**Introduction To Text Analytics with R: SVD with R**

Ruchitha Chowdary B-120015080

Varanasi Chaitanya Sai-120015107

Procedure:

1)A csv file which contains a dataset is created and stored in local directories.

2)Libraries are imported,packages are installed.

3)spam is the csv file that is imported.

4)spam is read.

5)data frame 1 and 2 are labelled as label and text

6)checks if any missing values are present

7)if missing values are present they are removed

8)checks the length

9)class label is converted into factor

10)Distribution of class label is taken as ham and spam

11)Distribution of length of each message is seen

12)visualisation and distribution of ham and spam is noted

13)Training,validation and test is done for the test data set

14)Library caret is imported

15)caret-used to create a 70%/30% split up

16)Proportions are verified

17)escaped ampersand character implemented

18)tokenize messages are all the words

19)lower case tokens-used for converting all tokens to lowercase

20)stemming-used to stem the tokens from the set(like training is converted to train).

21)marix-used for taking specified values from the dataset.

Code in R:

# Install all required packages.

install.packages(c("ggplot2", "e1071", "caret", "quanteda",

"irlba", "randomForest"))

#LINK FOR DATASET

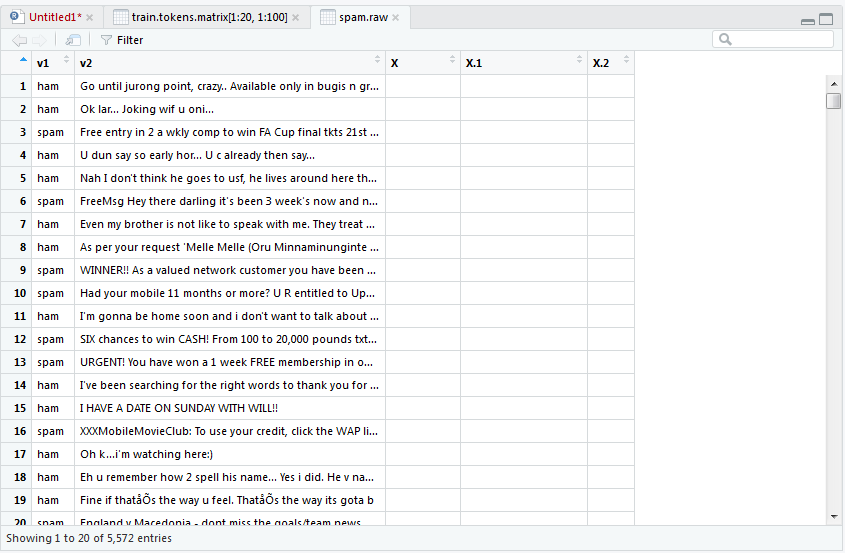
#<https://drive.google.com/drive/folders/19rD_d7qGvTK9XKWQlETgyIgbHIXpUa3R?usp=sharing>

# Load up the .CSV data and explore in RStudio.

spam.raw <- read.csv("spam.csv", stringsAsFactors = FALSE, fileEncoding = "UTF-16")

View(spam.raw)

#

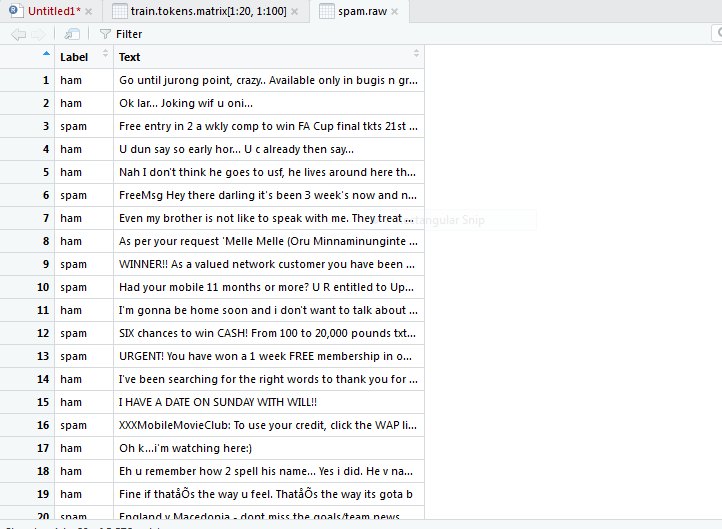


# Clean up the data frame and view our handiwork.

spam.raw <- spam.raw[, 1:2]

names(spam.raw) <- c("Label", "Text")

