References

[1] Telecommunication Regulatory Commission of Sri Lanka (TRCSL), “Financial Analysis of the Telecom Sector,”June, 2017.

[2] Ed. King, *Staying In Droves: How to Win the Customer Retention Revolution*, 2009.

[3] Bamfo, A. “Exploring the relationship between customer satisfaction and loyalty in the mobiletelecommunication industry in Ghana,”*Indian Journal of Economics and Business*, vol. 8, no. 20, pp. 299-311, 2009.

[4] Metzler, K., & Hinterhuber, H. “How to make product development projects more successful byintegrating Kano's model of customer satisfaction into quality function deployment,”*Technovation*, vol. 18, no. 1, pp.25-38, 1998.

[5]Judith Lamont, P. “Customer sentiment analysis: A shift to customer service,” *KMWorld Magazine*, vol. 22, no. 2, pp.8, 2013.

[6] Manasee Godsay. “The Process of Sentiment Analysis: A Study,”*International Journal of Computer Applications*, vol. 126, no. 7, pp. 26-30, 2015.

[7] L. Almuqren and A. I. Cristea, “Twitter analysis to predict the satisfaction of telecom company customers,” in *Late-breaking Results, Demos, Doctoral Consortium, Workshops Proceedings and Creative Track of the 27th ACM Conference on Hypertext and Social Media*, 2016.

[8] W. Kasper and M. Vela, “Sentiment analysis for hotel reviews,”in*Proceedings of the Computational Linguistics-Applications Conference*, 2011.

[9]C. Ramasubramanian and R. Ramya, “Effective Pre-Processing Activities in Text Miningusing Improved Porter’s Stemming Algorithm,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 12, 2013.

[10] P. Willett, “The Porter stemming algorithm: then and now,” Program: electronic library and information systems, vol. 40, no. 3, pp.219-223, 2006.

[11] G. Chakraborty, M. Pagou and S. Garla, *Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS*. Cary, North Carolina, USA, 2013.

[12] S. Kolkur, G. Dantal and R. Mahe, “Study of Different Levels for Sentiment Analysis,” *International Journal of Current Engineering and Technology*,vol. 5, no. 2, 2015.

[13] B. Liu and L. Zhang, “A Survey of Opinion Mining and Sentiment Analysis,”in*Proceedings of Mining Text Data*, 2013.

[14] P. Patil and P. Yalagi, “Sentiment Analysis Levels and Techniques: A Survey,” *International Journal of Innovations in Engineering and Technology*, vol. 6, no. 4, pp. 523-528, 2016.

[15]W. Medhat, A. Hassan and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093-1113, 2014.

[16] V. B. Vaghela and B. M. Jadav, “Analysis of Various Sentiment ClassificationTechniques,” *International Journal of Computer Applications*, vol. 140, no. 3, pp. 22-27, 2016.

[17] M. W. Berry, A. Mohammed and B.W. Yap, “Soft Computing in Data Science,”in*Proceedings of Soft Computing in Data Science,*Putrajaya, Malaysia, 2015.

[18] D. L. Olson andD. Delen, “Performance Evaluation for Predictive Modeling,” in *Advanced Data Mining Techniques*, Springer-Verlag Berlin Heidelberg, pp. 137-139.

[19]F. M. Kundi, A. Khan, S. Ahmad, M. Z. Asghar, “Lexicon-Based Sentiment Analysis in the Social,” *Journal of Basic and Applied Scientific Research*, vol. 4, no. 6, pp. 238-248, 2014.

[20] A. Esuli ,F. Sebastiani, “Senti Word Net: A Publicly Available Lexical Resource for Opinion Mining,” In*Proceedings of International Conference on Language Resources and Evaluation (LREC)*, 2006, pp. 417-422.

[21] S. M. Vohira and J. B. Teraiya, “A Comparative Study of Sentiment Analysis Techniques,” *Journal of Information, Knowledge and Research in Computer Engineering*, vol. 2, no. 2, pp. 313-317, 2013.

