# CASE STUDY ON SENTIMENT ANALYSIS

**Group No**.: Group 20

**Student Names**: Archisha Dama and Arijeet Biswas

## Executive Summary:

The goal of our study is to clean the data, analyse and visualise it. Predict the comments showing signs of hate crime.

Data was taken from Kaggle. We took half the data of pro-ISIS fanboys' Tweet and the other half from normal Tweets ( having a combination of negative, positive or neutral sentiment). This will ensure that we have good variety of data to train our model. Total size of the dataset is about 10,000 records. Key attributes such as Type and Tweet are taken into consideration.

Data is not of good quality as it is not in readable format for Naïve Bayes to work on. Categorical variables are not factored.
Cases, numbers, punctuation, filler/stop words, white spaces needs to be processed so that Naïve Bayes classifier can work on the dataset

The following steps are carried out for Cleansing and standardizing data such as converting tweets into documents, splitting text documents into words and finally creating training and test data set.
For this classification problem data mining technique such as naïve bayes is used to predict the outcome and it is further optimized through Laplace transformation.
The outcome is visualized through confusion matrix with true positive, true negative false positive, false negative.

## I. Background and Introduction

- The background:

Social media is not just a platform where people talk to each other, It has become a medium where people express their interests, Share their views and displeasures. Internet is a major source of spreading terrorism through speeches and videos. It has become necessary to analyze these information to avoid any form of hate crime or violence.

- The problem:

Social media is not just a platform where people talk to each other, It has become a medium where people express their interests, Share their views and displeasures. Internet is a major source of spreading terrorism through speeches and videos. It has become necessary to analyze this information to avoid any form of hate crime or violence. With machine learning algorithms it is now possible to create profile of users based on their comments. This became very handy in many security domains as follows:

Business: Companies use Twitter Sentiment Analysis to develop their business strategies, to assess customers' feelings towards products or brand, how people respond to their campaigns or product launches and also why consumers are not buying certain products.

Politics: In politics Sentiment Analysis Dataset Twitter is used to keep track of political views, to detect consistency and inconsistency between statements and actions at the government level. Sentiment Analysis Dataset Twitter is also used for analyzing election results.

Public Actions: Twitter Sentiment Analysis also is used for monitoring and analyzing social phenomena, for predicting potentially dangerous situations and determining the general mood of the blogosphere.

Also, this technique is very valuable to the marketing domain by analyzing the reviews and the comments of users and automatically detect positive and negative reviews.

- Goal of the study:

Our proposed project aims at text mining to check prominence of a keyword commonly used in terrorist activities. So, this system can be used to point out users spreading hate crime. The goal of the study is To clean the data, analyse and visualise it. Predict the comments showing signs of hate crime.
Tweet analysis can be done through popular package such as "e1071" in R specifically used by Naïve Bayes.

- Possible solution:

We will use Naïve Bayes algorithm for text classification which has become the de facto standard for text classification. Naïve Bayes is named as such because it makes some "naïve' assumptions. But still the algorithm is considered to be very versatile and can be used across many types of conditions. This might be because it is not important to obtain a precise estimate of probability, so long as the predictions are accurate. For instance, if our hate message filter correctly identifies hate messages, it doesn't matter whether it was 51 percent or 99 percent confident in its prediction.

## II. Data Exploration and Visualization

To gather the data many options are possible.
In some previous paper studies, they built a program to collect automatically a corpus of tweets based on two classes, "positive" and "negative", by querying Twitter with two type of emoticons:
● Happy emoticons, such as ":)", ":P", ":)" etc.
● Sad emoticons, such as ":(", ":'(", "=(".
Others make their own dataset of tweets my gathering and annotating them manually which very long and fastidious. Additionally to find a way of getting a corpus of tweets, we need to be sure of having a balanced data set, meaning we should have an equal

number of positive and negative tweets, but it needs also to be large enough. Indeed, more the data we have, more we can train our classifier and more the accuracy will be. After many researches, we found a dataset of 10,000 tweets in English coming from two sources: Kaggle and Sentiment140.

It is composed of six columns that are Type, ID, Date, Query , Name and Tweet
We are only interested by the Sentiment column corresponding to our Type class taking a binary value, 0 if the tweet is negative, 1 if the tweet is neutral and 4 if the tweet is positive. The SentimentText columns containing the tweets in a raw format.

