iJournals: International Journal of Software & Hardware Research in Engineering
ISSN-2347-4890
Volume 7 Issue 10 October 2019

Machine Learning: Sentiment Analysis - a Review

Author: Satyanarayana Medicherla
Consultant, Canon U.S.A. Inc, 1, Canon Park, New York - 11747, email- satyams@hotmail.com

Abstract

Understanding the opinion or the intent of an author or speaker from author's written text or document is of utmost importance in the modern internet world. Most of the data generated on the internet is in unstructured textual format [1]. The automated process of understanding the intent of this unstructured data is called the Sentiment Analysis in the machine learning jargon. Sentiment analysis is also called opinion mining or text mining. The sentiment expressed in a document could be positive, negative or neutral. Some of the important applications of sentiment analysis are analyzing the facebook posts, twitter tweets, amazon product reviews etc.

Introduction

The process of extracting or understanding the intent or the opinion of the author from the written text or document is called the sentiment analysis. Sentiment Analysis helps us structure this unstructured textual format of the text and extract the sentiment. There are different types of sentiment analysis based on how the sentiment analysis process is implemented. When it is implemented based on a pre-defined set of rules, it is called unsupervised type of sentiment analysis, where there is no model created from the data. The other method is called the supervised method of sentiment analysis, where a model is crated from already classified data. Then the model is used to predict the sentiment of unknown data or text. We will study these methods in detail in the following sections.

Supervised Method of Sentiment Analysis

In Supervised Method of Sentiment Analysis, we train our sentiment analysis model using a set of labeled training examples and then we use that model to predict the sentiment of unlabeled text or document. This method uses the Naive Bayes' Theorem, which is defined for a set of independent features. The bayes' theorem is defined as follows:

Bayes' theorem is based on conditional probabilities. The Bayes' theorem is stated as follows: [3]

$$Probability(A|B) = \frac{Probability(B|A) * Probability(A)}{Probability(B)}$$

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Where

- P(A|B) is the probability of A given the event B happened
- P(B|A) is the probability of B given the event A happened
- P(A) is the prior probability of A
- P(B) is the probability of B

The Naive Bayes' Theorem is defined as follows. [4]

Let us say that our feature (words in Sentiment Analysis) vector is defined as $W = (w_1, w_2, w_3, \ldots\ldots, w_n)$ and our classes are $C = (c_1, c_2, \ldots, c_k)$

When the features are independent of each other, we can write it as

$$P(C|w_1, w_2, w_3, \ldots . w_n) = \frac{P(w_1|C) * P(w_2|C) * \ldots * P(w_n|C) * P(C)}{P(w_1) * P(w_2) * \ldots * P(w_n)}$$

$$= \frac{P(C) * \prod_{i=1}^{n} P(w_i|C)}{\prod_{i=1}^{n} P(w_i)}$$

We will use the Naive Bayes' Theorem in Supervised Method of Sentiment Analysis. We will discuss this further in detail later in this article with practical examples.

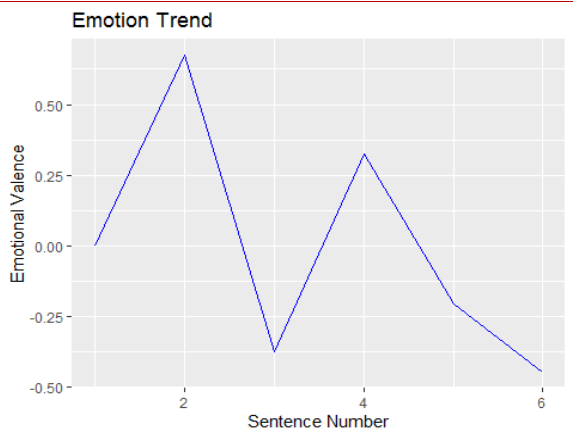
Unsupervised Sentiment Analysis

In unsupervised method of Sentiment Analys, we use a pre-defined set of rules or a Lexicon to determine the sentiment of a text or a document.

