MENTAL HEALTH MONITORING USING CHATBOT

Shreyansh Kumar

Keywords: Sentimental Analysis, Emotion Classification, Mental Health, Chat Bot

Abstract:

There is enough evidence to prove that overall physical, psychological and communal welfare of a human being is predominantly dependent on their mental health. Depression is a common illness worldwide, with more than 264 million people affected. Close to 800 000 people die due to suicide every year. Suicide is the second leading cause of death in 15-29-year-olds. Thus, early recognition and mediation in addressing issues regarding it is of utmost importance. Individualized and ubiquitous sensing technologies such as smartphones, smartwatches etc. allow continuous tracking and gathering of data in an undisturbed and low-profile manner. Sentimental Analysis using Natural Language Processing has been proposed to be applied to the collected data to monitor. This project aims at gathering the data from the user through a chatbot which will be analysed by the system to prepare and publish the report to the user to keep them aware of their mental health.

1. INTRODUCTION:

1.1 Objective:

We propose to design a novel health monitoring system using Telegram chatbot which interacts with the user and makes observations on their conversation patterns. The system essentially tries to understand the emotional and mental state of the user based on the words they use to converse with the bot. The system then keeps a track of the changes of emotional and mental state of the person to make a monthly report on the mental health of the user. This system can be very crucial in keeping a track of the user to let them know how they are doing mentally.

1.2 Libraries Used:

telegram.bot:

This package provides a pure R interface for the Telegram Bot API. In addition to the pure API implementation, it features a number of tools to make the development of Telegram bots with R easy and straightforward, providing an easy-to-use interface that takes some work off the programmer.

tm:

The tm package utilizes the Corpus as its main structure. A corpus is simply a collection of documents, but like most things in R, the corpus has specific attributes that enable certain types of analysis. Corpora in R exist in two ways: 1) Volatile Corpus (VCorpus) is a temporary object within R and is the default when assigning documents to a corpus. 2) Permanent Corpus (PCorpus) is a permanent object that can be stored outside of R.

wordcloud:

Text mining methods allow us to highlight the most frequently used keywords in a paragraph of texts. One can create a word cloud. The text mining package (tm) and the word cloud generator package

(wordcloud) are available in R for helping us to analyze texts and to quickly visualize the keywords as a word cloud.

SnowballC:

An R interface to the C 'libstemmer' library that implements Porter's word stemming algorithm for collapsing words to a common root to aid comparison of vocabulary. Currently supported languages are Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Romanian, Russian, Spanish, Swedish and Turkish.

RColorBrewer:

RColorBrewer is an R package that uses the work from http://colorbrewer2.org/ to help you choose sensible color schemes for figures in R. The colors are split into three groups, sequential, diverging, and qualitative.

- Sequential – Light colors for low data, dark for high data
- Diverging – Light colors for mid-range data, low and high contrasting dark colors
- Qualitative – Colors designed to give maximum visual difference between classes

syuzhet:

Syuzhet package is used for generating sentiment scores, it has four sentiment dictionaries and offers a method for accessing the sentiment extraction tool developed in the NLP group at Stanford. The get_sentiment function accepts two arguments: a character vector (of sentences or words) and a method. The selected method determines which of the four available sentiment extraction methods will be used. The four methods are syuzhet (this is the default), bing, afinn and nrc. Each method uses a different scale and hence returns slightly different results. Please note the outcome of the NRC method is more than just a numeric score, requires additional interpretations and is out of scope for this article.

ggplot:

ggplot2 is a plotting package that provides helpful commands to create complex plots from data in a data frame. It provides a more programmatic interface for specifying what variables to plot, how they are displayed, and general visual properties. Therefore, we only need minimal changes if the underlying data change or if we decide to change from a bar plot to a scatterplot. This helps in creating publication quality plots with minimal amounts of adjustments and tweaking.