Packages such as "wordcloud" and "RColorBrewer" are used for visualizing more effectively with a color palette. Corpus of data is separated out and positive and negative tweets are visualized individually to find the most frequently displayed words. The Word Clouds also helps us visually analyze which words are getting considered most while training our dataset.

The Word Cloud of the complete dataset is:

Now, the Word Cloud showing 40 most used words in non-hate messages.


And, the Word Cloud showing 40 most used words in hate messages.

## III. Data Preparation and Preprocessing

Only the key attributes of the data are taken to proceed with the data mining ie Type and Tweet.
Since the data set is in orderly manner ,it is randomized so that equal significance is given to both positive and negative tweets. We are using the sample() function to do it and to get the same random sets, we have used set.seed(1). So now the same set of data can be analyzed for its structure and quality by multiple users.
It is found that data is not of good quality and hence it is cleaned and standardized.
The type of the data is factored as positive and negative tweet type.
The tweets are converted to a bag of tokenized words which converts tweets into text documents by using corpus function. Each corpus is inspected for the message and characters in the message.
The data preprocessing is done as the data is incomplete.
Lacking attribute values: lacking certain attributes of interest, or containing only aggregate data.
Noisy: containing errors or outliers.
Inconsistent: containing discrepancies in codes or names.

The data cleaning and standardizing is done with the following steps:

1. Removal of Punctuations
2. Removal of Numbers
3. Changing text to lower case
4. Removing Stop Words ( filler words) like like, the, then etc
5. Removal of extra white space
6. Removing some more special characters which were not removed by replacing them with white spaces
7. Stemming the words to their root form, so that same root words are considered under a single feature

Also we found that some words split differently than when they are cleaned before tokenization. Which simply means the order of cleaning operations affect our prediction accuracy. We will walk more about it in Performance Evaluation.

The data is cleaned up by converting tweets into lower case
Output:
> inspect(corpus[1:3])
<<SimpleCorpus>>
Metadata: corpus specific: 1, document level (indexed): 0
Content: documents: 3

[1] Sadly though, I've never gotten to experience the post coitus cigarette before, and now I never will.
[2] @136BintRoses @DunyaTravelle4 @dunyatraveller5 of course itsn't you, read the profile, itsn't Amal)
[3] @WeaboIsNEET just wait... seems there'll be a new installation after Tibsho video.

> corpus <- tm_map(corpus,content_transformer(tolower))
> as.character(corpus[[1]])
[1] "sadly though, i've never gotten to experience the post coitus cigarette before, and now i never will."

Removing numbers from the tweets
Output:
> corpus <- tm_map(corpus, removeNumbers)
> as.character(corpus[[2]])
[1] "@bintroses @dunyatravelle @dunyatraveller of course itsn't you, read the profile, itsn't amal)"

Removing stop words
Output:
> corpus <- tm_map(corpus, removeWords, stopwords())
> as.character(corpus[[1]])
[1] "sadly though, never gotten experience post coitus cigarette , now never will."

Removing punctuation
Output:
> corpus <- tm_map(corpus, removePunctuation)
> as.character(corpus[[1]])
[1] "sadly though never gotten experience post coitus cigarette now never will"

Converting tweet into root form
Output:
> corpus <- tm_map(corpus, stemDocument)
> as.character(corpus[[1]])
[1] "sad though never gotten experi post coitus cigarett now never will"

