Tele_Chat_Analysis

Jemima

2024-05-10

Telegram Chat Analysis

In this mini project, I will be text mining one of my telegram chats and analysing it with bag-of-words. The telegram messages come from the chat between me and my boyfriend, and spans a few days.

I hope to find out things like the most common words and phrases we use, how differently we text, and who texts more.

Loading in Packages Used

The text messages where obtained by exporting directly from the telegram app. It exports as html file only, hence I used the package rvest.

```
library(rvest)
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(stringr)
library(tm)
```

```
## Loading required package: NLP
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(plotrix)
library(RWeka)
```

Data Extraction

To extract the text messages, I initialised empty vectors and looped each node to extract the message's unique ID, sender's username, and content of the message.

I could not extract each element individually as I would lose the connection between each of them. For example, when sending two messages consecutively, the second message would not have a .from_name node as it follows that of the first message. Hence to keep track of who send which message, I used an empty vector recent_sender which will update to the person who sent the first message.

```
# Extracting Telegram Messages from exported html file
link = "messages6.html"
page = read_html(link)

all_msg_nodes = page %>% html_nodes(".message.default.clearfix")

# initialise
senders = character(length = length(all_msg_nodes))
ids = character(length = length(all_msg_nodes))
msgcontent = character(length = length(all_msg_nodes))

recent_sender = NULL

for (i in seq_along(all_msg_nodes)) {
  # Get the id of the current node
  ids[i] = html_attr(all_msg_nodes[i], "id")

  # Check if the node has a from_name
  if (length(html_nodes(all_msg_nodes[i], ".from_name")) > 0) {
    # Update the most recent from_name
    recent_sender = html_text(html_nodes(all_msg_nodes[i], ".from_name"))
  }
  # Assign the most recent from_name to the current node
  senders[i] = recent_sender

  if (length(html_nodes(all_msg_nodes[i], ".text")) > 0) {
    msgcontent[i] = html_text(html_nodes(all_msg_nodes[i], ".text"))
  }
}
```

Data Cleaning

```
# removing new lines
senders <- str_replace_all(senders, "\n", " ") %>%
  str_replace_all(., "\\s+", " ") %>%
  str_trim()

msgcontent <- str_replace_all(msgcontent, "\n", " ") %>%
```

```
  str_replace_all(., "\\s+", " ") %>%
  str_trim()

# combine all three columns
msgdata = as.data.frame(cbind(ids, senders, msgcontent))

# separate msg from marc and jemima
msgdata_marc = msgdata %>% filter(senders == "Marc Lim") %>% select(-senders)
msgdata_jemima = msgdata %>% filter(senders == "jemima") %>% select(-senders)

str(msgdata_marc)
```

```
## 'data.frame':    481 obs. of  2 variables:
##  $ ids       : chr  "message167403" "message167404" "message167405" "message167407" ...
##  $ msgcontent: chr  "What's the use bruh just use exact values" "Like" "Idk" "Nvm" ...
```

```
str(msgdata_jemima)
```

```
## 'data.frame':    313 obs. of  2 variables:
##  $ ids       : chr  "message167401" "message167402" "message167406" "message167408" ...
##  $ msgcontent: chr  "because approximate is approximate what" "if i can get exact isnt that better" "
```

Seems like Marc sends more texts than me. It could be because he is more of a "one word for one message" texter, while I squeeze more words into a single message.

Text Mining

Corpus Creation

```
marc_corpus = VCorpus(VectorSource(msgdata_marc$msgcontent))
jemima_corpus = VCorpus(VectorSource(msgdata_jemima$msgcontent))
```

Preprocessing

First created a function that removes punctuation, white space, numbers, stop words, and converts the string to lower case. For functions like tolower() which are build in functions, I have to wrap it in content_transformer().

```
# create tm_clean function
tm_clean = function(corpus){
  corpus = tm_map(corpus, removePunctuation)
  corpus = tm_map(corpus, stripWhitespace)
  corpus = tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus,content_transformer(tolower))
  corpus <- tm_map(corpus, removeWords, stopwords("en"))
  return(corpus)
}

# apply function to corpra
cm_corpus = tm_clean(marc_corpus)
cj_corpus = tm_clean(jemima_corpus)
```