1.3 Features:

- A chatbot which wishes the user every night and gathers data by asking about their day.
- Telegram app is used for the Chatbot implementation
- Text cleaning and Text mining to process that uses to natural language processing to structure the data
- Bar plot and Word cloud based on the most frequent words in the data
- Sentimental analysis and emotion classification using syuzhet package
- Chatbot feature automated messaging and replying with sending photos and text

2. REQUIREMENTS AND PROPOSED SYSTEM

2.1 Software Requirements:

RStudio

The RStudio IDE is a set of integrated tools designed to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports

direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

Figure 1: RStudio logo

Telegram:

Telegram is a freeware, cross-platform, cloud-based instant messaging (IM) system. The service also provides end-to-end encrypted video calling, file sharing and several other features. It was launched for iOS on 14 August 2013 and Android in October 2013. The servers of Telegram are distributed worldwide to decrease frequent data load with five data centers in different regions. All of Telegram's official components are open source, with the exception of the server which is closed-source and proprietary. Telegram offers free to use bot services to the users.

Figure 2: Telegram

2.2 Proposed System

2.2.1 Background

The system makes use of the chatbot to get conversational inputs from the user. These texts are then cleaned and mined using various libraries to classify them based on various emotional sentiments and the results are analyzed.

2.2.2 Algorithm

Step1: Greet the user at the time set by the user and ask about their day.

Step2: Retrieve the message received from the user from the chatbot

Step3: Perform data cleaning using the tm library.

Step 4: Perform text mining using the Natural Language Processing (NLP) and structure the useful data from the original received data

Step5: Using the snowballc library eliminate the stem words and reduce all the words to their root form.

Step6: Build a term document matrix with the processed data.

Step7: Generate word cloud and a bar plot with the most frequent words and save the plots as images in the folder.

Step8: Perform Sentiment analysis using the syuzhet package with all the three available methods. I.e., syuzhet_vector, bing, affin and combine all the data

Step 9: Perform emotion classification using NRC emotion lexicon and visualize the results using barplots and save the image in the folder.

Step10: Send back the results to the user through the chatbot.

2.2.3 Block Diagram:

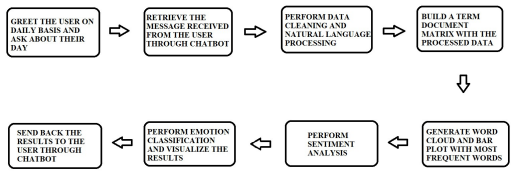

Figure 3: Block Diagram of the Algorithm

2.3 Advantages:

• keeps the user in track of their daily emotions.

• Easy to use procedure.

• Pictorially visualized results.

• Only the emotions will be stored in the system but not the user texts, so user data privacy is respected.

2.4 Limitations:

- No algorithm can predict the ideal result of sentiment
- System cannot accept the voice messages
- Chatbot solely relied on telegram app

3. MODULE DESCRIPTION

3.1 Chat Bot Creation:

First, one must have or create a Telegram account. Second, a Telegram Bot needs to be created in order to get an Access Token. It can be done so by talking to @BotFather and following a few simple steps. Telegram bots can receive messages or commands. The former is simply text that you send as if you were sending a message to another person, while the latter are prefixed with a / character. To create a new bot, send the following command to BotFather as a chat (exactly as if you were talking to another person on Telegram).

The telegram BOT can be controlled by sending HTTPS requests to Telegram. This means that the simplest way to interact with the Bot is through a web browser. By visiting different URLs, the bot can be sent different commands. For Example, the following simple command gets the information about the Bot. Visit the following URL in your browser (the corresponding TOKEN shall be substituted in the URL before)

https://api.telegram.org/bot<your-bot-token>/getMe

The first part of the URL indicates that the program wants to communicate with the Telegram API (api.telegram.org). This same URL is followed by /bot to say that you want to send a command to your Bot, and immediately after you add your TOKEN to identify which bot you want to send the command to and to prove that you own it. Finally, you specify the command that you want to send (/getMe) which in this case just returns basic information about our Bot using JSON.

3.2 Retrieving Messages From the Bot: The simplest way to retrieve messages sent to the Bot is through the getUpdates call:

https://api.telegram.org/bot<your-bot-token>/getUpdates

On hitting this URL, the program receives a JSON response of all the new messages sent to your Bot.