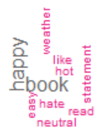
Let us take a simple example and discuss unsupervised method of sentiment analysis. We will see how to extract sentence level sentilemts, plot the sentiment levels, display the wordcloud for most frequently used words. We will discuss these unsupervised sentiment analys srelated concepts in detail in later sections of this article..

```
library(syuzhet)
library(sentimentr)

##
## Attaching package: 'sentimentr'

## The following object is masked from
'package:syuzhet':
##
##     get_sentences

library(ggplot2)
library(tm)

## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from
'package:ggplot2':
##
##     annotate

library(wordcloud)

## Loading required package: RColorBrewer

docs_demo <-     'This is a neutral statement.
              The am very happy.
              I am not happy.
              The book is easy to read.
              I do not like this book.
              I hate the hot weather'
```

```
# to get the individual sentence level sentiment

sentiment(docs_demo)

##    element_id sentence_id word_count  sentiment
## 1:          1           1          5  0.0000000
## 2:          1           2          4  0.6750000
## 3:          1           3          4 -0.3750000
## 4:          1           4          6  0.3265986
## 5:          1           5          6 -0.2041241
## 6:          1           6          5 -0.4472136

# to get the combined sentiment at the document level

sentiment_by(docs_demo)

##    element_id word_count       sd ave_sentiment
## 1:          1         30 0.434622  -0.004241556

sentences <-get_sentences(docs_demo)

sentiment_by(docs_demo)

##    element_id word_count       sd ave_sentiment
## 1:          1         30 0.434622  -0.004241556

extract_sentiment_terms(docs_demo)

##    element_id sentence_id negative positive
## 1:          1           1
## 2:          1           2            happy
## 3:          1           3            happy
## 4:          1           4             easy
## 5:          1           5             like
## 6:          1           6 hate,hot

# let us plot the sentiment level by sentence

df <-sentiment(docs_demo)

df_plot <-
data.frame(as.numeric(rownames(df)),df$sentiment)
colnames(df_plot) <-
c("sentence_number","sentiment_level")


ggplot(data = df_plot  , mapping =aes(x =
df_plot$sentence_number,
y=df_plot$sentiment_level),col=I("blue")) +
xlab("Sentence Number") +ylab("Emotional
Valence")  +ggtitle("Emotion Trend") +
geom_line(color='blue')
```



```
# wordcloud


docs <-VCorpus(VectorSource(docs_demo))

# Convert the text to lower case
docs <-tm_map(docs, content_transformer(tolower))

# Remove numbers
docs <-tm_map(docs, removeNumbers)

# Remove stop words
docs <-tm_map(docs, removeWords,
stopwords("english"))

# remove all punctuations
docs <-tm_map(docs, removePunctuation)

# remove white spaces created by above substitution
docs <-tm_map(docs, stripWhitespace)

dtm <-TermDocumentMatrix(docs)

m <-as.matrix(dtm)
v <-sort(rowSums(m),decreasing=TRUE)
d <-data.frame(word =names(v),freq=v)

# list all the words that are used at least 10 time in the
speech

wordcloud(words = d$word, freq = d$freq, min.freq
=1,scale=c(1,.2),
max.words=200, random.order=FALSE, rot.per=0.35,
colors=brewer.pal(8, "Dark2"))
```



Supervised method of Sentiment Analysis

This method of Sentiment Analysis uses some pre-labeled data for training the model and then the model is used to predict the sentiment on a document that is not labeled. Whilte testing the trained model, we will use a pre-labaled test data to evaluate our model. The model then is used to classified unlabelled test sets. We will discuss this furhter in the Naive Bayes' classifier.

Bayes' Theorem [3]

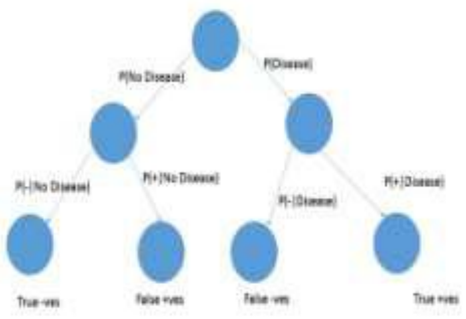
Bayes' theorem is based on conditional probabilities. The Bayes' theorem is stated as follows:

$$Probability(A|B) = \frac{Probability(B|A) * Probability(A)}{Probability(B)}$$

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

where

- P(A|B) is the probability of A given the event B happened
- P(B|A) is the probability of B given the event A happened
- P(A) is the prior probability of A
- P(B) is the probability of B



Bayes' Theorem Pictorial View [2]

From the above picture, we can clearly see that when someone is tested positive for the Disease, the probability of patient having the Disease is given by the following expression.

$$P(D|+) = \frac{P(+|D) * P(D)}{P(+|D) * P(D) + P(+|No\ D) * P(No\ D)}$$

Let us take the most popular example of breast cancer test. [10]

Here is the problem description:

- Let us say that there are 1% of the woman have breast cancer in the population
- Let us say 80 percent of the mammograms detect breast cance when the patient has breast cancer. So it misses 20% of the cases.
- Let us say 9 percent of the mammograms detect breast cancer when it is not present. So 91% of the cases it detects corretly.

Let us put this in terms of Bayes' theorem

- P(A|B) is the probability of having cancer when the test comes positive
- P(B|A) is the probability of testing positive when the the patient has cancer
- P(A) is the prior probability of cancer in the population
- P(B) is the probability of testing positive - this includes TRUE positives and FALSE positives

Let us say the doctor told that patient that the mammogram detected breast cancer. What are the patient's chances of having breast cancer? It is at 80 percent as our guess from the above explanation.

The result could be true positive or false positive.

Now let us compute true positive cases

- Only 1% have breast cancer and the mammogram detects 80 percent of that 1% correctly. So it is 80% of 1%, it is 0.008 - these are true positive cases.

Now let us compute the false positive cases

- 99% do not have breast cancer but the mammogram detect 9 percent of them as positive. So it is 9% of 99% which is 0.891.

If we compare those two figures, the false positives are far more than the true positives. This is because the disease exists only in 1% of the cases and true positives are computed over that 1%. The disease does not exist in 99 percent of the cases and false positives are computed on that 99%. That is why the false positives are far more than the true positives.

