

Frequency Analysis on N-gram phrases

```
# Read and parse HTML file
library("XML")
```

```
## Warning: package 'XML' was built under R version 3.6.2
```

```
library("RCurl")
```

```
## Warning: package 'RCurl' was built under R version 3.6.2
```

Importing data from website

```
doc.url1<-getURL(
  "https://www.blog.google/technology/safety-security/safety-center-helping-you-stay-safe-online/")
doc.html1 <- htmlParse(doc.url1)
```

```
# Extract all the paragraphs (HTML tag is p, starting at
# the root of the document). Unlist flattens the list to
# create a character vector.
```

```
plain.text1 <- xpathApply(doc.html1, "//div[@class='h-c-grid']/div[@class='uni-paragraph
  h-c-grid__col h-c-grid__col--8 h-c-grid__col-m--6 h-c-grid__col-l--6
  h-c-grid__col--offset-2 h-c-grid__col-m--offset-3 h-c-grid__col-l--offset-3']",
  xmlValue)
cat(paste(plain.text1, collapse = " "),file="Corpus/outfile1.txt")
actual.text1<-paste(readLines("Corpus/outfile1.txt"),collapse=" ")
```

```
## Warning in readLines("Corpus/outfile1.txt"): incomplete final line found on
## 'Corpus/outfile1.txt'
```

```
doc.url2<-getURL(
  "https://www.blog.google/products/google-cloud/new-security-tools-to-help-improve/")
doc.html2 <- htmlParse(doc.url2)
```

```
# Extract all the paragraphs (HTML tag is p, starting at
# the root of the document). Unlist flattens the list to
# create a character vector.
```

```
plain.text2 <- xpathApply(doc.html2, "//div[@class='h-c-grid']/div[@class='uni-paragraph
  h-c-grid__col h-c-grid__col--8 h-c-grid__col-m--6 h-c-grid__col-l--6
  h-c-grid__col--offset-2 h-c-grid__col-m--offset-3 h-c-grid__col-l--offset-3']",
  xmlValue)
cat(paste(plain.text2, collapse = " "),file="Corpus/outfile2.txt")
actual.text2<-paste(readLines("Corpus/outfile2.txt"),collapse=" ")
```

```
## Warning in readLines("Corpus/outfile2.txt"): incomplete final line found on
## 'Corpus/outfile2.txt'
```

```
doc.url3<-getURL("https://www.blog.google/products/google-cloud/bolstering-security-across-google-cloud,")
doc.html3 <- htmlParse(doc.url3)
```

```
# Extract all the paragraphs (HTML tag is p, starting at
# the root of the document). Unlist flattens the list to
# create a character vector.
```

```
plain.text3 <- xpathApply(doc.html3, "//div[@class='h-c-grid']/div[@class='uni-paragraph
      h-c-grid__col h-c-grid__col--8 h-c-grid__col-m--6 h-c-grid__col-l--6
      h-c-grid__col--offset-2 h-c-grid__col-m--offset-3 h-c-grid__col-l--offset-3']",
      xmlValue)
cat(paste(plain.text3, collapse = " "),file="Corpus/outfile3.txt")
actual.text3<-paste(readLines("Corpus/outfile3.txt"),collapse=" ")

## Warning in readLines("Corpus/outfile3.txt"): incomplete final line found on
## 'Corpus/outfile3.txt'
```

Cleaning text data in R

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.6.2
## Loading required package: NLP
```

To find out the path of the destination folder

```
###{r} ##file.choose() ##
```

To retrieve only the txt files from the destination folder

```
folder<- "C:\\Users\\roopa\\Documents\\Corpus"
list.files(path=folder)

## [1] "outfile1.txt"          "outfile2.txt"
## [3] "outfile3.txt"          "R-Graphics4-data viz.pdf"
## [5] "Rgraphics-data viz.pdf"

filelist<-list.files(path=folder,pattern="*.txt")
```

Building the corpus

```
filelist<-paste(folder, "\\ ", filelist, sep="")
typeof(filelist)