View(spam.raw)



# Check data to see if there are missing values.

length(which(!complete.cases(spam.raw)))

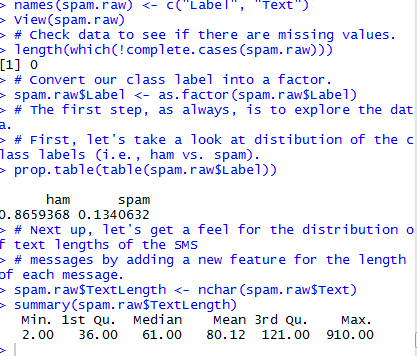
# Convert our class label into a factor.

spam.raw$Label <- as.factor(spam.raw$Label)

# The first step, as always, is to explore the data.

# First, let's take a look at distibution of the class labels (i.e., ham vs. spam).

prop.table(table(spam.raw$Label))

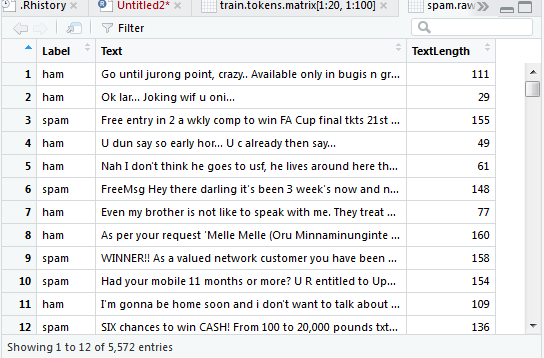


# Next up, let's get a feel for the distribution of text lengths of the SMS

# messages by adding a new feature for the length of each message.

spam.raw$TextLength <- nchar(spam.raw$Text)

summary(spam.raw$TextLength)



# Visualize distribution with ggplot2, adding segmentation for ham/spam.

library(ggplot2)

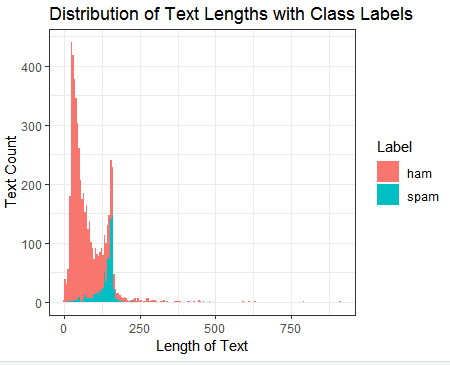
ggplot(spam.raw, aes(x = TextLength, fill = Label)) +

theme\_bw() +

geom\_histogram(binwidth = 5) +

labs(y = "Text Count", x = "Length of Text",

title = "Distribution of Text Lengths with Class Labels")



# At a minimum we need to split our data into a training set and a

# test set. In a true project we would want to use a three-way split

# of training, validation, and test.

#

# As we know that our data has non-trivial class imbalance, we'll

# use the mighty caret package to create a randomg train/test split

# that ensures the correct ham/spam class label proportions (i.e.,

# we'll use caret for a random stratified split).

library(caret)

help(package = "caret")

# Use caret to create a 70%/30% stratified split. Set the random

# seed for reproducibility.

set.seed(32984)

indexes <- createDataPartition(spam.raw$Label, times = 1,

p = 0.7, list = FALSE)

train <- spam.raw[indexes,]

test <- spam.raw[-indexes,]

# Verify proportions.

prop.table(table(train$Label))

prop.table(table(test$Label))

# Text analytics requires a lot of data exploration, data pre-processing

# and data wrangling. Let's explore some examples.

# HTML-escaped ampersand character.

train$Text[21]