Bayes' Theorem is stated as follows:

$$P(disease|tested + ve) = \frac{P(testing + ve|disease) * P(disease)}{P(testing + ve)}$$

$$= \frac{P(testing + ve|disease) * P(isease)}{P(testing + ve|dieseae) * P(disease) + P(testing + ve|no\ dies}$$

$$P(D|+) = \frac{P(+|D) * P(D)}{P(+)}$$

$$= \frac{P(+|D) * P(D)}{(P(+|D) * P(D)) + (P(+|no\ D) * P(no\ D)}$$

$$= \frac{0.8 * 0.01}{(0.8 * 0.01) + (0.09 * 0.99)}$$

$$= 0.08238929$$

$$= about\ 8\%$$

Naive Bayes' Theorem for Sentiment Analysis [4]

Naive Bayes' theorem is a classification technique based on Bayes' theorem for a set of independent features. The Naive Bayes' theorem assumes that the freatures are independent. That means, the presence of a feature is not affected by the presence or absence of any other freature. When we are talking about sentiment analysis, the features are words within a document.

Let us say that our feature (words in Sentiment Analysis) vector is defined as

$W = (w_1, w_2, w_3, \ldots\ldots, w_n)$ and our classes are
$C = (c_1, c_2, \ldots, c_k)$

We are trying to classify the documents into one of the above k classes based on n-feature or word occurrences. For example, if we are working on email spam filter the classes would be just spam or non-spam (ham) and the features or the individual word occurrence within the emails.

For our set of features, the Naive Bayes' Theorem is stated as follows:

$$P(C|w_1, w_2, w_3, \ldots. w_n) = \frac{P(w_1, w_2, w_3, \ldots. w_n|C) * P(C)}{P(w_1, w_2, w_3, \ldots. w_n)}$$

When the features are independent of each other, we can write it as

$$P(C|w_1, w_2, w_3, \ldots. w_n) = \frac{P(w_1|C) * P(w_2|C) * P(w_3|C) * \ldots * P(w_n|C) * P(C)}{P(w_1) * P(w_2) * P(w_3) * \ldots * P(w_n)}$$

$$= \frac{P(C) * \prod_{i=1}^{n} P(w_i|C)}{\prod_{i=1}^{n} P(w_i)}$$

Since the denominator is common for all the class probabilities that we are computing, we can as well ignore it.

$$\propto P(C) * \prod_{i=1}^{n} P(w_i|C)$$

If there are k different classes, we will determine the above probability for each of those classes and take the class with highest probability as the target class. Our word occurrences within a document would determine the class of that document. When some of the words from our word vector do not appear in our document, our joint probability, which is a product of the individual probabilities, would become zero. We do a Laplace smoothing in order to accommodate this zero probability issue.

Laplace Smoothing [5]

If any word does not occur in any class, the probability would be zero and the whole product would become zero. We would apply Laplace Smoothing for such words. The expression for Laplace Smoothing is give as follow:

$$\theta_i = \frac{x_i + 1}{N + \alpha * d}$$

where, θ_i is the probability after Laplace Smoothing and x_i is the frequency of the i^{th} word in that specific class. α is the smoothing parameter, N is the number of unique words in the class and d is the number of unique words in the entire training set of documents.

Naive Bayes' General Example

This is a general example of Naive Bayes' theorem. [6]

```
docs <-matrix(c('Fruit','yellow','sweet','long','total',
'Mango','350','450','0','650',
'Banana','400','300','350','400',
'Others','50','100','50','150',
'Total','800','850','400','1200'
            ),ncol=5,byrow=TRUE)


docs

##      [,1]     [,2]     [,3]    [,4]   [,5]
## [1,] "Fruit"  "yellow" "sweet" "long" "total"
## [2,] "Mango"  "350"    "450"   "0"    "650"
## [3,] "Banana" "400"    "300"   "350"  "400"
## [4,] "Others" "50"     "100"   "50"   "150"
## [5,] "Total"  "800"    "850"   "400"  "1200"
```

So out of 650 mangoes, 350 are yellow, 450 are sweet and 0 are long

There are 800 yellow, 850 sweet and 400 long fruits

So P(mango|yellow) = 350/800 P(banana/yellow) = 400/800

Now let us say we are given a fruit that is yellow, sweet and long, now we want to classify it.

What is the probability of it being, Mango

$$P(Mango|(yello, sweet, long)) = \frac{P(yellow|mango) * P(sweet|mango) * P(long|mango) * P(m}{P(yello, sweet, long)}$$

= 0 because, a mango being long is zero

$$P(banana|(yello, sweet, long)) = \frac{P(yellow|banana) * P(sweet|banana) * P(long|banana) * P(}{P(yello, sweet, long)}$$

$$= \frac{\frac{400}{400} * \frac{300}{400} * \frac{350}{400} * \frac{400}{1200}}{P(yello, sweet, long)}$$

$$= \frac{0.21875}{P(yello, sweet, long)}$$

$$P(others|(yello, sweet, long)) = \frac{P(yellow|others) * P(sweet|others) * P(long|others) * P(others)}{P(yello, sweet, long)}$$

$$= \frac{\frac{50}{150} * \frac{100}{150} * \frac{50}{150} * \frac{150}{1200}}{P(yello, sweet, long)}$$

$$= \frac{0.00926}{P(yello, sweet, long)}$$

We classify it as bana a, since the probability of it being a banana is the maximum,

Naive Bayes' Sentiment Analysis Example [12]

We have 5 different documents in our training set. Three of them are Sports and the other two are non-sports. Now we will train our model on these 5 different documents and then we will try to classify a given document as one of those classes i.e. Sport or Not sports.