Appendixes

Appendix A – R code for using only Lexicon Dictionary

```
#Install & Load Required R packages
#install.packages("data.table")
library(data.table)
library(qdapRegex)
library(plyr)
library(stringr)
library(qdap)


#Import Twitter Feeds
x <- fread ("C:\\Users\\Sachi\\Desktop\\MscProject\\TwitterFeeds\\TestData01.txt",
header = TRUE, select = c("Text","Rating"))

#Remove Duplicate tweets
x <- x[!duplicated(x), ]

#Duplicate "Text" field column
x$OriginalText = x$Text

#Lower all the letters
x$Text <- tolower(x$Text)

#Text Preprocessing
##Removal of Retweets
x$Text <- gsub("RT @[a-z,A-Z]*:", "", x$Text)

##Removal of HTML Links - Need qdapRegex Package to use rm_url
x$Text <- rm_url(x$Text, pattern=pastex("@rm_twitter_url", "@rm_url"))

##Removal of @People
x$Text <- gsub("@\\w+", "", x$Text)

##Removal of Special Characters ?& .
x$Text <- gsub("?", " ", x$Text, fixed = TRUE)
x$Text <- gsub(".", " ", x$Text, fixed = TRUE)
x$Text <- gsub("!", " ", x$Text, fixed = TRUE)
x$Text <- gsub("\"", " ", x$Text, fixed = TRUE)


#Text Refinement
#Remove Stop words
rm_words <- function(string, words) {
stopifnot(is.character(string), is.character(words))
spltted<- strsplit(string, " ", fixed = TRUE) # fixed = TRUE for speedup
vapply(spltted, function(x) paste(x[!tolower(x) %in% words], collapse = " "),
character(1))
}

x$Text <-  rm_words(x$Text, tm::stopwords("en"))


#Function for Positive & Negative Words match
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
require(plyr)
require(stringr)


scores = laply(sentences, function(sentence, pos.words, neg.words) {

   # convert to lower case:
sentence = tolower(sentence)

   # split into words. str_split is in the stringr package
   word.list = str_split(sentence, '\\s+')
   # sometimes a list() is one level of hierarchy too much
words = unlist(word.list)

   # compare our words to the dictionaries of positive & negative terms
   pos.matches = match(words, pos.words)
   neg.matches = match(words, neg.words)

   # match() returns the position of the matched term or NA
   # we just want a TRUE/FALSE:
   pos.matches = !is.na(pos.matches)
   neg.matches = !is.na(neg.matches)

   # and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():
score = sum(pos.matches) - sum(neg.matches)

return(score)
  }, pos.words, neg.words, .progress=.progress )

  scores.df = data.frame(score=scores, text=sentences)
return(scores.df)
}

#Import Positive & Negative Words
pos_words <-
scan("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\Positive.txt",
what='character', comment.char=';')

neg_words <-
scan("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\Negative.txt",
what='character', comment.char=';')
```

```
#Add additional words to dictionaries
#neg_words = c(neg_words,'no')

#Score based on Positive & Negative words
result1 <- score.sentiment( x$Text, pos_words, neg_words)

#Calculate Lexicon Score
x$Lexicon = result1$score

#Calculate Total Score
x$TotalScore = (x$Lexicon)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))

Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+Performance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
```

```
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+Performance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+Performance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step01_Lexicon = Performance_matrix
Performance_matrix_Step01_Lexicon
```