Replace special characters with white space

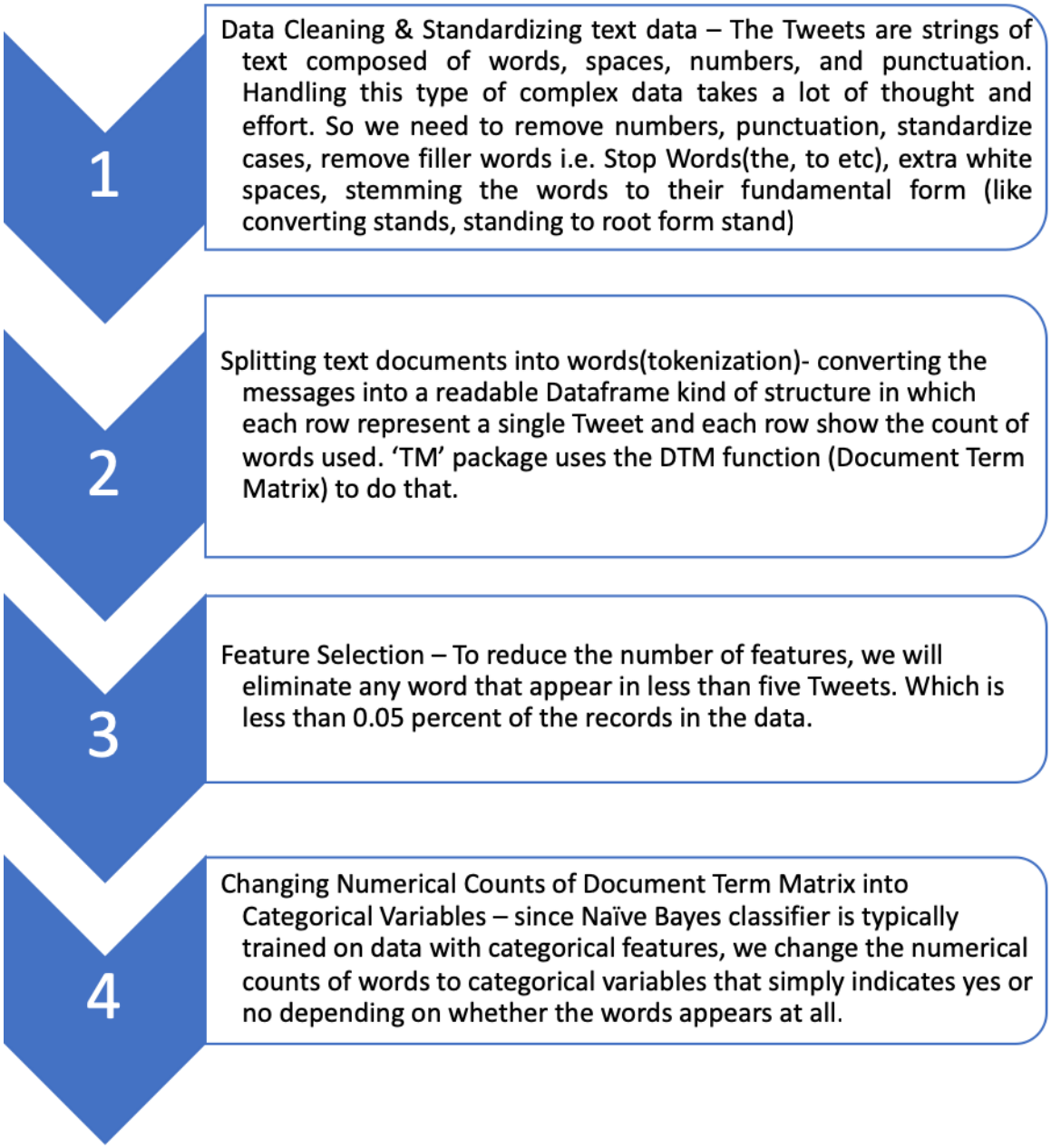
Output:
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
corpus <- tm_map(corpus, toSpace, "/")
corpus <- tm_map(corpus, toSpace, "@")
corpus <- tm_map(corpus, toSpace, "\\|")
> as.character(corpus[[2]])
[1] "bintros dunyatravell dunyatravel cours itsnt read profil itsnt amal"

Removing white space
Output:
corpus.clean <- tm_map(corpus, stripWhitespace)
as.character(corpus.clean[[1]])

## IV. Data Mining Techniques and Implementation

Our problem is a classification problem in which we are classifying if messages are supposed to be Hate Messages or Non-Hate Messages. If the probability of predicting Hate Message is more than 51% then our prediction will be Hate Message. In our case study, we are mostly using 'TM" library for doing text mining in R. It has lot of useful functions that makes work easier for fetching and modifying data easier.
The steps involved in implementation part are as follows:

1. Data Preparation- The various steps of data preparation are shown below




2. Training the model- we train our model with in which we are using 80 percent of our Tweets (total 8000 Tweets) to train the model on Naïve Bayes Classifier by using naiveBayes() function from "e1071" package.

3. Evaluating model performance- we evaluate the performance of the model by predicting the performance on the remaining 20 percent training set (2000 Tweets) through Confusion Matrix.