## [1] "character"
a<-lapply(filelist, FUN=readLines)

## Warning in FUN(X[[i]], ...): incomplete final line found on 'C:
## \Users\roopa\Documents\Corpus\outfile1.txt'

## Warning in FUN(X[[i]], ...): incomplete final line found on 'C:
## \Users\roopa\Documents\Corpus\outfile2.txt'

## Warning in FUN(X[[i]], ...): incomplete final line found on 'C:
## \Users\roopa\Documents\Corpus\outfile3.txt'

corpus<-lapply(a, FUN=paste, collapse=" ")
```

Cleaning the corpus'

```
# Removing spaces and punctuations
clean.corpus<- gsub(pattern="\\W", replace=" ", corpus)

# Removing digits
clean.corpus<- gsub("\\d", replace=" ", clean.corpus)

# converting all words to lowercase
clean.corpus<- tolower(clean.corpus)

# removing stopwords
clean.corpus<- removeWords(clean.corpus, stopwords("english"))

# removing single length words
# if we want to remove words starting with a
#particular alphabet, say 'd', then write gsub("\\bd\\b", replace=" ", clean.text)

# if we want to remove words starting with a particular alphabet,
#say 'd' of length 1, then write gsub("\\bd\\b{1},replace=" ",clean.text)

# Similarly for removing words starting with a set of alphabets
#(say: d,a and s), write \\b[c('d','a','s')]\\b inside gsub

#Here, we are removing only single letter words
clean.corpus<- gsub("\\b[A-z]\\b{1}",replace=" ", clean.corpus)

#Removing extra whitespaces
clean.corpus<- stripWhitespace(clean.corpus)
```

```
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.6.2
```

```
library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 3.6.2
```

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

Creating wordcloud

```
wordcloud(clean.corpus,random.order=FALSE, scale=c(3,0.5),color=rainbow(3))
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation):  
## transformation drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,  
## tm::stopwords())): transformation drops documents
```



```
real.corpus<-Corpus(VectorSource(clean.corpus))
real.corpus
```

```
## <<SimpleCorpus>>
## Metadata: corpus specific: 1, document level (indexed): 0
## Content: documents: 3
```

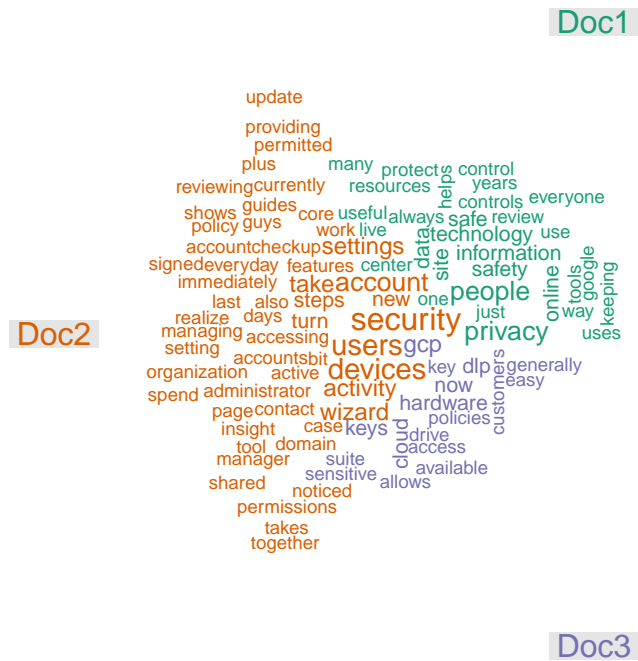
Creating Document Matrix

```
doc.matrix<-TermDocumentMatrix(real.corpus)
doc.matrix
```

```
## <<TermDocumentMatrix (terms: 352, documents: 3)>>
## Non-/sparse entries: 412/644
## Sparsity           : 61%
## Maximal term length: 17
## Weighting          : term frequency (tf)
```