Appendix B *– R code for using SentiWordNet*

```
#Step 01 - Run Python Script
import pandas as pd

x = pd.read_csv('C:/Users/Sachi/Desktop/MscProject/TwitterFeeds/TestData01.txt',
sep='\t' , encoding='latin-1', skiprows=1, names = ["Text", "Rating"])

x = x.fillna(")

def sentiwordnet_python(doc):
import nltk
from nltk.corpus import sentiwordnet as swn
  #doc=   "Nice and friendly place with excellent food and friendly and helpful staff.
You need a car though. The children wants to go back! Playground and animals
entertained them and they felt like at home. I also recommend the dinner! Great value
for the price!"
sentences = nltk.sent_tokenize(doc)
stokens = [nltk.word_tokenize(sent) for sent in sentences]
taggedlist=[]
for stoken in stokens:
taggedlist.append(nltk.pos_tag(stoken))
wnl = nltk.WordNetLemmatizer()

  score_list=[]
for idx,taggedsent in enumerate(taggedlist):
    score_list.append([])
for idx2,t in enumerate(taggedsent):
newtag="
lemmatized=wnl.lemmatize(t[0])
if t[1].startswith('NN'):
newtag='n'
elif t[1].startswith('JJ'):
newtag='a'
elif t[1].startswith('V'):
newtag='v'
elif t[1].startswith('R'):
newtag='r'
else:
newtag="
if(newtag!="):
synsets = list(swn.senti_synsets(lemmatized, newtag))
        #Getting average of all possible sentiments, as you requested
score=0
if(len(synsets)>0):
for syn in synsets:
score+=syn.pos_score()-syn.neg_score()
          score_list[idx].append(score/len(synsets))

  #print(score_list)
```

```
    sentence_sentiment=[]

for score_sent in score_list:
if len(score_sent)>0:
            sentence_sentiment.append(sum([word_score for word_score in
score_sent])/len(score_sent))
        #print("Sentiment for each sentence for:"+doc)
        #print(sentence_sentiment)
return sentence_sentiment

for row in x.itertuples():
x['SentiWordNetScore'] = x.apply(lambda row: sentiwordnet_python(row.Text),
axis=1)

x.to_csv('C:/Users/Sachi/Desktop/MscProject/Outputs/step01_sentiwordnet.txt',
sep='\t', encoding='latin-1')
```

```
#Step 02 - Evaluation on R

#Calculate Sentiwordnet
y <- fread ("C:/Users/Sachi/Desktop/MscProject/Outputs/step01_sentiwordnet.txt",
header = TRUE, select = c("Text", "Rating", "SentiWordNetScore"))

#Remove Null values
y$SentiWordNetScore <- gsub("[[]]", "0.0", y$SentiWordNetScore)

#Remove Special characters from score column
y$SentiWordNetScore <- gsub("[[]", "", y$SentiWordNetScore)
y$SentiWordNetScore <- gsub("[]]", "", y$SentiWordNetScore)


#Assign to x
x <- y

#Convert SentiWordNet Score to numeric value
x$SentiWordNetScore = as.numeric(x$SentiWordNetScore)

#Replace NUll values to 01m1q
x$SentiWordNetScore = ifelse(is.na(x$SentiWordNetScore) == 'TRUE', 0.00,
x$SentiWordNetScore)

#Calculate Total Score
x$TotalScore = (x$SentiWordNetScore)

#Assign Predicted Rating
```

```
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))


Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
```

```
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+Performance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step01_SentiWordNet = Performance_matrix
Performance_matrix_Step01_SentiWordNet
```

Appendix C – R code for using Lexicon Dictionary with Stop Words Amendments

```
#Import Twitter Feeds
x <- fread ("C:\\Users\\Sachi\\Desktop\\MscProject\\TwitterFeeds\\TestData01.txt",
header = TRUE, select = c("Text","Rating"))

#Remove Duplicate tweets
x <- x[!duplicated(x), ]

#Duplicate "Text" field column
x$OriginalText = x$Text

#Lower all the letters
x$Text <- tolower(x$Text)

#Text Preprocessing
##Removal of Retweets
x$Text <- gsub("RT @[a-z,A-Z]*:", "", x$Text)

##Removal of HTML Links - Need qdapRegex Package to use rm_url
x$Text <- rm_url(x$Text, pattern=pastex("@rm_twitter_url", "@rm_url"))

##Removal of @People
x$Text <- gsub("@\\w+", "", x$Text)

##Removal of Special Characters ?& .
x$Text <- gsub("?", " ", x$Text, fixed = TRUE)
x$Text <- gsub(".", " ", x$Text, fixed = TRUE)
x$Text <- gsub("!", " ", x$Text, fixed = TRUE)
x$Text <- gsub("\"", " ", x$Text, fixed = TRUE)


#Text Refinement
#Remove Stop words
rm_words <- function(string, words) {
stopifnot(is.character(string), is.character(words))
splted<- strsplit(string, " ", fixed = TRUE) # fixed = TRUE for speedup
vapply(splted, function(x) paste(x[!tolower(x) %in% words], collapse = " "),
character(1))
}

#Customize the stop words
exceptions<- c("no")
my_stopwords <- setdiff(tm::stopwords("en"), exceptions)

#x$Text <-  rm_words(x$Text, tm::stopwords("en"))
x$Text <-  rm_words(x$Text, my_stopwords)

#Score based on Positive & Negative words
result1 <- score.sentiment( x$Text, pos_words, neg_words)
```

```
#Calculate Lexicon Score
x$Lexicon = result1$score

#Calculate Total Score
x$TotalScore = (x$Lexicon)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))

Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
```

```
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step03_Lexicon_StopWords = Performance_matrix
Performance_matrix_Step03_Lexicon_StopWords
```