## V. Performance Evaluation

For evaluating the predicting accuracy we are using Confusion Matrix. At first our model's accuracy was coming out to be just 71.14 %. After which we tried to optimize the predictive accuracy of our Naïve Bayes Classifier through the following steps:

1. Reducing the number of class from 3 to 2 – at first we had 3 classes. We took the hate messages as "0" i.e. negative sentiment, then we took two types of non-hate messages. Type "4" i.e. positive sentiment and "1" i.e. neutral sentiment. Finally the 2 classes which we took were "Hate_Mssg" and "Non_Hate_Mssg".
2. Increasing the proportion of the dataset we want to predict. At first we took 1/3 data of hate message, 1/3 data of non-hate positive sentiment message and 1/3 data of non-hate neutral sentiment message. But to get the best prediction accuracy we had to take 60% of hate message and remaining 40% of non-hate message.
3. Increasing the size of dataset from 6000 to 10000 also increased the prediction accuracy a lot. Now we are working with 6000 hate messages and 4000 non-hate messages.
4. Changing the sequence of data cleaning and standardizing also improved the prediction accuracy. We tried all the steps of cleaning as mentioned in Section III. in different sequences. It turns out that in our case cleaning text data before or after stemming makes some prediction difference. One good example was not removing white space which brought down our model's predictive accuracy from 92.25% to 67.03%. Also we found out that optimizing our Naïve Bayes model with Laplace Transformation doesn't work with white space. The final sequence of data cleaning and standardization is shown in Appendix: R Part-2.
5. The training set to validation set ratio of 80/20 worked best for us.
6. Using Laplace Transformation to optimize the prediction of Naïve Bayes- we added a small number (in our case count 1) to each of the counts in the frequency table to ensure that each feature has a non-zero probability of occurring with each class. In our case after going through the above mentioned 5 steps we were getting prediction accuracy of 87.09%. By applying Laplace Transformation we were able to increase our prediction accuracy to 92.25%.

For more reference please refer to Appendix: R Part-1 for our initial Naïve Bayes prediction model, and Appendix: R Part-2 for our optimized Naïve Bayes & Laplace Transformation prediction model.

## VI. Discussion and Recommendation

In order to be sure that our model is capturing a pattern, we must use different tools and metrics to evaluate the system. The gained results will give us understandings on the reliability of the model and where it can be employed. Also these results will give us a direction for better improvements in the future.

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to distinguish the sentiments of s corpus of texts very precisely because of the complexity in the English language and even more if we consider other languages such as Japanese, Tamil or Chinese. In this project we tried to show the basic way of classifying

tweets into positive or negative category using Naive Bayes as baseline and how language models are related to the Naive Bayes and can produce better results. We could additionally improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together.

## VII. Summary

In this work, we have presented the task of automatically classifying the users as promoting hate crime or not . This work is of interest for a number of potential applications like security, politics, field of education etc. We have performed our experiment on the 10000 training data provided by the Kaggle and ISIS Twitter data. The tests presented in this report are done with combining these two test sets. We have acquired the accuracy of 92.25% in hate crime classifications. In our future work, the accuracy of the classification can be improved by performing deeper features engineering for finding specific area the comments are targeted on.

References:

1. http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-simple-steps-you-should-know
2. https://rpubs.com/cen0te/naivebayes-sentimentpolarity
3. Lecture slides from the Stanford NLP Coursera course by Dan Jurafsky and Christopher Manning: https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html
4. Machine Learning with R by Brett Lantz: https://www.packtpub.com/big-data-and-business-intelligence/machine-learning-r