3.3 Sending Messages Through The Bot:

The chat ID for the chat where you want to send the message is required to perform this task. There are a bunch of different IDs in the JSON response from the getUpdates call, select the id present in the id field which is inside the chat field. Substituting the <chat-id> in the below URL the program will be able to hit it and send the required message to the user through chatbot.

https://api.telegram.org/bot<your-bot-token>/sendMessage?chat_id=<chat-id>&text=TestReply

The required text to be sent can be given in the text field inside the URL

3.4 Sentimental Analysis:

Sentiments can be classified as positive, neutral or negative. They can also be represented on a numeric scale, to better express the degree of positive or negative

strength of the sentiment contained in a body of text.

This example uses the Syuzhet package for generating sentiment scores, which has four sentiment dictionaries and offers a method for accessing the sentiment extraction tool developed in the NLP group at Stanford. The get_sentiment function accepts two arguments: a character vector (of sentences or words) and a method. The selected method determines which of the four available sentiment extraction methods will be used. The four methods are syuzhet (this is the default), bing, afinn and nrc. Each method uses a different scale and hence returns slightly different results. The outcome of the nrc method is more than just a numeric score, requires additional interpretations and is out of scope for this article.

3.5 Syuzhet Vector:

The scale for sentiment scores using the syuzhet method is decimal and ranges from -1(indicating most negative) to +1(indicating most positive). summary(syuzhet_vector) gives the maximum, minimum, mean, median and quartile values of the vector. Note that the summary statistics of the syuzhet vector represents a positive emotion for the values greater than zero and negative emotion for the values less than zero.

3.6 BING and AFFIN:

Each method has its own process of calculating and classifying the emotion and is represented on unique scales as below

- bing – binary scale with -1 indicating negative and +1 indicating positive sentiment
- afinn – integer scale ranging from -5 to +5

Because these different methods use different scales, it's better to convert their output to a common scale before comparing them. This basic scale conversion can be done easily using R's built-in sign function, which converts all positive numbers to 1, all negative numbers to -1 and all zeros remain 0.

3.7 Emotion Classification:

Emotion classification is built on the NRC Word-Emotion Association Lexicon (aka EmoLex). The NRC Emotion Lexicon is a list of English words and their associations with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive). The annotations were manually done by crowdsourcing.

The get_nrc_sentiments function, returns a data frame with each row representing a sentence from the original file. The data frame has ten columns (one column for each of the eight emotions, one column for positive sentiment valence and one for negative sentiment valence). The data in the columns (anger, anticipation, disgust, fear, joy, sadness, surprise, trust, negative, positive) can be accessed individually or in sets.

4. RESULTS AND DISCUSSIONS

4.1 Overview:

The system is basically two parts combined together. One is sentimental analysis and emotion classification and the other is chatbot implementation. The sentimental analysis is done with the help of the syuzhet package developed at Stanford which has a standard of 70 to 80% accuracy in results. The system job is to receive the text from the user and then retrieve the text into the program and analyze the sentiment and then to send back the results to the user through the chatbot. The results of the system are explained in two parts below.

4.2 Sentimental Analysis and Emotion Classification:

For this part the data has to be cleaned and mined. And the most frequently occuring words in the data are visualized through the following plots on the barplot and the word cloud as shown below. Figure 4 shows the bar graph for the top 5 most frequent words and figure 6 shows the word cloud

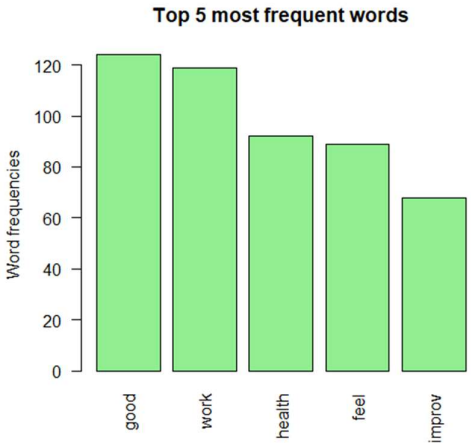

Figure 4: Top 5 Most Frequent Words

Figure 5: Word Cloud of Frequent Words

Then the data is used for sentimental analysis using the syuzhet library with all the three methods. The results of all the three methods are shown in the following figure 6.