Stemming

After removing stop words, which are uninformative, I need to reduce the remaining words into their base form. Albeit, I am using a crude method of stemming by simply stripping the ends of the words.

```
cm_corpus = tm_map(cm_corpus, stemDocument)
cj_corpus = tm_map(cj_corpus, stemDocument)
```

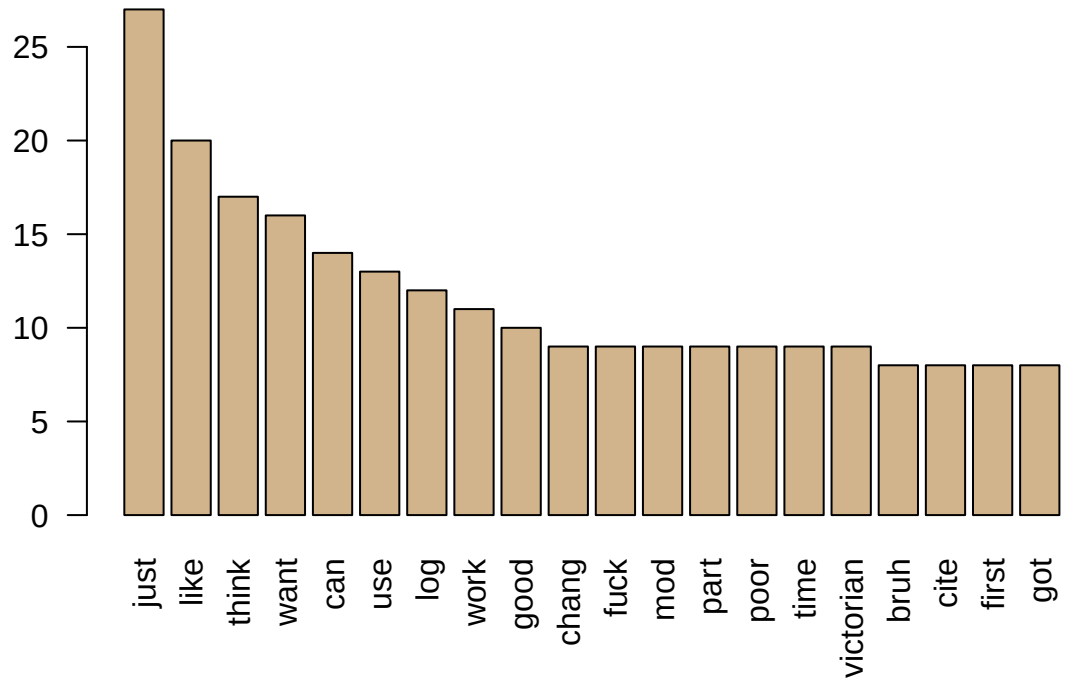
Lastly, Tokenisation

I will then transform the corpra into Term-Document Matrices, where the words are tokenised and term frequency for each document is calculated.

```
marc_tdm = TermDocumentMatrix(cm_corpus)
jemima_tdm = TermDocumentMatrix(cj_corpus)
```