## Appendix: R Code for use case study

Part-1: Initial Naïve Bayes Prediction Model

```
install.pacakges("RtextTools")
install.packages("e1071")
install.packages("dplyr")
install.packages("caret")
install.packages("doMC")
install.packages("SnowballC") # for text stemming
install.packages("wordcloud") # word-cloud generator
install.packages("RColorBrewer") # color palettes
library(tm)
library(RTextTools)
library(e1071)
library(dplyr)
library(caret)
library(doMC)
registerDoMC(cores=detectCores())
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
#set directory
setwd("~/Desktop/Data Mining in Engg/Case Study/working_folder")
getwd()
#read data
library(readxl)
project_xlsb <- read_excel("project.xlsb2.xlsx", sheet = "Sheet1")
View(project_xlsb)
# separate only columns needed for analysis
df <- project_xlsb[ , -c(2,3,4,5)]
glimpse(df)
#randomise data set
set.seed(1)
df <- df[sample(nrow(df)), ]
df <- df[sample(nrow(df)), ]
glimpse(df)
# analyse the data
str(df)
#factorise the type
df$TYPE<- as.factor(df$TYPE)
str(df$TYPE)
table(df$TYPE)

#bag of words tokenization
```

```
corpus <- Corpus(VectorSource(df$TWEET))
corpus
inspect(corpus[1:3])#no of characters in message
print(corpus)
as.character(corpus[[1]]) #displays the message
#data clean up
#convert to lower case
corpus <- tm_map(corpus,    content_transformer(tolower))
as.character(corpus[[1]])
# remove numbers
corpus <- tm_map(corpus, removeNumbers)
as.character(corpus[[1]])
# remove stop words
corpus <- tm_map(corpus, removeWords, stopwords())
as.character(corpus[[1]])
# remove punctuation
corpus <- tm_map(corpus, removePunctuation)
as.character(corpus[[1]])
# words to root form
corpus <- tm_map(corpus, stemDocument)
as.character(corpus[[1]])
#replace special characters with white space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
corpus <- tm_map(corpus, toSpace, "/")
corpus <- tm_map(corpus, toSpace, "@")
corpus <- tm_map(corpus, toSpace, "\\|")
# remove white space
corpus.clean <- tm_map(corpus, stripWhitespace)
as.character(corpus.clean[[1]])

#dtm
dtm <- DocumentTermMatrix(corpus.clean)
inspect(dtm[40:50, 10:15])
# word cloud
wordcloud(corpus.clean, min.freq = 10, max.words=200, random.order=FALSE,
rot.per=0.35,
        colors=brewer.pal(8, "Dark2"))
positive <- subset(corpus.clean, df$TYPE == 4 )
negative <- subset(corpus.clean, df$TYPE == 0)
neutral <- subset(corpus.clean, df$TYPE==1)
wordcloud(positive, max.words = 30, scale = c(3, 0.1))
wordcloud(negative, max.words = 40, scale = c(3, 0.1))
wordcloud(neutral, max.words = 40, scale = c(3, 0.1))
# frequent terms in DTM
findFreqTerms(dtm, lowfreq = 4)
```

```
#partitioning data
df.train <- df[1:4800,]
df.test <- df[4801:5999,]

dtm.train <- dtm[1:4800,]
dtm.test <- dtm[4801:5999,]

corpus.clean.train <- corpus.clean[1:4800]
corpus.clean.test <- corpus.clean[4801:5999]
#feature selection
dim(dtm.train)
fivefreq <- findFreqTerms(dtm.train, 5)
length((fivefreq))
dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary =
fivefreq))

dim(dtm.train.nb)
dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary =
fivefreq))
dim(dtm.test.nb)
# naive bayers alogorithm
# Function to convert the word frequencies to yes (presence) and no (absence) labels
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
# Apply the convert_count function to get final training and testing DTMs
trainNB3 <- apply(dtm.train.nb, 2, convert_count)
testNB3 <- apply(dtm.test.nb, 2, convert_count)
# Train the classifier with Naive Bayes
classifier3 <- naiveBayes(trainNB3, df.train$TYPE)

# test the predictions
pred3 <- predict(classifier3, newdata=testNB3)

# confusion matrix
conf3.mat <- confusionMatrix(pred3, df.test$TYPE)
conf3.mat
```