Now let us try to classify the sentence "A very close game" into one of those classes. The process here is to check this sentence if this exists as it is in one of those classes, we can take that class as the target class. But this sentence does not exist in our documents. Now we will use Naive Bayes' theorm where we assume that these words are completely independent of each other and hence we can multipy the resulting probabilities to obtain the joint probability.

Now the joint probability is given by the following

For Sports

$$P(Sports|A\ very\ close\ game) = \frac{(P(A\ very\ close\ game|Sports) * P(Sports)}{P(A\ very\ close\ game)}$$

For Not Sports

$$P(Non\ Sports|A\ very\ close\ game) = \frac{(P(A\ very\ close\ game|Non\ Sports) * P(Non\ Sports)}{P(A\ very\ close\ game)}$$

The denominator is the same for both the above and hence we can ingore the denominator and compare the numerators.

Now

$$P(Sports|A\ very\ close\ game) = (P(A\ very\ close\ game|Sports) * P(Sports)$$

$$= P(A|Sports) * P(very|Sports) * P(close|Sports) * P(game|Sports) * P(Sports)$$

and

$$P(Non\ Sports|A\ very\ close\ game) = (P(A\ very\ close\ very\ game|Non\ Sports) * P(Non\ Sports)$$

$$= P(A|Non\ Sports) * P(very|Non\ Sports) * P(close|Non\ Sports) * P(game|Non\ Sports) * P(Non\ Sports)$$

Now let us compute the individual probabilites:

The prior-probability of 'Sports' is 3/5 and for 'Not Sports' it is 3/5 because 3 out of 5 documents are Sports and the rest are Not Sports.

- P(A|Sports) = The total number of times 'A' appears in sports by the total number of words in Sports = $\frac{2}{11}$
- P(close|Sports) = The total number of times 'close' appears in sports by the total number of words in Sports = $\frac{0}{11}$
- P(close|Sports) = The total number of times 'close' appears in sports by the total number of words in Sports = $\frac{2}{11}$
- P(game|Sports) = The total number of times 'game' appears in sports by the total number of words in Sports = $\frac{2}{11}$

P(close|Sports) makes the whole product zero. We will apply Laplace smoothing to workaround this issue.

$$\theta_i = \frac{x_i + 1}{N + \alpha * d}$$

where, θ_i is the probability after Laplace Smoothing. x_i is the frequency of the i^{th} word in that specific class. α is the smoothing parameter, N is the number of unique words in the that class and d is the number of unique words in the entire training set of documents.

Now the probabilities for Sports would become:

- P(A|Sports) = The total number of times 'A' appears in sports by the total number of words in Sports = $\frac{2+1}{11+14}$
- P(close|Sports) = The total number of times 'close' appears in sports by the total number of

words in Sports = $\frac{0+1}{11+14}$

- P(close|Sports) = The total number of times ‘close’ appears in sports by the total number of words in Sports = $\frac{2+1}{11+14}$
- P(game|Sports) = The total number of times ‘game’ appears in sports by the total number of words in Sports = $\frac{2+1}{11+14}$

Now the probabilities for Not Sports would become:

- P(A|Not Sports) = $\frac{1+1}{11+14}$
- P(close|Not Sports) = $\frac{0+1}{11+14}$
- P(close|Not Sports) = $\frac{1+1}{11+14}$
- P(game|Not Sports) = $\frac{0+1}{11+14}$

So we classify our new document as Sports.