Analysis and Visualisation

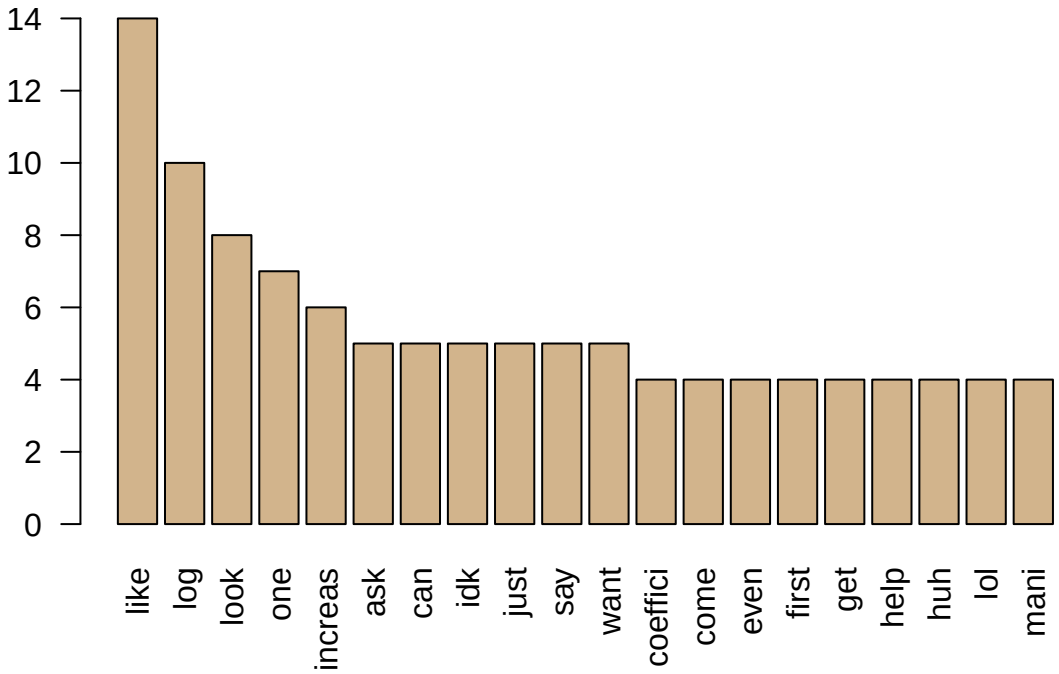
```
# marc
marc_m = as.matrix(marc_tdm)
term_freqm = rowSums(marc_m)
term_freqm = sort(term_freqm, decreasing = T)
barplot(term_freqm[1:20],
        col = "tan",
        las = 2)
```



Term Frequency Plot

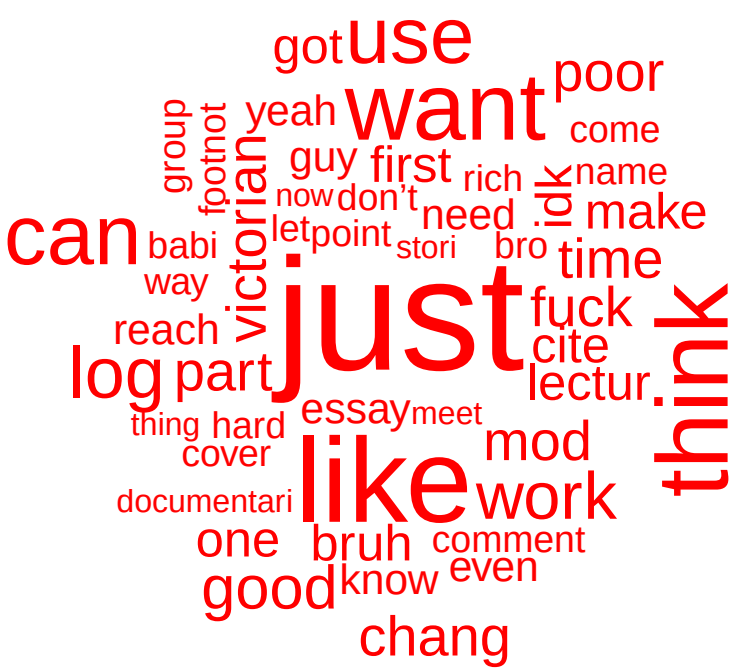
For marc, his most common word is just, like and log.

```
# jemima
jemima_m = as.matrix(jemima_tdm)
term_freqj = jemima_m %>% rowSums() %>% sort(decreasing = T)
barplot(term_freqj[1:20], col = "tan", las = 2)
```



For me, my most common word is log increase and coefficient, and I was asking him about some math equations.

```
# marc
word_freqm = data.frame(term = names(term_freqm), num = term_freqm)
wordcloud(word_freqm$term, word_freqm$num, max.words = 50, colors= 'red', scale = c(5, 0.1))
```