Part-2: Optimized Naïve Bayes & Laplace Transformation Prediction Model

```
getwd()
setwd("~/Desktop/Data Mining in Engg/Case Study/working_folder")
install.packages("RtextTools")
install.packages("e1071")
install.packages("dplyr")
install.packages("caret")
install.packages("doMC")
install.packages("SnowballC") # for text stemming
install.packages("wordcloud") # word-cloud generator
install.packages("RColorBrewer") # color palettes
library(tm)
library(RTextTools)
library(e1071)
library(dplyr)
library(caret)
library(doMC)
registerDoMC(cores=detectCores())
library("SnowballC")
library("wordcloud")
library("RColorBrewer")
#read data
library(readxl)
project_xlsb <- read_excel("project.xlsb.xlsx", sheet = "Sheet1")
View(project_xlsb)

# separate only columns needed for analysis
df <- project_xlsb[ , -c(2,3,4,5)]
glimpse(df)
str(df)

#randomise data set
set.seed(1)

df <- df[sample(nrow(df)), ]
df <- df[sample(nrow(df)), ]
glimpse(df)
```

```
# analyse the data
str(df)

#factorise the type
df$TYPE<- as.factor(df$TYPE)
str(df$TYPE)
table(df$TYPE)

#bag of words tokenization
corpus <- Corpus(VectorSource(df$TWEET))
corpus

inspect(corpus[1:3])#no of characters in message

print(corpus)

as.character(corpus[[1]]) #displays the message

#data clean up
#convert to lower case
corpus <- tm_map(corpus,content_transformer(tolower))
as.character(corpus[[1]])

# remove numbers
corpus <- tm_map(corpus, removeNumbers)
as.character(corpus[[1]])

# remove stop words
corpus <- tm_map(corpus, removeWords, stopwords())
as.character(corpus[[1]])

# remove punctuation
corpus <- tm_map(corpus, removePunctuation)
as.character(corpus[[10000]])

# words to root form
corpus <- tm_map(corpus, stemDocument)
as.character(corpus[[1]])

#replace special characters with white space
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
corpus <- tm_map(corpus, toSpace, "/")
corpus <- tm_map(corpus, toSpace, "@")
corpus <- tm_map(corpus, toSpace, "\\|")
```

```
# remove white space
corpus.clean <- tm_map(corpus, stripWhitespace)
as.character(corpus.clean[[1]])

#dtm
dtm <- DocumentTermMatrix(corpus.clean)
inspect(dtm[6000:6005, 10:15])

# word cloud
wordcloud(corpus.clean, min.freq = 10, max.words=200, random.order=FALSE,
rot.per=0.35,
        colors=brewer.pal(8, "Dark2"))
non_hate_mssg <- subset(corpus.clean, df$TYPE == 'non_hate_mssg' )
hate_mssg <- subset(corpus.clean, df$TYPE == 'hate_mssg')
#neutral <- subset(corpus.clean, df$TYPE==1)
wordcloud(non_hate_mssg, max.words = 40, scale = c(3, 0.1))
wordcloud(hate_mssg, max.words = 40, scale = c(3, 0.1))
#wordcloud(neutral, max.words = 40, scale = c(3, 0.1))

# frequent terms in DTM
findFreqTerms(dtm, lowfreq = 4)

#partitioning data
df.train <- df[1:8000,]
df.test <- df[8001:9999,]

dtm.train <- dtm[1:8000,]
dtm.test <- dtm[8001:9999,]

corpus.clean.train <- corpus.clean[1:8000]
corpus.clean.test <- corpus.clean[8001:9999]

#feature selection
dim(dtm.train)
fivefreq <- findFreqTerms(dtm.train, 5)
length((fivefreq))
dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary =
fivefreq))

dim(dtm.train.nb)
dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary =
fivefreq))
dim(dtm.test.nb)

# naive bayers alogorithm
# Function to convert the word frequencies to yes (presence) and no (absence) labels
```

```
convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}

# Apply the convert_count function to get final training and testing DTMs
trainNB <- apply(dtm.train.nb, 2, convert_count)
testNB <- apply(dtm.test.nb, 2, convert_count)

# training naive bayers algorithm

# Train the classifier with Naive Bayes
classifier2 <- naiveBayes(trainNB, df.train$TYPE)

# test the predictions
pred2 <- predict(classifier2, newdata=testNB)

# confusion matrix
conf2.mat <- confusionMatrix(pred2, df.test$TYPE)
conf2.mat
# training naive bayers algorithm and optimizing with Laplace Transformation
# Train the classifier
classifier <- naiveBayes(trainNB, df.train$TYPE, laplace = 1)

# test the predictions
# Use the NB classifier we built to make predictions on the test set.
pred <- predict(classifier, newdata=testNB)

# Create a truth table by tabulating the predicted class labels with the actual class labels
table("Predictions"= pred,  "Actual" = df.test$TYPE )

# confusion matrix
conf.mat <- confusionMatrix(pred, df.test$TYPE)
conf.mat
```