```
matrixformat<-as.matrix(doc.matrix)
```

```
colnames(matrixformat)<-c("Doc1", "Doc2", "Doc3")
comparison.cloud(matrixformat,random.order=FALSE,scale=c(1,.5),
  max.words = 100,title.size=1,match.colors=TRUE)
```



Sentiment analysis

```
#Identify working directory and copy paste the sentiments words to text files
#and place it in that directory
getwd()
```

```
## [1] "C:/Users/roopa/Documents"
```

```
pos.text<-scan('positive_sentiments.txt',what='character',comment.char=';')
neg.text<-scan('negative_sentiments.txt', what='character',comment.char=';')
```

```
#to convert to bag or list of words
```

```
clean.corpus.bag <- str_split(clean.corpus,pattern="\\s+")
clean.corpus.bag
```

```
## [[1]]
## [1] "" "making" "technology" "everyone"
## [5] "means" "protecting" "everyone" "uses"
## [9] "years" "google" "building" "useful"
## [13] "products" "help" "make" "people"
## [17] "lives" "easier" "beginning" "ve"
## [21] "always" "known" "people" "use"
## [25] "services" "re" "trusting" "us"
## [29] "information" "job" "protect" "today"
## [33] "started" "rolling" "newly" "expanded"
## [37] "google" "safety" "center" "ve"
```

##	[41]	"updated"	"resources"	"pulled"	"even"
##	[45]	"information"	"one"	"site"	"dedicated"
##	[49]	"educating"	"empowering"	"people"	"important"
##	[53]	"topics"	"like"	"data"	"security"
##	[57]	"privacy"	"controls"	"online"	"protections"
##	[61]	"users"	"site"	"will"	"available"
##	[65]	"languages"	"coming"	"weeks"	"safety"
##	[69]	"center"	"just"	"one"	"way"
##	[73]	"inform"	"people"	"keep"	"personal"
##	[77]	"information"	"private"	"safe"	"give"
##	[81]	"control"	"links"	"many"	"easy"
##	[85]	"use"	"privacy"	"controls"	"people"
##	[89]	"can"	"choose"	"settings"	"right"
##	[93]	"features"	"helpful"	"security"	"tips"
##	[97]	"care"	"keeping"	"safe"	"whenever"
##	[101]	"re"	"online"	"just"	"google"
##	[105]	"help"	"families"	"better"	"manage"
##	[109]	"technology"	"provides"	"useful"	"resources"
##	[113]	"tools"	"teach"	"kids"	"digital"
##	[117]	"safety"	"citizenship"	"helping"	"people"
##	[121]	"manage"	"privacy"	"security"	"integral"
##	[125]	"everything"	"years"	"ve"	"created"
##	[129]	"many"	"tools"	"always"	"improving"
##	[133]	"upon"	"re"	"control"	"google"
##	[137]	"account"	"gives"	"access"	"settings"
##	[141]	"safeguard"	"data"	"privacy"	"privacy"
##	[145]	"checkup"	"helps"	"quickly"	"review"
##	[149]	"adjust"	"data"	"google"	"uses"
##	[153]	"personalize"	"experience"	"activity"	"helps"
##	[157]	"review"	"delete"	"activity"	"data"
##	[161]	"connected"	"account"	"technology"	"continues"
##	[165]	"change"	"way"	"live"	"work"
##	[169]	"play"	"commitment"	"keeping"	"safe"
##	[173]	"secure"	"online"	"grows"	"site"
##	[177]	"latest"	"example"	"live"	"responsibility"
##	[181]	"protect"	"		
##					
##	[[2]]				
##	[1]	"	"manager"	"realize"	"spend"
##	[5]	"lot"	"time"	"managing"	"devices"
##	[9]	"applications"	"security"	"settings"	"everyone"
##	[13]	"organization"	"make"	"job"	"bit"
##	[17]	"easier"	"today"	"re"	"announcing"
##	[21]	"new"	"security"	"tools"	"help"
##	[25]	"suite"	"users"	"take"	"control"
##	[29]	"security"	"online"	"new"	"devices"
##	[33]	"activity"	"dashboard"	"gives"	"users"
##	[37]	"additional"	"insight"	"devices"	"accessing"
##	[41]	"google"	"account"	"page"	"shows"
##	[45]	"comprehensive"	"view"	"devices"	"active"
##	[49]	"account"	"last"	"days"	"currently"
##	[53]	"signed"	"case"	"suspicious"	"activity"
##	[57]	"noticed"	"setting"	"immediately"	"take"
##	[61]	"steps"	"secure"	"account"	"change"

## [65]	"password"	"also"	"launching"	"security"
## [69]	"wizard"	"google"	"work"	"accounts"
## [73]	"security"	"wizard"	"guides"	"users"
## [77]	"steps"	"can"	"take"	"turn"
## [81]	"adjust"	"security"	"features"	"like"
## [85]	"providing"	"contact"	"info"	"account"
## [89]	