**Sentiment Analysis**

**INTRODUCTION**

What is **Sentimental Analysis**?

It is one of the Natural Language Processing fields. It is contextual mining of text which identifies and extracts subjective information in source material. It aims to extract people’s opinions, sentiments, and subjectivity from the texts. It analyses an incoming message and tells whether the underlying sentiment is positive, negative our neutral.

## Sentiment Analysis Algorithms

### Rule-based approach

Rule-based sentiment analysis is based on an algorithm with a clearly defined description of an opinion to identify. Includes identify subjectivity, polarity, or the subject of opinion. The rule-based approach involves basic [Natural Language Processing](https://theappsolutions.com/blog/development/natural-language-processing/) routine. It involves the following operations with the text corpus:

* Stemming
* Tokenization
* Parsing
* Lexicon analysis

Here’s how it works:

* There are three lists of words. One of them includes valence for the words, the other includes the words which negate the sentence, and the third contains booster words.
* The algorithm goes through the text, finds the words that match the criteria.
* After that, the algorithm calculates which type of words is more prevalent in the text. If there are more positive words, then the text is deemed to have a positive polarity.

### Automatic Sentiment Analysis

This type of sentiment analysis uses machine learning to figure out the gist of the message. Because of that, the precision and accuracy of the operation drastically increase and you can process information on numerous criteria without getting too complicated. In essence, the automatic approach involves [supervised machine learning](https://theappsolutions.com/blog/development/supervised-machine-learning/) classification algorithms. In addition to that, [unsupervised machine learning](https://theappsolutions.com/blog/development/unsupervised-machine-learning/) algorithms are used to explore data.

Overall, Sentiment analysis may involve the following types of classification algorithms:

* Linear Regression
* Naive Bayes
* Support Vector Machines
* RNN derivatives LSTM and GRU

**OVERVIEW**

Data are imported using R and the data is cleaned by removing emoticons and URLs. Lexical Analysis is used to predict the sentiment of data.

**SYSTEM REQUIREMENTS**

* Installation of R

**FEATURES**

* **Cleaning Data**

The Data is cleaned by removing:

* URLS/Email Address
* Punctuations
* Emoticons
* Redundant Blank spaces
* Stop Words
* **Loading Word Database**

Stop words are most commonly used words in a language like the, is, at, which, and, on, etc.

Negative words are most commonly words in a language which can reverse the sentiment of the sentence.

Sentimental Values & Booster words are a list of English words rated for valence with a value. The file is tab-separated.

All of the above are collected from NLTK Library.

* **Algorithms Used**

Lexical Analysis: By comparing uni-grams to the pre-loaded word database collected from nltk library, the data is assigned sentiment score which is used to calculate positive, negative or neutral and overall score.

* **Calculation**

I have presented the scores, the data as well as the percentage of positive / neutral / negative emotion in the text. This calculated using simple arithmetic to understand the overall sentiment.

**LIMITATIONS**

Cannot get 100% efficiency in analysing sentiment of data.

**OUTPUT**

The default output is a single text that is one of these values:

Positive Sentiment

Negative Sentiment

Neutral Sentiment

Overall Sentiment

**REFERENCES**

https://www.nltk.org/\_modules/nltk/sentiment/vader.html

<https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>

https://www.nltk.org/nltk\_data/

<http://www.endmemo.com/r/gsub.php>

**R Code**

library(readr)

text=read\_file('Input\_Data.txt')

#text="I can definitely do it"

sentiments\_values=read.csv('Sentiment\_Values.csv',header=FALSE)

stop\_words=unlist(read.csv("Stop\_Words.csv",header=FALSE))

negate=unlist(read.csv("Negate.csv",header=FALSE))

boost=read.csv("Boost.csv",header=FALSE)

clean\_data=function(text)

{

text=tolower(text)

text=gsub("http\\S+|www\\S+|@\\S+","",text)

text=gsub("[[:punct:]]", "\\1",text)

text=gsub("\\s{2,}|[[:cntrl:]]"," ",text)

return (text)

}

tokenize\_text=function(text)

{

words=unlist(strsplit(text," "))

return (words)

}

remove\_stopword=function(words,stop\_words)

{

cleared\_words=c()

for(i in words)

if(length(i)>0& !i %in% stop\_words)

cleared\_words=c(cleared\_words,i)

return (cleared\_words)

}

scalar=function(word,valence,boost)

{

scalar=0

map=new.env(T,emptyenv())

for(i in 1:nrow(boost))

map[[boost[i,1]]]=boost[i,2]

if (word %in% boost[,1])

{

scalar=map[[word]]

if (valence<0)

scalar=scalar\*-1

}

return (scalar)

}

negated=function(words,negate)

{

for(i in words)

if(i %in% negate)

return (TRUE)

return (FALSE)

}

never\_check=function(valence,sentitext,i,j,negate)

{

if(negated(sentitext[i],negate)||negated(sentitext[i-j],negate))

valence=valence\*-0.74

return (valence)

}

sentiment\_val=function(valence,sentitext,item,i,sentiments,sentiments\_values,boost,negate)

{

map=new.env(T,emptyenv())

for(j in 1:nrow(sentiments\_values))

map[[sentiments\_values[j,1]]]=sentiments\_values[j,2]

if(item %in% sentiments\_values[,1])

{

valence=map[[item]]

for(j in 0:2)

{

if(i>j&&is.null(map[[sentitext[i-j]]]))

{

s=scalar(sentitext[i-j],valence,boost);

if(s!=0)

{

if(j==1)

s=s\*0.95

if(j==2)

s=s\*0.9

}

valence=valence+s

valence=never\_check(valence,sentitext,j,i,negate)

}

}

}

sentiments=c(sentiments,valence)

return (sentiments)

}

polarity\_scores=function(words,sentiments\_values,boost,negate)

{

sentiments=c()

for(i in 1:length(words))

{

if(words[i] %in% boost[,1])

{

sentiments=c(sentiments,0)

next

}

sentiments=sentiment\_val(0,words,words[i],i,sentiments,sentiments\_values,boost,negate)

}

compound=pos=neg=neu=0

if(length(sentiments)>0)

{

sumofsent=sum(sentiments)

compound=round(sumofsent/sqrt((sumofsent\*\*2)+15),2)

pos\_sum=neg\_sum=neu\_sum=0

for(i in sentiments)

{

if(i>0)

pos\_sum=pos\_sum+i+1

else if(i<0)

neg\_sum=neg\_sum+i-1

else

neu\_sum=neu\_sum+1

}

neg\_sum=abs(neg\_sum)

total=(pos\_sum+neg\_sum+neu\_sum)

pos=round(abs(pos\_sum/total),4)\*100

neu=round(abs(neu\_sum/total),4)\*100

neg=round(abs(neg\_sum/total),4)\*100

}

print(paste("Postive Sentiment:",pos,"%"))

print(paste("Neutral Sentiment:",neu,"%"))

print(paste("Negative Sentiment:",neg,"%"))

if(compound>=0.05)

print("Overall Sentiment: Positive")

else if(compound<=-0.05)

print("Overall Sentiment: Negative")

else

print("Overall Sentiment: Neutral")

return (list("pos"=pos,"neu"=neu,"neg"=neg,"compound"=compound))

}

cleaned\_text=clean\_data(text)

tokenized\_words=tokenize\_text(cleaned\_text)

cleared\_words=remove\_stopword(tokenized\_words,stop\_words)

result=polarity\_scores(cleared\_words,sentiments\_values,boost,negate)