Naive Bayes’ Sentiment Analysis Example [5]

```
library(tm)
library(e1071)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from
'package:stats':
##
##     filter, lag

## The following objects are masked from
'package:base':
##
##     intersect, setdiff, setequal, union

library(caret)

## Loading required package: lattice

docs <-matrix(c('A great game','Sports',
'The election was over','Not Sports',
'Very clean match','Sports',
'A clean but forgettable game','Sports',
'It was a close election','Not
Sports'),ncol=2,byrow=TRUE)

df_train <-data.frame(docs)

colnames(df_train) <-c("text","class")

df_train$class <-as.factor(df_train$class)

corpus_train <-
VCorpus(VectorSource(df_train$text))

dtm <-DocumentTermMatrix(corpus_train)

Naive_Bayes_Model=naiveBayes(as.matrix(dtm),df_
train$class,laplace=1)

docs_test <-matrix(c(
'A great game','Sports',
'close election','Not Sports',
'clean match','Sports',
'A great game','Sports',
'close election','Not Sports',
'clean match','Sports'
            ),ncol=2,byrow=TRUE)


df_test <-data.frame(docs_test)
colnames(df_test) <-c("text","class")

df_test$class <-as.factor(df_test$class)

corpus_test <-VCorpus(VectorSource(df_test$text))

dtm_test <-DocumentTermMatrix(corpus_test)

pred <-
predict(Naive_Bayes_Model,as.matrix(dtm_test))

## Warning in
predict.naiveBayes(Naive_Bayes_Model,
as.matrix(dtm_test)):
## Type mismatch between training and new data for
variable 'but'. Did you use
## factors with numeric labels for training, and
numeric values for new data?

## Warning in
predict.naiveBayes(Naive_Bayes_Model,
as.matrix(dtm_test)): Type
## mismatch between training and new data for
variable 'forgettable'. Did you
## use factors with numeric labels for training, and
numeric values for new
## data?

## Warning in
predict.naiveBayes(Naive_Bayes_Model,
as.matrix(dtm_test)): Type
## mismatch between training and new data for
variable 'over'. Did you use
## factors with numeric labels for training, and
```

```
numeric values for new data?

## Warning in
predict.naiveBayes(Naive_Bayes_Model,
as.matrix(dtm_test)):
## Type mismatch between training and new data for
variable 'the'. Did you use
## factors with numeric labels for training, and
numeric values for new data?

## Warning in
predict.naiveBayes(Naive_Bayes_Model,
as.matrix(dtm_test)): Type
## mismatch between training and new data for
variable 'very'. Did you use
## factors with numeric labels for training, and
numeric values for new data?

## Warning in
predict.naiveBayes(Naive_Bayes_Model,
as.matrix(dtm_test)):
## Type mismatch between training and new data for
variable 'was'. Did you use
## factors with numeric labels for training, and
numeric values for new data?

table("Predictions"=pred,  "Actual" =df_test$class)

##            Actual
## Predictions  Not Sports Sports
##   Not Sports          2      0
##   Sports              0      4
```

Sentiment Analysis using Naive Bayes' Theorem

Airline Tweets Sentiment Analysis [9]

This dataset contains Airline reviews along with their associated sentiment polarity labels Positive, Negative and Neutral. We will use this pre-labeled data set to train our model and then use it on our test set to determine the sentiment. The column 'text' is the actual tweet by the customer and airline_sentiment column is the pre-labeled sentiment. We will use only these two columns for our analysis.

```
# Load required libraries

library(tm)
library(e1071)
library(dplyr)
library(caret)

setwd("C:/Coursera/R/MachineLearning")
tweets <-read.csv("AirlineTweets.csv")

# use only those tweets with confidence level of 1

tweets_good <-
tweets[tweets$airline_sentiment_confidence ==1,]

df <-
data.frame(tweets_good$text,tweets_good$airline_se
ntiment)

colnames(df) <-c("text","sentiment")

#To removing hashtag , urls and other special
characters from the tweets

df$text <-gsub("http.*","",df$text)
df$text <-gsub("https.*","",df$text)
df$text <-gsub("#.*","",df$text)
df$text <-gsub("@\\w+", "", df$text)

set.seed(111)
df <-df[sample(nrow(df)), ]

df$sentiment <-as.factor(df$sentiment)

corpus <-VCorpus(VectorSource(df$text))

corpus_tweets_clean <-corpus %>%
tm_map(content_transformer(tolower)) %>%
tm_map(removePunctuation) %>%
tm_map(removeNumbers) %>%
tm_map(removeWords, stopwords(kind="en"))
%>%
tm_map(stripWhitespace)

#  We will geneate the Document Term Matrix for our
tweets - this DTM is a matrix containing the
words/terms used by each tweet

dtm <-DocumentTermMatrix(corpus_tweets_clean)

# use 7000 tweets for training and 500 for testing

df_tweets_train <-df[1:7000,]
df_tweets_test <-df[7001:7500,]

corpus_tweets_clean_train <-
corpus_tweets_clean[1:7000]
corpus_tweets_clean_test <-
corpus_tweets_clean[7001:7500]

# Let us ignore the words that were used in less than
```

```
five tweets

word_list <-findFreqTerms(dtm, 5)

dtm_train_less_words <-
DocumentTermMatrix(corpus_tweets_clean_train,
control=list(dictionary = word_list))

dim(dtm_train_less_words)

## [1] 7000 2099

dtm_test_less_words <-
DocumentTermMatrix(corpus_tweets_clean_test,
control=list(dictionary = word_list))

dim(dtm_test_less_words)

## [1]  500 2099

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


dtm_train_new <-apply(dtm_train_less_words, 2,
transform_count)
dtm_test_new <-apply(dtm_test_less_words, 2,
transform_count)

length(dtm_train_new)

## [1] 14693000

trained_model <-naiveBayes(dtm_train_new,
df_tweets_train$sentiment, laplace =1)

pred <-predict(trained_model,
newdata=dtm_test_new)

table("Predictions"=pred,  "Actual"
=df_tweets_test$sentiment)

##           Actual
## Predictions negative neutral positive
##    negative      326      20       11
##    neutral        24      51        1
##    positive       14       5       48
```