Word Cloud

```
# jemima
word_freqj = data.frame(term = names(term_freqj), num = term_freqj)
wordcloud(word_freqj$term, word_freqj$num, max.words = 50, colors = 'blue', scale = c(5, 0.1))
```

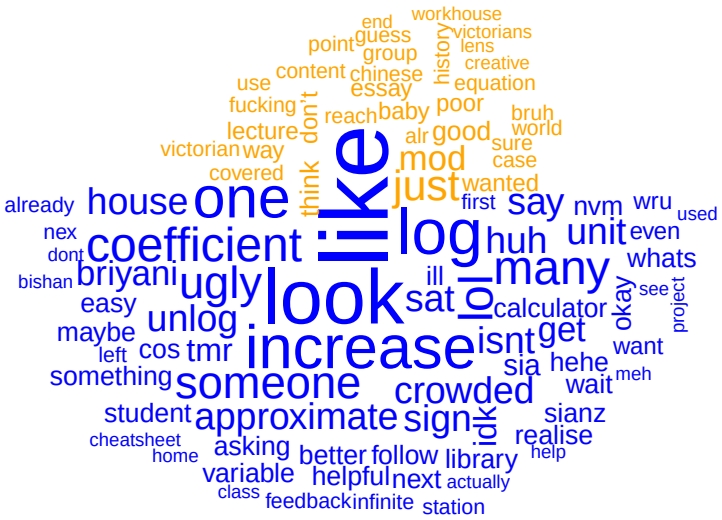
Comparison Visualisations

```
# combine both corpora first
all_marc = paste(msgdata_marc$msgcontent, collapse = " ")
all_jemima = paste(msgdata_jemima$msgcontent, collapse = " ")
all_msg = c(all_marc, all_jemima)

# clean all_msg
all_corpus = VCorpus(VectorSource(all_msg))
all_clean = tm_clean(all_corpus)
all_tdm = TermDocumentMatrix(all_clean)
colnames(all_tdm) = c("marc", 'jemima')
all_m = as.matrix(all_tdm)
```

```
# dissimilar words
comparison.cloud(all_m, colors = c("orange","blue"), max.words = 100, scale = c(3, 0.1), title.size = 2
```

marc



jemima

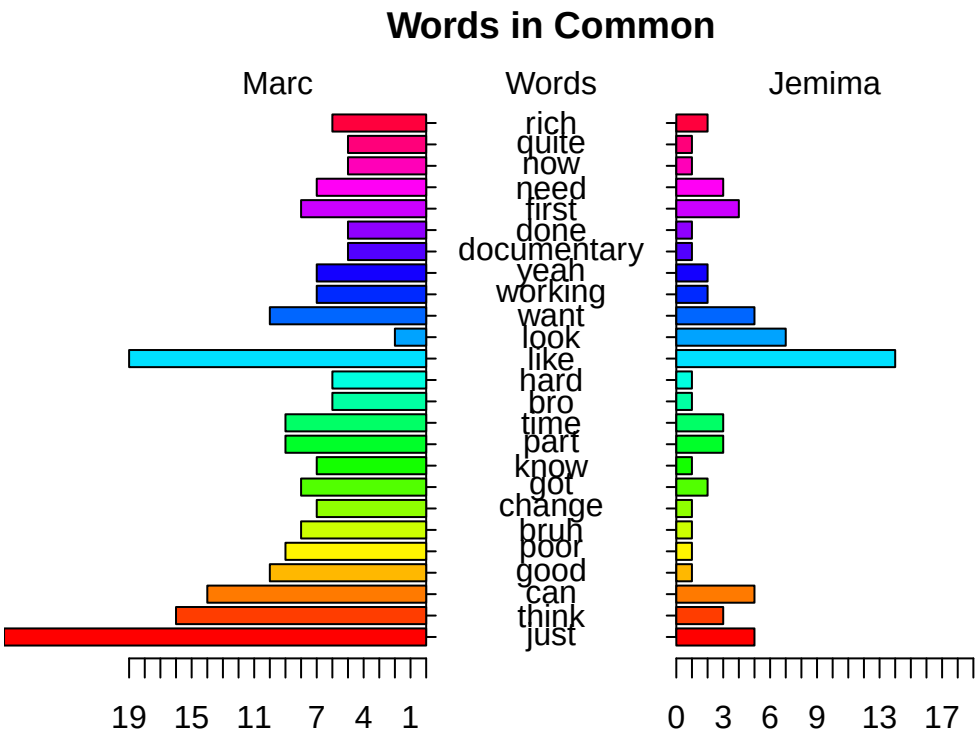
Similarity Word Clouds

```
# similar words
commonality.cloud(all_m, colors = "steelblue", max.words = 50, scale = c(5, 0.1))
```