An inspection of the Syuzhet vector shows the first element has the value of 2.60. It means the sum of the sentiment scores of all meaningful words in the first response(line) in the text file, adds up to 2.60. The scale for sentiment scores using the syuzhet method is decimal and ranges from -1(indicating most negative) to +1(indicating most positive). Note that the summary statistics of the syuzhet vector show a median value of 1.6, which is above zero and can be interpreted as the overall average sentiment across all the responses is positive.

The summary statistics of bing and afinn vectors also show that the Median value of Sentiment scores is above 0 and can be interpreted as the overall average sentiment across all the responses is positive.

```
> summary(syuzhet_vector)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -1.450   0.900   1.600   1.883   2.650   9.000
> summary(bing_vector)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.000   1.000   2.000   2.007   3.000   9.000
> summary(afinn_vector)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -6.00    2.00    4.00    4.42    7.00   18.00
> rbind(
+     +   sign(head(syuzhet_vector)),
+     +   sign(head(bing_vector)),
+     +   sign(head(afinn_vector))
+
+ )
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    1    1    1    1
[2,]    1    1    1   -1    1    1
[3,]    1    1    1    1    1    1
```

Figure 6: Syuzhet Library Results

Because these different methods use different scales, it's better to convert their output to a common scale before comparing them. This basic scale conversion can be done easily using R's built-in sign function, which converts all positive numbers to 1, all negative numbers to -1 and all zeros remain 0.

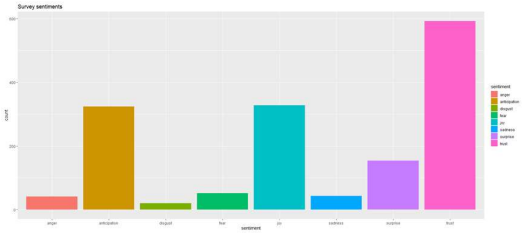

Figure 7: Emotion Classification

The above figure 7 shows that words associated with the positive emotion of

"trust" occurred about five hundred times in the text, whereas words associated with the negative emotion of "disgust" occurred less than 25 times. A deeper understanding of the overall emotions occurring in the survey response can be gained by comparing these numbers as a percentage of the total number of meaningful words. Add the following code to your R script and run it.

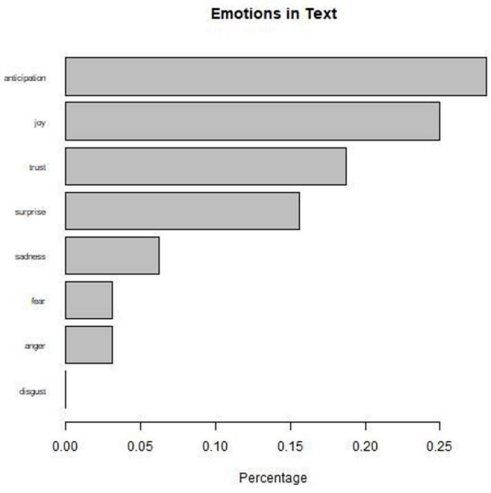

Figure 8: Emotions in Percentage

This bar plot in figure 8 allows for a quick and easy comparison of the proportion of words associated with each emotion in the text. The emotion "trust" has the longest bar and shows that words associated with this positive emotion constitute just over 35% of all the meaningful words in this text. On the other hand, the emotion of "disgust" has the shortest bar and shows that words associated with this negative emotion constitute less than 2% of all the meaningful words in this text. Overall, words associated with the positive emotions of "trust" and "joy" account for almost 60% of the meaningful words in the text, which can be interpreted as a good sign of team health.

4.3 Chatbot Implementation:

The sentiment analysis discussed in the previous section is implemented in the chatbot through the telegram app. The chatbot is expected to greet the user on time basis and exchange the messages including results with the user. The figure 9 shows the scenario where the chatbot greeted the user according to the time set by saying "Hey! How's ur day?". And the user has also replied using the text message.