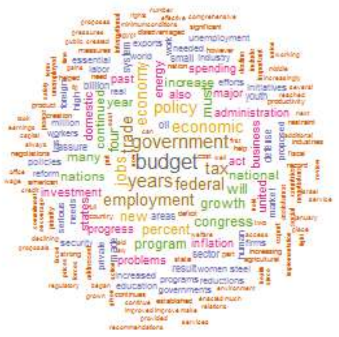
Unsupervised Sentiment Analysis

Word Clouds [7]

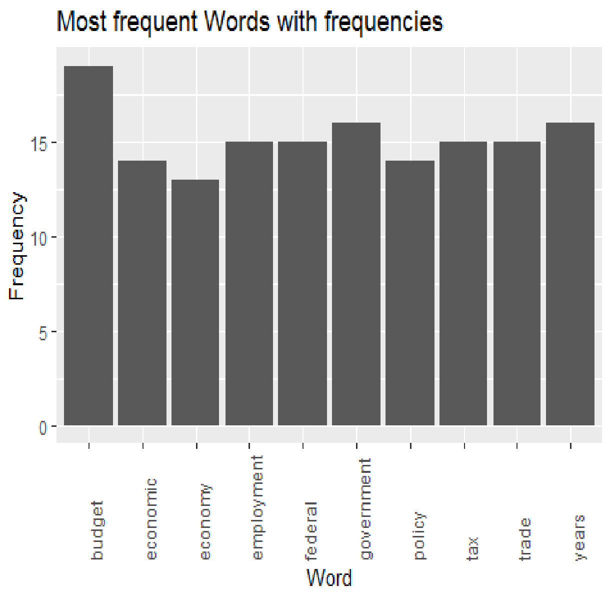
Wordcloud is a very simple and useful tool, which helps us to display the words that are most frequently used words in a document. The most frequently used words can be listed and font size of the word in the wordcloud is proportional to the number of occurrences of that word in the document. So the wordsclouds are easy to generate and easy to comprehend.

Let us generate a wordcloud for the Former President Mr. Jimmy Carter's Democratic Party's nomination acceptance speech- 1976. [8]

```
#https://www.jimmycarterlibrary.gov/assets/documents
/speeches/su81jec.phtml

library(tm)
library(wordcloud)
library(ggplot2)

# Carter's State of the Union 1981

setwd("C:/Coursera/R/MachineLearning")
speech=readLines("Carter-state-of-the-union-
1981.txt")

class(speech)

## [1] "character"

docs <-VCorpus(VectorSource(speech))


blank <-content_transformer(function (x , pattern )
gsub(pattern, " ", x))
docs <-tm_map(docs, blank, "/")
docs <-tm_map(docs, blank, "@")
docs <-tm_map(docs, blank, "\\|")


# Convert the text to lower case
docs <-tm_map(docs, content_transformer(tolower))


# Remove numbers
docs <-tm_map(docs, removeNumbers)


# Remove stop words
docs <-tm_map(docs, removeWords,
stopwords("english"))

# remove all punctuations
docs <-tm_map(docs, removePunctuation)

# remove white spaces created by above substitution
docs <-tm_map(docs, stripWhitespace)


dtm <-TermDocumentMatrix(docs)
inspect(dtm[2,])

## <<TermDocumentMatrix (terms: 1, documents: 150)>>
## Non-/sparse entries: 2/148
## Sparsity           : 99%
## Maximal term length: 4
## Weighting          : term frequency (tf)
## Sample             :
##       Docs
## Terms  1 126 132 2 3 4 5 6 7 8
##   able 0   1   1 0 0 0 0 0 0 0

m <-as.matrix(dtm)
v <-sort(rowSums(m),decreasing=TRUE)
d <-data.frame(word =names(v),freq=v)
head(d, 10)

##                  word freq
## budget         budget   19
## government government   16
## years           years   16
## employment employment   15
## federal       federal   15
## tax               tax   15
## trade           trade   15
## economic     economic   14
## policy         policy   14
## economy       economy   13

# list all the words that are used at least 10 time in the speech
wordcloud(words = d$word, freq = d$freq, min.freq =1,scale=c(1,.2),
max.words=200, random.order=FALSE, rot.per=0.35, colors=brewer.pal(8, "Dark2"))
```