It seems we both overuse the words "like", "just" and "can".

```
common_words = as.data.frame(subset(all_m, all_m[,1]>0 & all_m[,2] > 0))
common_words = common_words %>% mutate(difference = abs(marc - jemima)) %>% arrange(desc(difference))
top_25 = common_words[1:25,]
top_25 = top_25 %>% select(-difference) %>% mutate(labels = rownames(top_25))

pyramid.plot(top_25$marc, top_25$jemima,
             labels = top_25$labels,
             main = "Words in Common",
             gap = 8, laxlab = NULL,
             raxlab = NULL,
             unit = NULL,
             top.labels = c("Marc", "Words", "Jemima"))
```



Pyramid Plot

```
## 27 27
```

```
## [1] 5.1 4.1 4.1 2.1
```

Bigrams

Perhaps there are certain phrases that we commonly text? Current analysis thus far would not tell me anything about it. While phrases can be any number of words, I will focus on two-word phrases in this exercise.

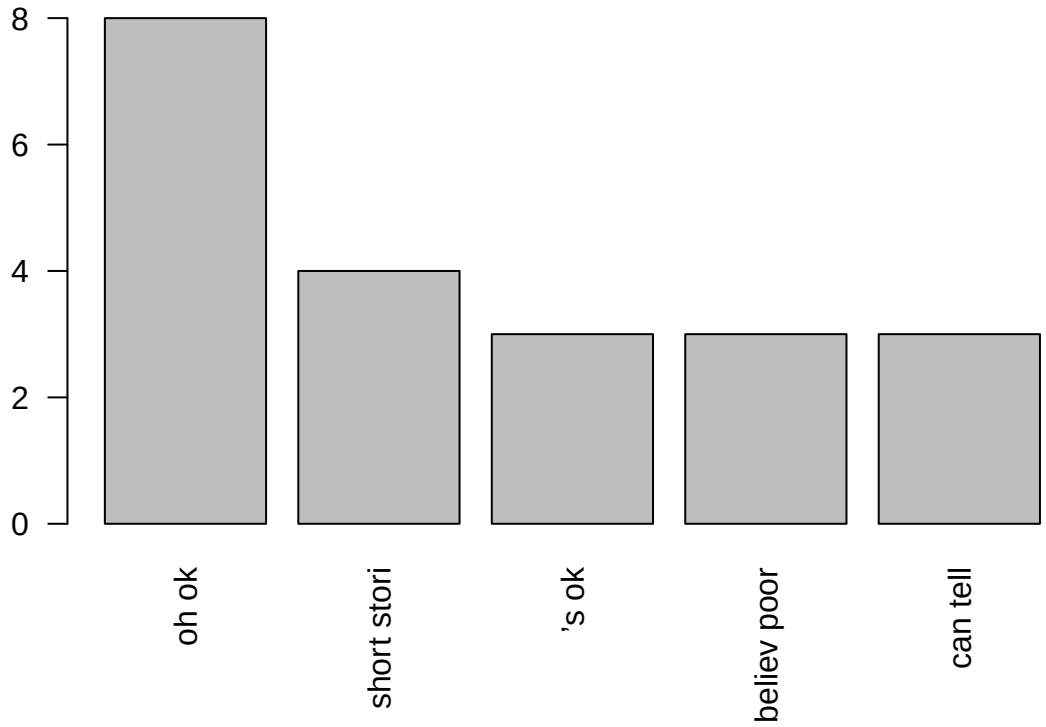
I will first create a tokenizer function that sets the number of words in a term to be 2.

```
# new bigram function
tokeniser = function(x){
  NGramTokenizer(x, Weka_control(min= 2, max = 2))
}

# tokenise
bigram_tdm_m = TermDocumentMatrix(cm_corpus, control= list(tokenize = tokeniser))
bigram_tdm_j = TermDocumentMatrix(cj_corpus, control= list(tokenize = tokeniser))
```