# HTML-escaped '<' and '>' characters. Also note that Mallika Sherawat

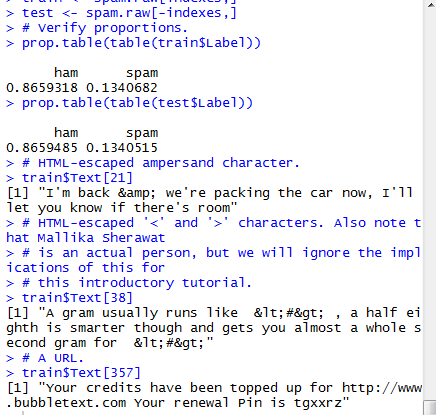
# is an actual person, but we will ignore the implications of this for

# this introductory tutorial.

train$Text[38]

# A URL.

train$Text[357]



# There are many packages in the R ecosystem for performing text

# analytics. One of the newer packages in quanteda. The quanteda

# package has many useful functions for quickly and easily working

# with text data.

library(quanteda)

help(package = "quanteda")

# Tokenize SMS text messages.

train.tokens <- tokens(train$Text, what = "word",

remove\_numbers = TRUE, remove\_punct = TRUE,

remove\_symbols = TRUE, remove\_hyphens = TRUE)

# Take a look at a specific SMS message and see how it transforms.

train.tokens[[357]]

# Lower case the tokens.

train.tokens <- tokens\_tolower(train.tokens)

train.tokens[[357]]

# Use quanteda's built-in stopword list for English.

# NOTE - You should always inspect stopword lists for applicability to

# your problem/domain.

train.tokens <- tokens\_select(train.tokens, stopwords(),

selection = "remove")

train.tokens[[357]]

# Perform stemming on the tokens.

train.tokens <- tokens\_wordstem(train.tokens, language = "english")

train.tokens[[357]]



# Create our first bag-of-words model.

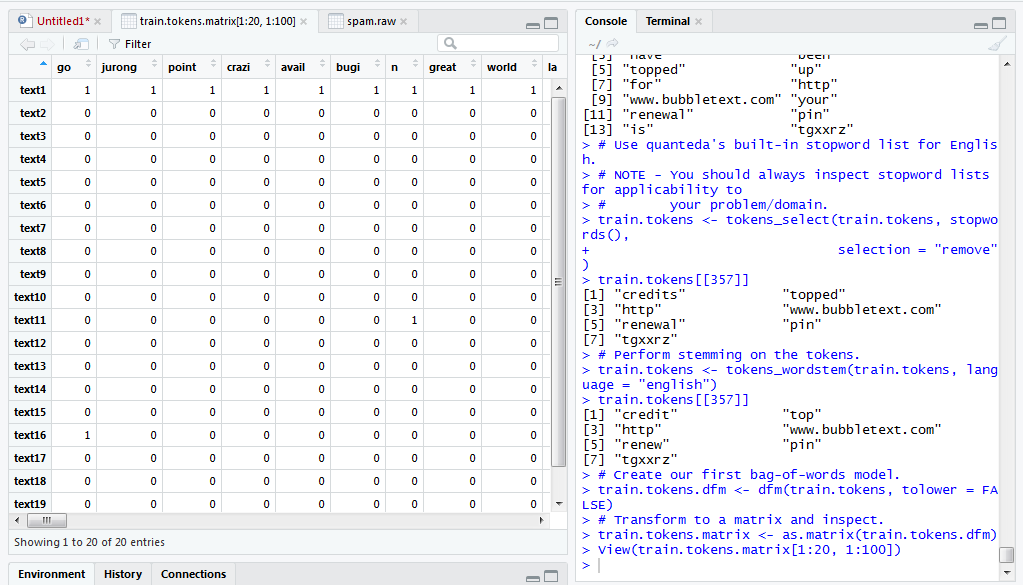
train.tokens.dfm <- dfm(train.tokens, tolower = FALSE)

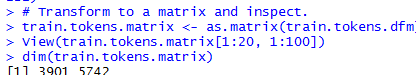
# Transform to a matrix and inspect.

train.tokens.matrix <- as.matrix(train.tokens.dfm)

View(train.tokens.matrix[1:20, 1:100])

dim(train.tokens.matrix)





THANK YOU