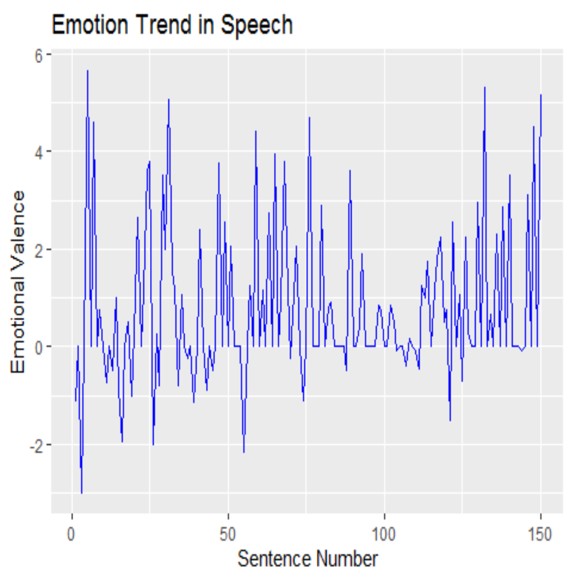


```
# Barplot of word frequencies

p<-ggplot(data=d[1:10,], aes(x=d[1:10,]$word, y=d[1:10,]$freq)) +
xlab("Word") +ylab("Frequency") +
ggtitle("Most frequent Words with frequencies")+
geom_bar(stat="identity")+theme(axis.text.x =element_text(angle =90, hjust=0))

p
```



```
sentence_sentiment <-get_sentiment(speech)

df <-data.frame(sentence_sentiment)
df$sentence_number <-as.numeric(rownames(df))
```

```
str(df)

## 'data.frame':    150 obs. of  2 variables:
##  $ sentence_sentiment: num  -1.15 0 -3 0 5.65 0 4.6
0 0.75 0 ...
##  $ sentence_number   : num  1 2 3 4 5 6 7 8 9 10 ...

ggplot(data = df  , mapping =aes(x =
df$sentence_number,
y=df$sentence_sentiment),col=I("blue")) +
xlab("Sentence Number") +ylab("Emotional
Valence")  +ggtitle("Emotion Trend in Speech") +
geom_line(color='blue')
```



The above wordcloud shows the some of the most frequently used words in the 1981 State of the Union Address. The font size is proportional to the number of occurrences of that word in the speech.

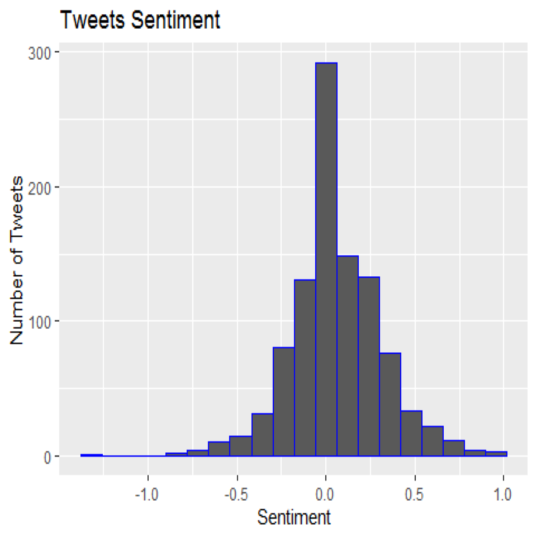
Unsupervised Sentiment Analysis - Twitter Airline Sentiment Data at [9] [11]

Let us use the same Airline reviews data set for the unsupervised method of sentiment analysis also.

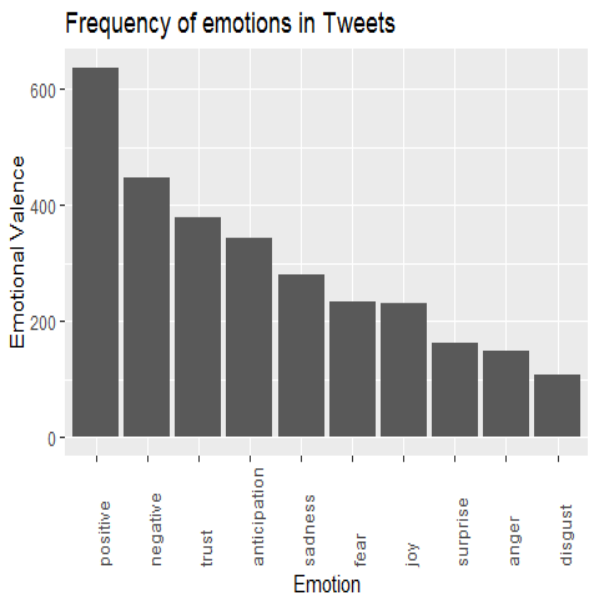
```
# Load Requried Packages
library(SnowballC) # stemming
library(tm)  # text mining
library(twitteR)

##
## Attaching package: 'twitteR'

## The following objects are masked from
'package:dplyr':
##
##     id, location
```

```
library(syuzhet)
library(sentimentr)
library(wordcloud)  # word cloud
library(ggplot2)

tweets <-read.csv("AirlineTweets.csv")
dim(tweets)

## [1] 14640    15

#################################################
######  remove the following statement
##########################################
tweets <-tweets[1:1000,]

head(tweets$text)

## [1] @VirginAmerica What @dhepburn said.
## [2] @VirginAmerica plus you've added
commercials to the experience... tacky.
## [3] @VirginAmerica I didn't today... Must mean I
need to take another trip!
## [4] @VirginAmerica it's really aggressive to blast
obnoxious "entertainment" in your guests' faces &amp;
they have little recourse
## [5] @VirginAmerica and it's a really big bad thing
about it
## [6] @VirginAmerica seriously would pay $30 a
flight for seats that didn't have this playing.\nit's really
the only bad thing about flying VA
## 14427 Levels: "LOL you guys are so on it" - me,
had this been 4 months ago...â\200œ@JetBlue: Our
fleet's on fleek. http://t.co/LYcARlTFHlâ\200\235 ...

#Removing hashtag , urls and other special characters

tweets_clean <-gsub("http.*","",tweets$text)
tweets_clean <-gsub("https.*","",tweets_clean)
tweets_clean <-gsub("#.*","",tweets_clean)
tweets_clean <-gsub("@\\w+", "", tweets_clean)
length(tweets_clean)

## [1] 1000

tweets_sentiment <-sentiment_by(tweets_clean)
head(tweets_sentiment)

##    element_id word_count        sd ave_sentiment
## 1:          1          2        NA     0.0000000
## 2:          2          8 0.1767767    -0.1364216
## 3:          3         11 0.0000000     0.0000000
## 4:          4         16        NA    -0.6075000
## 5:          5          9        NA    -0.3000000
## 6:          6         21 0.1353203    -0.1476474
```