"recovery"	"domain"	"security"	"policy"
## [93]	"allows"	"reviewing"	"recent"	"account"
## [97]	"activity"	"account"	"permissions"	"plus"
## [101]	"takes"	"minutes"	"users"	"update"
## [105]	"settings"	"tool"	"prioritizes"	"administrator"
## [109]	"settings"	"security"	"features"	"end"
## [113]	"users"	"permitted"	"turn"	"access"
## [117]	"wizard"	"co"	"accountcheckup"	"security"
## [121]	"cloud"	"shared"	"responsibility"	"keeping"
## [125]	"company"	"information"	"secure"	"core"
## [129]	"everyday"	"making"	"users"	"aware"
## [133]	"security"	"settings"	"activity"	"devices"
## [137]	"can"	"work"	"together"	"stay"
## [141]	"step"	"ahead"	"bad"	"guys"
## [145]	" "			
##				
## [[3]]				
## [1]	" "	"san"	"francisco"	
## [4]	"today"	"google"	"cloud"	
## [7]	"next"	"launched"	"following"	
## [10]	"new"	"features"	"google"	
## [13]	"cloud"	"platform"	"gcp"	
## [16]	"suite"	"designed"	"help"	
## [19]	"safeguard"	"company"	"assets"	
## [22]	"prevent"	"disruption"	"business"	
## [25]	"identity"	"aware"	"proxy"	
## [28]	"iap"	"gcp"	"now"	
## [31]	"beta"	"allows"	"manage"	
## [34]	"granular"	"access"	"applications"	
## [37]	"running"	"gcp"	"based"	
## [40]	"risk"	"rather"	"nothing"	
## [43]	"approach"	"vpn"	"access"	
## [46]	"provides"	"secure"	"application"	
## [49]	"access"	"anywhere"	"access"	
## [52]	"determined"	"user"	"identity"	
## [55]	"group"	"iap"	"easy"	
## [58]	"deploy"	"can"	"integrated"	
## [61]	"phishing"	"resistant"	"security"	
## [64]	"keys"	"data"	"loss"	
## [67]	"prevention"	"dlp"	"api"	
## [70]	"gcp"	"now"	"beta"	
## [73]	"lets"	"scan"	"sensitive"	
## [76]	"data"	"types"	"can"	
## [79]	"identify"	"redact"	"sensitive"	
## [82]	"data"	"dlp"	"deep"	
## [85]	"content"	"analysis"	"help"	
## [88]	"ensure"	"matter"	"want"	
## [91]	"keep"	"safe"	"credit"	

## [94]	"cards"	"account"	"numbers"
## [97]	"know"	"protected"	"level"
## [100]	"want"	"dlp"	"api"
## [103]	"gcp"	"joins"	"dlp"
## [106]	"gmail"	"drive"	"allowing"
## [109]	"admins"	"write"	"policies"
## [112]	"manage"	"sensitive"	"data"
## [115]	"ways"	"aren"	"possible"
## [118]	"cloud"	"key"	"management"
## [121]	"service"	"gcp"	"now"
## [124]	"generally"	"available"	"allows"
## [127]	"generate"	"use"	"rotate"
## [130]	"destroy"	"symmetric"	"encryption"
## [133]	"keys"	"use"	"cloud"
## [136]	"gives"	"customers"	"ability"
## [139]	"manage"	"encryption"	"keys"
## [142]	"multi"	"tenant"	"cloud"
## [145]	"service"	"without"	"need"
## [148]	"maintain"	"premise"	"key"
## [151]	"management"	"system"	"hardware"
## [154]	"security"	"module"	"security"
## [157]	"key"	"enforcement"	"ske"
## [160]	"gcp"	"suite"	"now"
## [163]	"generally"	"available"	"allows"
## [166]	"require"	"security"	"keys"
## [169]	"used"	"two"	"step"
## [172]	"verification"	"factor"	"stronger"
## [175]	"authentication"	"whenever"	"user"
## [178]	"signs"	"suite"	"accesses"
## [181]	"gcp"	"resource"	"ske"
## [184]	"easy"	"admins"	"easy"
## [187]	"users"	"hard"	"phishers"
## [190]	"google"	"vault"	"google"
## [193]	"drive"	"team"	"drives"
## [196]	"google"	"groups"	"now"
## [199]	"generally"	"available"	"ediscovey"
## [202]	"compliance"	"solution"	"suite"
## [205]	"vault"	"allows"	"customers"
## [208]	"set"	"retention"	"policies"
## [211]	"place"	"legal"	"holds"
## [214]	"perform"	"searches"	"across"
## [217]	"drive"	"gmail"	"hangouts"
## [220]	"groups"	"export"	"search"
## [223]	"results"	"support"	"legal"
## [226]	"compliance"	"requirementstitan"	"google"
## [229]	"purpose"	"built"	"chip"
## [232]	"establish"	"hardware"	"root"
## [235]	"trust"	"machines"	"peripherals"
## [238]	"cloud"	"infrastructure"	"allowing"
## [241]	"us"	"securely"	"identify"
## [244]	"authenticate"	"legitimate"	"access"
## [247]	"hardware"	"level"	"purpose"
## [250]	"built"	"hardware"	"titan"
## [253]	"part"	"google"	"layered"