Appendix D – R code for comparing Lexicon Dictionary with SentiWordNet

```
#Step 01-Input to Python

library(NLP)
library(tm)
library(regexr)

# Loading Negative words into two files
# Here *words have to be removed such as d*mn
neg_words_part1 <-
scan("C:/Users/Sachi/Desktop/MscProject/Dictionaries/Negative_Part1.txt",
what='character', comment.char=';')
neg_words_part2 <-
scan("C:/Users/Sachi/Desktop/MscProject/Dictionaries/Negative_Part2.txt",
what='character', comment.char=';')

# Remove positive words
a = removeWords(x$Text,pos_words)

# Remove negative words
b = removeWords(a,neg_words_part1)
x$SentiWordNet_Text = removeWords(b,neg_words_part2)


# Export data to a text file
write.table(x, "C:/Users/Sachi/Desktop/MscProject/Outputs/rdata_Step04.txt",
sep="\t",row.names=F)


#Step 02 - Run Python Script
import pandas as pd

x = pd.read_csv('C:/Users/Sachi/Desktop/MscProject/Outputs/rdata_Step04.txt',
sep='\t' , encoding='latin-1', skiprows=1, names = ["Text", "Rating", "OriginalText",
"Lexicon", "TotalScore", "PredictedRating", "SentiWordNet_Text"])

x = x.fillna('')

def sentiwordnet_python(doc):
import nltk
from nltk.corpus import sentiwordnet as swn
   #doc=   "Nice and friendly place with excellent food and friendly and helpful staff.
You need a car though. The children wants to go back! Playground and animals
entertained them and they felt like at home. I also recommend the dinner! Great value
for the price!"
sentences = nltk.sent_tokenize(doc)
stokens = [nltk.word_tokenize(sent) for sent in sentences]
taggedlist=[]
for stoken in stokens:
```

```
taggedlist.append(nltk.pos_tag(stoken))
wnl = nltk.WordNetLemmatizer()

   score_list=[]
for idx,taggedsent in enumerate(taggedlist):
      score_list.append([])
for idx2,t in enumerate(taggedsent):
newtag=''
lemmatized=wnl.lemmatize(t[0])
if t[1].startswith('NN'):
newtag='n'
elif t[1].startswith('JJ'):
newtag='a'
elif t[1].startswith('V'):
newtag='v'
elif t[1].startswith('R'):
newtag='r'
else:
newtag=''
if(newtag!=''):
synsets = list(swn.senti_synsets(lemmatized, newtag))
            #Getting average of all possible sentiments, as you requested
score=0
if(len(synsets)>0):
for syn in synsets:
score+=syn.pos_score()-syn.neg_score()
               score_list[idx].append(score/len(synsets))

   #print(score_list)
   sentence_sentiment=[]

for score_sent in score_list:
if len(score_sent)>0:
         sentence_sentiment.append(sum([word_score for word_score in
score_sent])/len(score_sent))
      #print("Sentiment for each sentence for:"+doc)
      #print(sentence_sentiment)
return sentence_sentiment

for row in x.itertuples():
x['SentiWordNetScore'] = x.apply(lambda row:
sentiwordnet_python(row.SentiWordNet_Text), axis=1)

x.to_csv('C:/Users/Sachi/Desktop/MscProject/Outputs/step04_sentiwordnet.txt',
sep='\t', encoding='latin-1')
```

```
#Step 03 - Evaluation on R

#Calculate Sentiwordnet
y <- fread ("C:/Users/Sachi/Desktop/MscProject/Outputs/step04_sentiwordnet.txt",
header = TRUE, select = c("Text", "Rating", "OriginalText", "Lexicon", "TotalScore",
"PredictedRating", "SentiWordNet_Text", "SentiWordNetScore"))

#Remove Null values
y$SentiWordNetScore <- gsub("[[]]", "0.0", y$SentiWordNetScore)

#Remove Special characters from score column
y$SentiWordNetScore <- gsub("[[]", "", y$SentiWordNetScore)
y$SentiWordNetScore <- gsub("[]]", "", y$SentiWordNetScore)


#Assign to x
x <- y

#Convert SentiWordNet Score to numeric value
x$SentiWordNetScore = as.numeric(x$SentiWordNetScore)

#Replace NULL values to 0
x$SentiWordNetScore = ifelse(is.na(x$SentiWordNetScore) == 'TRUE', 0.00,
x$SentiWordNetScore)

#Calculate Total Score
x$TotalScore = (x$Lexicon)+(x$SentiWordNetScore)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))


Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
```

```
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+P
erformance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step04_Lexicon_SentiWordNet = Performance_matrix
Performance_matrix_Step04_Lexicon_SentiWordNet
```