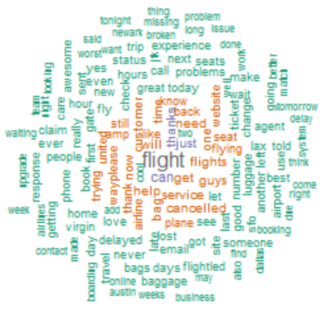
```
dim(tweets_sentiment)

## [1] 1000    4

mean(tweets_sentiment$ave_sentiment)

## [1] 0.05458895

p <-qplot(tweets_sentiment$ave_sentiment,
geom="histogram",bins=20,col=I("blue"),xlab="Senti
ment",ylab="Number of Tweets", main="Tweets
Sentiment")
p
```



```
# now let us work on the emotional content of the
tweets using NRC sentiment lexicon

# Emotions for each tweet using NRC Lexican
emotions_df <-get_nrc_sentiment(tweets_clean)

# Let us sum by emotion

emotions_sum =colSums(emotions_df)

emotions_sum_df =data.frame(count=emotions_sum,
emotion=names(emotions_sum))
emotions_sum_df$emotion
=factor(emotions_sum_df$emotion,
levels=emotions_sum_df$emotion[order(emotions_su
m_df$count, decreasing =TRUE)])

# Basic barplot
p<-ggplot(data=emotions_sum_df,
aes(x=emotions_sum_df$emotion,
y=emotions_sum_df$count)) +
xlab("Emotion") +ylab("Emotional Valence") +
ggtitle("Frequency of emotions in Tweets")+
geom_bar(stat="identity")+theme(axis.text.x
=element_text(angle =90, hjust=0))

p
```



```
# word cloud to check the most frequently used words

docs <-VCorpus(VectorSource(tweets_clean))

docs <-tm_map(docs, content_transformer(tolower))

# Remove numbers
docs <-tm_map(docs, removeNumbers)

# Remove stop words
docs <-tm_map(docs, removeWords,
stopwords("english"))

# remove all punctuations
docs <-tm_map(docs, removePunctuation)

# remove white spaces created by above substitution
docs <-tm_map(docs, stripWhitespace)

tdm <-TermDocumentMatrix(docs)
inspect(tdm[2,])

## <<TermDocumentMatrix (terms: 1, documents:
1000)>>
## Non-/sparse entries: 1/999
## Sparsity           : 100%
## Maximal term length: 3
## Weighting          : term frequency (tf)
## Sample             :
##      Docs
## Terms 1 2 3 4 401 5 6 7 8 9
##   â\200¦ 0 0 0 0   1 0 0 0 0 0

tdm_matrix <-as.matrix(tdm)
word_sorted <-
sort(rowSums(tdm_matrix),decreasing=TRUE)
word_sorted_df <-data.frame(word
=names(word_sorted),freq=word_sorted)
head(word_sorted_df, 10)

##              word freq
## flight     flight  212
## can           can   71
## just         just   65
## thanks     thanks   58
## get           get   53
## will         will   53
## help         help   49
## service   service   47
## flights   flights   46
## customer customer   43

# list all the words that are used at least 100 times in
the tweets

wordcloud(words = word_sorted_df$word, freq =
word_sorted_df$freq, min.freq =10,scale=c(1,.5),
max.words=200, random.order=FALSE, rot.per=0.35,
colors=brewer.pal(8, "Dark2"))
```



References

1. https://link.springer.com/chapter/10.1007/978-981-10-3433-6_21
2. https://www.youtube.com/watch?v=j2tNxIaGpR4
3. https://en.wikipedia.org/wiki/Bayes%27_theorem
4. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c
5. https://medium.com/syncedreview/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation-4f5271768ebf
6. https://www.hackerearth.com/blog/developers/introduction-naive-bayes-algorithm-codes-python-r/
7. http://www.sthda.com/english/wiki/word-cloud-generator-in-r-one-killer-function-to-do-everything-you-need
8. https://www.jimmycarterlibrary.gov/assets/documents/speeches/su81jec.phtml
9. https://blog.cambridgespark.com/50-free-machine-learning-datasets-sentiment-analysis-b9388f79c124
10. https://betterexplained.com/articles/an-intuitive-and-short-explanation-of-bayes-theorem/
11. https://cran.r-project.org/web/packages/syuzhet/vignettes/syuzhet-vignette.html
12. https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/