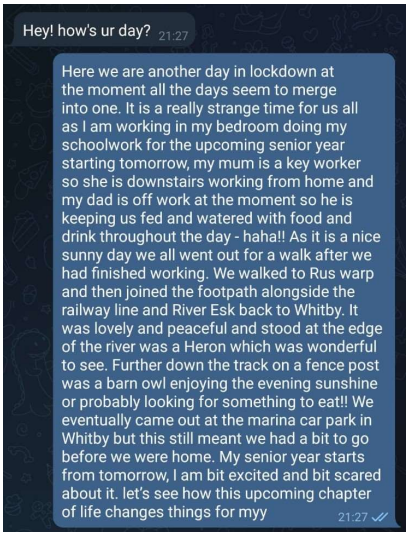

Figure 9: Automated Greetings by Bot

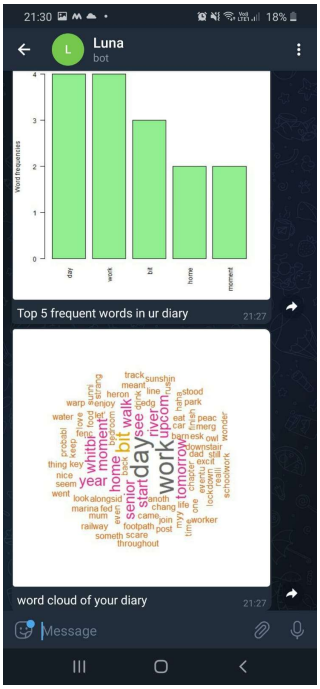

Figure 10: Chat Bot replies

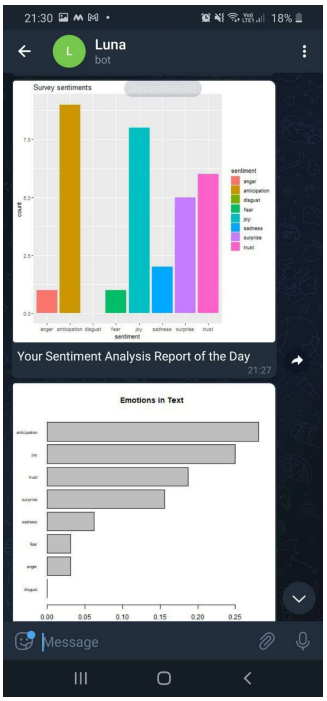

Figure 11: Chat Bot Replies

The above pictures show the result or output of sentimental analysis received by the user like the most frequent word, word cloud, emotions in text and sentimental analysis report. With the help of all this, the user can find his/her state of mind and can take precautionary measures if needed.

Hence, the main goal of mental health monitoring and reporting using chatbot has been implemented. The project is successful in performing its task of sentimental analysis of its user to help them with their mental state on a daily basis. The system implemented is highly efficient, the parameter for its efficiency being how well it identifies the sentiment involved in the text reply to the chatbot. By using this system, the user can better monitor their mental condition and try to lead a better, more fulfilling life.

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion:

The project, Mental health report using chatbot was designed and realized using RStudio. The System is implemented on various test text files which are received through a telegram to RStudio and the results are inferred. The software of this project system is simple and can be implemented with ease. The Sentimental analysis done using the Syuzhet package with its all methods like bing, affin, etc. functions properly in all the scenarios as pre-programmed. Since the outputs are pictorial representations of sentiment produced by the system using the syuzhet package which has an accuracy of 70-80 % so, one can consider it successfully functioning and can be used for non-professional use.

5.2 Future Works:

- Add voice message feature as a method of input along with the text.
- Make the chatbot to automatically suggest some movies or books to the user based on the estimated emotion result of the user.
- Can take the chatbot to more platforms like whatsapp, discord etc.