Appendix E – R code for using Lexicon Dictionary with Slang Replacements

```
#Import Slang Dictionary-special charators
slangs_sc <- fread
("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\slangs_specialchar.csv",
header = TRUE, select = c("Slang","Slang_Desc"))

slangs_other <- fread
("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\slangs_others.csv", header =
TRUE, select = c("Slang","Slang_Desc"))

slangs_sc$Slang <- tolower(slangs_sc$Slang)
slangs_other$Slang <- tolower(slangs_other$Slang)

z <- x

#Duplicate "Text" field column
#z$OriginalText = z$Text


#Replace twitter feeds with slangs
for (q in 1:nrow(z))
for(t in 1:nrow(slangs_sc))
    #For special characters
  { z[q,1] <- gsub((str_c(" \\", slangs_sc[t,1]," ")), str_c(" ", slangs_sc[t,2]," "),
z[q,1])}

for (q in 1:nrow(z))
z[q,1] <- gsub(":\\)", "Positive", z[q,1])

for (q in 1:nrow(z))
z[q,1] <- gsub(":\\(", "Negative", z[q,1])

for (q in 1:nrow(z))
z[q,1] <- gsub(":p", "Positive", z[q,1])

for (q in 1:nrow(z))
z[q,1] <- gsub(":d", "Positive", z[q,1])

#for (q in 1:nrow(z))
#z[q,1] <- gsub("xd", "Positive", z[q,1])



for (q in 1:nrow(z))
for(t in 1:nrow(slangs_other))
    #For others characters
  { z[q,1] <- gsub((str_c(" ",slangs_other[t,1]," ")),  (str_c(" ", slangs_other[t,2]," ")),
z[q,1])}
```

```
#Replace twitter feeds with slangs
#for (q in 1:nrow(z))
# for(t in 1:nrow(slangs))
#  #For special characters
# if (substring(slangs[t,1], 1, 1) %in% c("$","*","/","<",">","?","@","^",":"))
#{ z[q,1] <- gsub((str_c(" \\", slangs[t,1]," ")), str_c(" ", slangs[t,2]," "), z[q,1])} else {
z[q,1] <- gsub((str_c(" ",slangs[t,1]," ")),  (str_c(" ", slangs[t,2]," ")), z[q,1])}


x$Slangs_Text <- tolower(z$Text)

#Text Preprocessing
##Removal of Retweets
x$Slangs_Text <- gsub("RT @[a-z,A-Z]*:", "", x$Slangs_Text)

##Removal of HTML Links - Need qdapRegex Package to use rm_url
x$Slangs_Text <- rm_url(x$Slangs_Text, pattern=pastex("@rm_twitter_url",
"@rm_url"))

##Removal of @People
x$Slangs_Text <- gsub("@\\w+", "", x$Slangs_Text)

##Removal of Special Characters ?& .
x$Slangs_Text <- gsub("?", " ", x$Slangs_Text, fixed = TRUE)
x$Slangs_Text <- gsub(".", " ", x$Slangs_Text, fixed = TRUE)
x$Slangs_Text <- gsub("!", " ", x$Slangs_Text, fixed = TRUE)
x$Slangs_Text <- gsub("\"", " ", x$Slangs_Text, fixed = TRUE)

#Text Refinement
#Remove Stop words
rm_words <- function(string, words) {
stopifnot(is.character(string), is.character(words))
spltted<- strsplit(string, " ", fixed = TRUE) # fixed = TRUE for speedup
vapply(spltted, function(x) paste(x[!tolower(x) %in% words], collapse = " "),
character(1))
}
#Customize the stop words
exceptions<- c("no")
my_stopwords <- setdiff(tm::stopwords("en"), exceptions)

#x$Text <-  rm_words(x$Text, tm::stopwords("en"))
x$Slangs_Text <-  rm_words(x$Slangs_Text, my_stopwords)

#Score based on Positive & Negative words
result1 <- score.sentiment(x$Slangs_Text, pos_words, neg_words)

#Calculate Lexicon Score
x$Lexicon_Slangs_Score = result1$score
```

```
#Calculate Total Score
x$TotalScore = (x$Lexicon_Slangs_Score)

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))


Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")


#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+P
erformance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+P
erformance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))
```

```
#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+Performance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step05_Lexicon_Slang = Performance_matrix
Performance_matrix_Step05_Lexicon_Slang
```