```
## [256] "security"      "architecture"  "spanning"
## [259] "physical"      "security"      "data"
## [262] "centers"       "secure"        "boot"
## [265] "across"        "hardware"      "software"
## [268] "operational"   "security"      "baking"
## [271] "security"      "everything"    "offering"
## [274] "innovative"    "capabilities"  "build"
## [277] "upon"          "secure"        "foundation"
## [280] "create"        "many"          "different"
## [283] "layers"        "prevent"       "defend"
## [286] "attacks"       "implement"     "enterprise"
## [289] "security"      "policies"      "customers"
## [292] "can"           "feel"          "confident"
## [295] "partnering"    "us"            "achieve"
## [298] "business"      "goals"         ""
```

```
#Finding matching positive word count
```

```
lapply(clean.corpus.bag, function(x){sum(!is.na(match(x,pos.text)))})
```

```
## [[1]]
## [1] 23
##
## [[2]]
## [1] 8
##
## [[3]]
## [1] 20
```

```
#Finding matching negative word count
```

```
lapply(clean.corpus.bag, function(x){sum(!is.na(match(x,neg.text)))})
```

```
## [[1]]
## [1] 0
##
## [[2]]
## [1] 3
##
## [[3]]
## [1] 12
```

```
#Finding total sentiment score
```

```
score<-unlist(lapply(clean.corpus.bag,
  function(x){sum(!is.na(match(x,pos.text)))- sum(!is.na(match(x,neg.text)))}))
score
```