6. REFERENCES

6.1 Recent Researches in This Field:

Doaa Mohey El-Din Mohamed Hussein [1] proposed a new technique to analyze online reviews in the scientific research domain called: "Sentiment Analysis Of Online Papers" (SAOOP). SAOOP aims at supporting researchers and saving their time and efforts by enabling them to report the total evaluation for the papers.

Akshi Kumar and Teeja Mary Sebastian[2] proposed and investigated a paradigm to mine the sentiment from a popular real-time microblogging service, Twitter, where users post real time reactions to and

opinions about “everything”. In this paper, they expound a hybrid approach using both corpus based and dictionary based methods to determine the semantic orientation of the opinion words in tweets.

Changliang Li, Bo Xu,Gaowei Wu, Sk He, Guanhua Tian and Yujun Zhou[3] focused on the task of predicting sentiment label distribu-tions, which aims to predict sentiment label distributions on phrase and sentence level. There has recently been considerable progress in predicting sentiment label distributions.

[1] Doaa Mohey El-Din Mohamed Hussein(January 2016). Analyzing Scientific Papers Based on Sentiment Analysis (First Draft).

[2] Akshi Kumar, Teeja Mary Sebastian(July 2012).Sentiment Analysis on Twitter. International Journal of Computer Science Issues 9(3):372-378

[3]Changliang Li, Bo Xu,Gaowei Wu, Sk He, Guanhua Tian, Yujun Zhou(May 2015).Parallel Recursive Deep Model for Sentiment Analysis.Conference: Pacific-Asia Conference on Knowledge Discovery and Data Mining.

