = spam.col)

title("Spam Wordcloud", col.main = "grey14")

**Building the Wordclouds for ham**

* Subset and filter all the ham text messages from the message corpus
* Generate a document feature matrix which is a sparse matrix consisting of the frequency of words that occur in a document
* Transformed raw text with normalization techniques such as tokenization & case folding
* Only the alphabetical & alphanumerical tokens are stored. Tokens with punctuation marks, special characters, url & numbers, words like lt ,gt , ü, po are ignored
* No word stemming is done here.
* The effect of abbreviations and misspellings in the messages are ignored,
* Perform language-specific pre-processing techniques such as stop word removal.
* Frequency count of each token per message is stored as Bag of Words
* Picks and selects the most commonly occurring words in the sentences i.e the words having the highest frequencies and plots them, the more the frequency of a particular word the greater is the size of the word in the word-cloud.
* Size of the smallest word is min\_size(0.9)
* Size of the largest word is max\_size(4)
* Words with frequency below (min\_count ) 50 ignored



The below table shows twenty tokens that most appeared in ham messages

|  |  |
| --- | --- |
| Token | Token Number of ham Msg |
| can | 378 |
| get | 303 |
| just | 296 |
| now | 295 |
| ok | 285 |
| go | 251 |
| know | 237 |
| got | 233` |
| like | 233 |
| Call | 233 |
| good | 231 |
| come | 230 |
| time | 199 |
| day | 199 |
| love | 197 |
| Going | 169 |
| want | 165 |
| lor | 162 |
| home | 161 |
| need | 157 |

The code snippet is given below

###################################

### ham world cloud

###################################

ham.plot <- corpus\_subset(corphamspam, docvar1 == "ham")

ham.plot <- dfm(ham.plot, tolower = TRUE,

remove\_punct = TRUE,

remove\_url=TRUE,

removeNumbers=TRUE,

remove\_hyphens =TRUE,

remove\_symbols = TRUE,

remove\_separators = TRUE,

remove\_numbers = TRUE,

remove=c("lt","gt","ur","ü","po","cs",letters,stopwords("english")),

valuetype="fixed",

verbose=TRUE)

topfeatures(ham.plot, 50)

ham.col <- brewer.pal(10, "BrBG")

par(mar=rep(0.4, 4))

plot(0:1, 0:1, type = "n", axes = FALSE, ann = FALSE)

ham.cloud <- textplot\_wordcloud(ham.plot, min\_count =50,min\_size=0.9,max\_size = 4,

random\_order = FALSE,random\_color = FALSE,

rotation = 0.1,

fixed\_aspect = TRUE,

color = ham.col)

title(main = list("ham Wordcloud", cex = 1.5,

col = "grey14", font = 1))

**Predictive Modelling**

Naive Bayes text classifier is used to compute the probabilities of a message being (spam,ham)

i.e **P(Y=Spam|message) and P(Y=ham|message)**  
Naive Bayes classifiers are a class of simple linear classifiers which use conditional probability models based on **Bayes Theorem**

Where  are the number of inputs and Y is a categorical response variable and are the number of class labels.

* Extracting tokens for all messages in the dataset will result in 9,148 features. However, not all of these features are useful in the classification. Going through the extracted tokens, we removed the ones with less than 5 and more than 500 times frequency in the dataset, since those tokens are either too rare or too common, and do not contribute to the content of the messages.
* Divide the corpus in two parts: the first 80% of the messages were separated for training (4,459 messages) and the remainder ones for testing (1,115 messages).
* As all the messages are fairly short, we did not use any kind of method to reduce the dimensionality of the training space

The code snippet is given below

dfmhamspam <- dfm(corphamspam,

tolower = TRUE,

stem = TRUE,

removeNumbers=TRUE,

remove\_punct = TRUE,

remove\_url=TRUE,

remove\_hyphens =TRUE,

remove\_symbols = TRUE,

remove\_separators = TRUE,

remove\_twitter = TRUE,

valuetype="fixed",

verbose=TRUE

)

dfmhamspam <- dfm\_trim(dfmhamspam, min\_docfreq = 5, max\_docfreq = 500,sparsity = NULL)

summary(dfmhamspam)

docvars(corphamspam) <-corp$class

intTraingSet<-as.integer(nrow(corp)\*0.80)

intTestSet<-intTraingSet+1

corp.train<-corp[1:intTraingSet,]

corp.test<-corp[intTestSet:nrow(corp),]

dfmhamspam.train<- dfmhamspam[1:intTraingSet]

dfmhamspam.test<-dfmhamspam[intTestSet:nrow(corp),]