Appendix F – R code for using Lexicon Dictionary with Slang Replacements and Emoticons

```
#Import Emoticon Dictionary
emoticons<- fread
("C:\\Users\\Sachi\\Desktop\\MscProject\\Dictionaries\\Emoticons.txt", header =
TRUE, select = c("Emoticon","Sentiment_score"))

emoticons$Tag <- ""

#Tag Positive & Negative Emoticons
for (q in 1:nrow(emoticons))
  emoticons[q,3] <-
ifelse(emoticons[q,2]>0,"Positive",ifelse(emoticons[q,2]==0,"Neutral","Negative"))

emoticons_pos <-  emoticons[emoticons$Tag %in% "Positive"]
emoticons_neg <-  emoticons[emoticons$Tag %in% "Negative"]


#Score based on Emoticons
result2 <- score.sentiment(x$Text, tolower(emoticons_pos$Emoticon),
tolower(emoticons_neg$Emoticon))

#Calculate Lexicon Score
x$Emoticons = result2$score

#Calculate Total Score
x$TotalScore = x$Lexicon_Slangs_Score  + x$Emoticons

#Assign Predicted Rating
x$PredictedRating = ifelse(x$TotalScore < 0, 'Negative',
(ifelse(x$TotalScore==0,'Neutral','Positive')))

#Calculate Performance Matrix
#For Positive Reviews
TP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating == 'Positive', 1, 0))
FP1 = sum(ifelse(x$PredictedRating == 'Positive' & x$Rating != 'Positive', 1, 0))
FN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating == 'Positive', 1, 0))
TN1 = sum(ifelse(x$PredictedRating != 'Positive' & x$Rating != 'Positive', 1, 0))

Performance_matrix <- matrix(ncol=4,nrow=3,byrow=TRUE)
rownames(Performance_matrix) <- c("Positive","Negative","Neutral")
colnames(Performance_matrix) <- c("Accuracy","Precision","Recall","F_Measure")

#Performance Metrics
Performance_matrix[1,1] = (TP1+TN1)/(TP1+FP1+FN1+TN1)
Performance_matrix[1,2] = TP1/(TP1+FP1)
Performance_matrix[1,3] = TP1/(TP1+FN1)
```

```
Performance_matrix[1,4]=
(2*Performance_matrix[1,2]*Performance_matrix[1,3])/(Performance_matrix[1,2]+Performance_matrix[1,3])


#For Negative Reviews
TP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating == 'Negative', 1, 0))
FP2 = sum(ifelse(x$PredictedRating == 'Negative' & x$Rating != 'Negative', 1, 0))
FN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating == 'Negative', 1, 0))
TN2 = sum(ifelse(x$PredictedRating != 'Negative' & x$Rating != 'Negative', 1, 0))


#Performance Metrics
Performance_matrix[2,1] = (TP2+TN2)/(TP2+FP2+FN2+TN2)
Performance_matrix[2,2] = TP2/(TP2+FP2)
Performance_matrix[2,3] = TP2/(TP2+FN2)
Performance_matrix[2,4]=
(2*Performance_matrix[2,2]*Performance_matrix[2,3])/(Performance_matrix[2,2]+Performance_matrix[2,3])

#For Neutral Reviews
TP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating == 'Neutral', 1, 0))
FP3 = sum(ifelse(x$PredictedRating == 'Neutral' & x$Rating != 'Neutral', 1, 0))
FN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating == 'Neutral', 1, 0))
TN3 = sum(ifelse(x$PredictedRating != 'Neutral' & x$Rating != 'Neutral', 1, 0))

#Performance Metrics
Performance_matrix[3,1] = (TP3+TN3)/(TP3+FP3+FN3+TN3)
Performance_matrix[3,2] = TP3/(TP3+FP3)
Performance_matrix[3,3] = TP3/(TP3+FN3)
Performance_matrix[3,4]=
(2*Performance_matrix[3,2]*Performance_matrix[3,3])/(Performance_matrix[3,2]+Performance_matrix[3,3])

#Label the Performance Metric
Performance_matrix_Step06_Lexicon_Slang_Emoticons = Performance_matrix
Performance_matrix_Step06_Lexicon_Slang_Emoticons
```