APPENDIX

Program:

```
library(telegram.bot)

library(stringr)

library("tm")

library("SnowballC")

library("wordcloud")

library("RColorBrewer")

library("syuzhet")

library("ggplot2")

bot_token <- "2022741682:AAEL3HuqJkU6gK-kG07aYZlM7hfUonid-Ik"

target_chat_id <- 909650045

bot <- Bot(token = bot_token)

bot_sendmessage <-

 function(text_to_sent) {

  bot$sendMessage(909650045,

          text_to_sent,

          parse_mode = NULL)

 }

Hour = "22"

Minute = "13"

remtime = paste(Hour, Minute, "00", sep ="")

yestext <- readLines("E:\\academics\\3rdyrfall\\data analytics\\Project\\dataset\\yestext.txt")

lasttext = yestext

greet = FALSE

clean = FALSE

donetoday = FALSE

while (!donetoday) {

 if(format(Sys.time(), "%H%M%S") == remtime){

 bot$clean_updates()

 clean = TRUE

 }

 if(greet){

  history <- bot$getUpdates()

  if (length(history) != 0) {
```

```
    lastmessage <- history[[length(history)]]
    lasttext <- lastmessage$message$text
    lastuser <- lastmessage$message$from$first_name
    print(lastuser)
    print(lasttext)
   }
  }
   if(lasttext != yestext){
    yestext <- lasttext
    writeLines(yestext, "E:\\academics\\3rdyrfall\\data analytics\\Project\\dataset\\yestext.txt")
    text <- lasttext
    TextDoc <- Corpus(VectorSource(text))
    toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
    TextDoc <- tm_map(TextDoc, toSpace, "/")
    TextDoc <- tm_map(TextDoc, toSpace, "@")
    TextDoc <- tm_map(TextDoc, toSpace, "\\|")
    # Convert the text to lower case
    TextDoc <- tm_map(TextDoc, content_transformer(tolower))
    # Remove numbers
    TextDoc <- tm_map(TextDoc, removeNumbers)
    # Remove english common stopwords
    TextDoc <- tm_map(TextDoc, removeWords, stopwords("english"))
    # Remove your own stop word
    # specify your custom stopwords as a character vector
    TextDoc <- tm_map(TextDoc, removeWords, c("s", "company", "team"))
    # Remove punctuations
    TextDoc <- tm_map(TextDoc, removePunctuation)
    # Eliminate extra white spaces
    TextDoc <- tm_map(TextDoc, stripWhitespace)
    # Text stemming - which reduces words to their root form
    TextDoc <- tm_map(TextDoc, stemDocument)
    # Build a term-document matrix
    TextDoc_dtm <- TermDocumentMatrix(TextDoc)
    dtm_m <- as.matrix(TextDoc_dtm)
    # Sort by descearing value of frequency
    dtm_v <- sort(rowSums(dtm_m),decreasing=TRUE)
    dtm_d <- data.frame(word = names(dtm_v),freq=dtm_v)
    # Display the top 5 most frequent words
    head(dtm_d, 5)
    # Plot the most frequent words
    png(filename="C:\\Users\\Tejaswanth Maram\\Downloads\\bplot.png")
    barplot(dtm_d[1:5,]$freq, las = 2, names.arg = dtm_d[1:5,]$word,
            col ="lightgreen", main ="Top 5 most frequent words",
            ylab = "Word frequencies")
    dev.off()
```

```
bot$sendPhoto(
  chat_id = 909650045,
  photo = "C:\\Users\\Tejaswanth Maram\\Downloads\\bplot.png",
  caption = "Top 5 frequent words in ur diary"
)
#generate word cloud
png(filename="C:\\Users\\Tejaswanth Maram\\Downloads\\wcloud.png")
set.seed(1234)
wordcloud(words = dtm_d$word, freq = dtm_d$freq, min.freq = 5,
          max.words=100, random.order=FALSE, rot.per=0.40,
          colors=brewer.pal(8, "Dark2"))
dev.off()
bot$sendPhoto(
  chat_id = 909650045,
  photo = "C:\\Users\\Tejaswanth Maram\\Downloads\\wcloud.png",
  caption = "word cloud of your diary"
)
#associations
findAssocs(TextDoc_dtm, terms = c("good","work","health"), corlimit = 0.25)

findAssocs(TextDoc_dtm, terms = findFreqTerms(TextDoc_dtm, lowfreq = 50), corlimit = 0.25)
#syuzhet
syuzhet_vector <- get_sentiment(text, method="syuzhet")
print(head(syuzhet_vector))
print(summary(syuzhet_vector))
# bing
bing_vector <- get_sentiment(text, method="bing")
print(head(bing_vector))
print(summary(bing_vector))
#affin
afinn_vector <- get_sentiment(text, method="afinn")
print(head(afinn_vector))
print(summary(afinn_vector))
#compare the first row of each vector using sign function
rbind(
  sign(head(syuzhet_vector)),
  sign(head(bing_vector)),
  sign(head(afinn_vector))
)
d<-get_nrc_sentiment(text)
print(head (d,10))
td<-data.frame(t(d))
td_new <- data.frame(rowSums(td))
names(td_new)[1] <- "count"
td_new <- cbind("sentiment" = rownames(td_new), td_new)
rownames(td_new) <- NULL
td_new2<-td_new[1:8,]
jpeg(filename="C:\\Users\\Tejaswanth Maram\\Downloads\\nrcplot.jpg")
quickplot(sentiment, data=td_new2, weight=count, geom="bar", fill=sentiment, ylab="count")+ggtitle("Survey sentiments")
```

```
    dev.off()
    bot$sendPhoto(
      chat_id = 909650045,
      photo = "C:\\Users\\Tejaswanth Maram\\Downloads\\nrcplot1.jpg",
      caption = "Your Sentiment Analysis Report of the Day"
    )
    jpeg(filename="C:\\Users\\Tejaswanth Maram\\Downloads\\emoplot.jpg")
    barplot(
      sort(colSums(prop.table(d[, 1:8]))),
      horiz = TRUE,
      cex.names = 0.7,
      las = 1,
      main = "Emotions in Text", xlab="Percentage"
    )
    dev.off()
    bot$sendPhoto(
      chat_id = 909650045,
      photo = "C:\\Users\\Tejaswanth Maram\\Downloads\\emoplot.jpg",
      caption = "Your Sentiment Analysis Report of the Day"
    )
    bot_sendmessage("Wish you a great day tommorow, good night!")
    donetoday = TRUE
  }
  if (clean && !donetoday && !greet) {
    bot_sendmessage("Hey! how's ur day?")
    greet = TRUE
  }
}
```