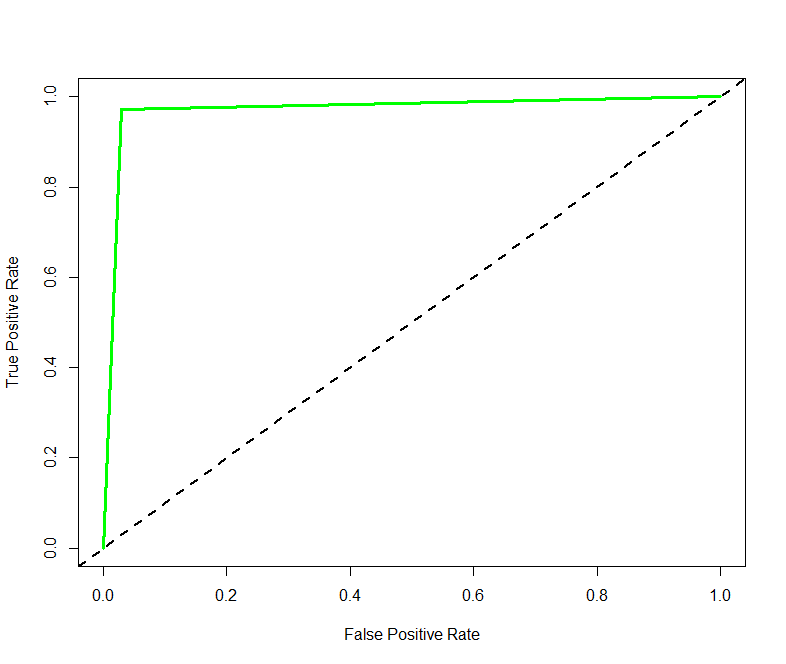
nb <- textmodel\_nb(dfmhamspam.train,corp.train[,1],smooth = 1 )

**Model Diagnosis**

* The accuracy of the model is 97.39%(spam SMS classified )
* The confusion matrix of the model is given below

|  |  |  |  |
| --- | --- | --- | --- |
| Predicted Class | Actual Class | | |
|  | ham | spam |
| ham | 949(TP) | 2(FP) |
| spam | 27(FN) | 137(TN) |

* The model classified 949 ham messages correctly as ham, and 2 ham messages incorrectly as spam.
* The model incorrectly classified 27 spam messages as ham, but 137 spam messages correctly as spam.
* Precision=TP/TP+FP=949/949+2=0.9978 which means model correctly classifies 99.78% of ham message
* Recall=TP/TP+FN=949/949+27=0.9723 which means 97.23% is the percentage of positive instances that were correctly identified.
* The ROC curve is given below is 97.89%



The code snippet for accuracy

pred\_data <- predict(nb, dfmhamspam.test)

predicted\_class <- pred\_data$nb.predicted

tb<-table(predicted\_class,corp.test$class)

accuracy<-sum(diag(tb))/sum(tb)

accuracy

The code snippet for ROC curve

predvec <- ifelse(predicted\_class=="ham", 1, 0)

realvec <- ifelse(corp.test$class=="ham", 1, 0)

pred <- prediction(predvec,realvec)

perf <- performance(pred, measure = "tpr", x.measure = "fpr")

plot(perf, main = "ROC curve for Naive Bayes Classifier",col = "blue", lwd = 3)

abline(a = 0, b = 1, lwd = 2, lty = 2)

perf.auc <- performance(pred, measure = "auc")

unlist([perf.auc@y.values](mailto:perf.auc@y.values))

**Run Model against new data set and Predict the New data set**

We run the above model against a new dataset to check if the model classifies the messages (ham or spam) . The dataset has only message field. The class field is blank.

The following steps are performed and result of the test is given below.

* Get 2500 sms (Refer. smsspam\_predict.csv) data without classification and predicting the new dataset against the model
* We found model correctly classified message ham/smpam for the new dataset

The sample code is given below

ab<-read.csv("smsspam\_predict.csv",header=TRUE,stringsAsFactors=FALSE)

test\_dfm <-dfm(corpus(ab$text),tolower = TRUE,

remove\_punct = TRUE,

remove\_url=TRUE,

remove\_hyphens =TRUE,

remove\_symbols = TRUE,

remove\_separators = TRUE,

remove\_numbers = TRUE,

remove=c("lt","gt","ur","ü","po","cs",letters,stopwords("english")),

valuetype="fixed",

verbose=TRUE)

test\_dfm <- dfm\_select(test\_dfm, dfmhamspam.train,selection = "keep")

## Make predictions on test set

predvec\_new\_1<- predict(nb, test\_dfm, type = "response")

predicted\_class <- predvec\_new\_1$nb.predicted

predvec<-ifelse(predicted\_class=="ham",1,0)

ab$class<-predicted\_class

write.csv(ab,"spsspam\_new\_predict.csv"

**Conclusion:**

In this study Naïve Byes Classification model is used for categorizing the SMS messages to spam and non-spam.

The accuracy of the model is 97.39% . This can be considered as a very good baseline.

Also in SMS spam filtering, the ratio of incorrectly filtered SMS (blocked ham ) is much more important than SC(spam caught) , accuracy, because wrongly labelled SMS spam can easily be deleted with a time lose, but the wrongly labelled ham message can be unread by the user, which means the SMS may be deleted after an automatic process.

The automatic SMS spam filtering task is quite difficult. The key challenges to build the model is given below

* Absence of public and real datasets
* The mobile phone messages often have a lot of abbreviations and idioms that may affect the filters accuracy.
* Low number of features in each message

Future work should consider to use different strategies to increase the dimensionality of the feature space. We also need to compare the performance of the model with other established machine learning methods so that, it can be used as a good baseline for further comparison.

Proposed model can be enhanced in following ways:

* Multilingual spam email classification
* Develop a technique that can catch the sentimental phrases and train methodology for those spams.

**ANNEXTURE**

1. 1 R Code(Text\_Mining\_Assignment-I\_Rajib\_Mandal\_TA17002.R)
2. Program Output(Text\_Mining\_Assignment-I\_Console\_Rajib\_Mandal\_TA17002.txt)
3. External Validation Dataset(smsspam\_predict.csv)