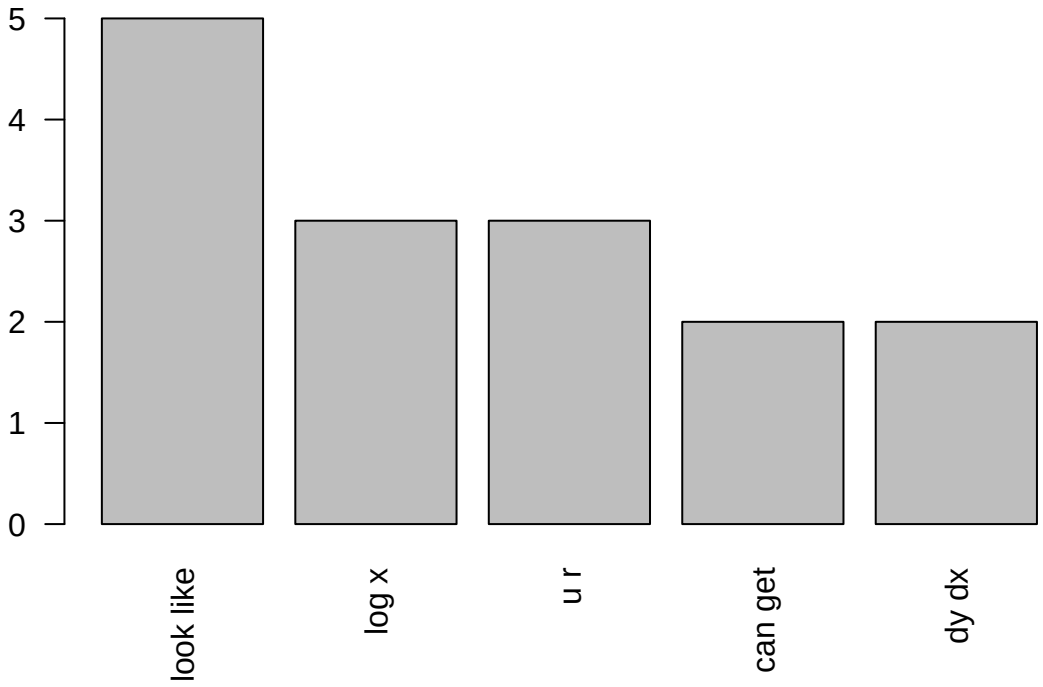
Analysis and Visualisation

```r
# top 5 most frequent bigrams for marc
bigram_m_m = as.matrix(bigram_tdm_m)
bigram_m_freq = rowSums(bigram_m_m)
bigram_m_freq = sort(bigram_m_freq, decreasing = T)
barplot(bigram_m_freq[1:5], las = 2)
```



I believe 's ok should be it's ok. Oh ok is indeed a phrase that he uses frequently when he does not know what to say.

```r
# top 5 most frequent bigrams for jemima
bigram_m_j = as.matrix(bigram_tdm_j)
bigram_j_freq = rowSums(bigram_m_j)
bigram_j_freq = sort(bigram_j_freq, decreasing = T)
barplot(bigram_j_freq[1:5], las = 2)
```



Meanwhile, I use the phrase "look[s] like" very frequently. Maybe I like to point out similarities/conjecture??

Conclusion

It was a fun exercise to see our texting patterns. But I think it would have been more representative to take a few more days of text messages so that I have more data to analyse. It would be interesting to compare how differently I text different people as well.