```
## [1] 23 5 8
```

```
# mean of sentiment scores
```

```
mean(score)
```

```
## [1] 12
```

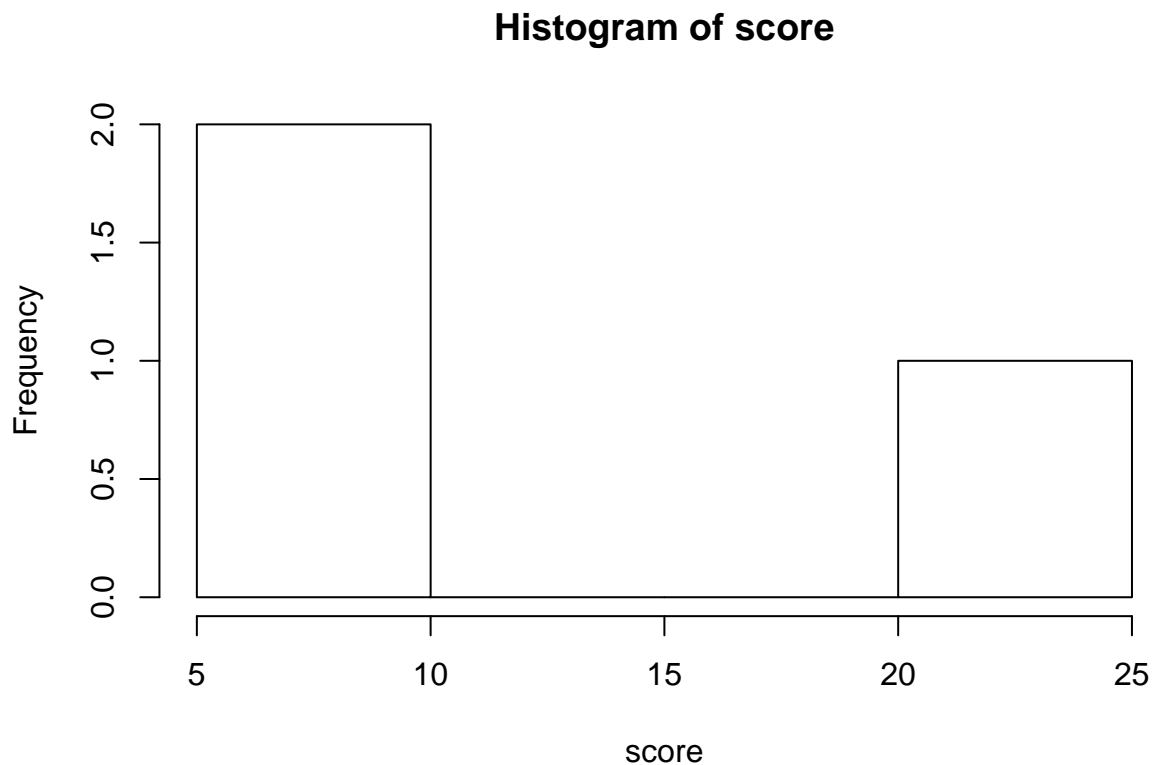
```
# standard deviation of sentiment score
```

```
sd(score)
```

```
## [1] 9.643651
```

```
# Histogram for sentiment scores
```

```
hist(score)
```



Unigram Tokenization

```
library(rJava)
```

```
##  
## Attaching package: 'rJava'  
## The following object is masked from 'package:RCurl':  
##  
##      clone
```

```
library(RWeka)
```

```
## Warning: package 'RWeka' was built under R version 3.6.2
```

```
library(ggplot2)
```

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

```
# Use Weka's n-gram tokenizer to create a TDM  
# that uses as terms the unigrams that appear in the corpus.
```

```
Unigram_Tokenizer <- function(x){
```

```

  NGramTokenizer(x, Weka_control(min=1, max=1))
}

#create a matrix
tdm <- TermDocumentMatrix(real.corpus, control = list(tokenize = Unigram_Tokenizer))

# Extract the frequency of each unigram and analyse the twenty most frequent ones.

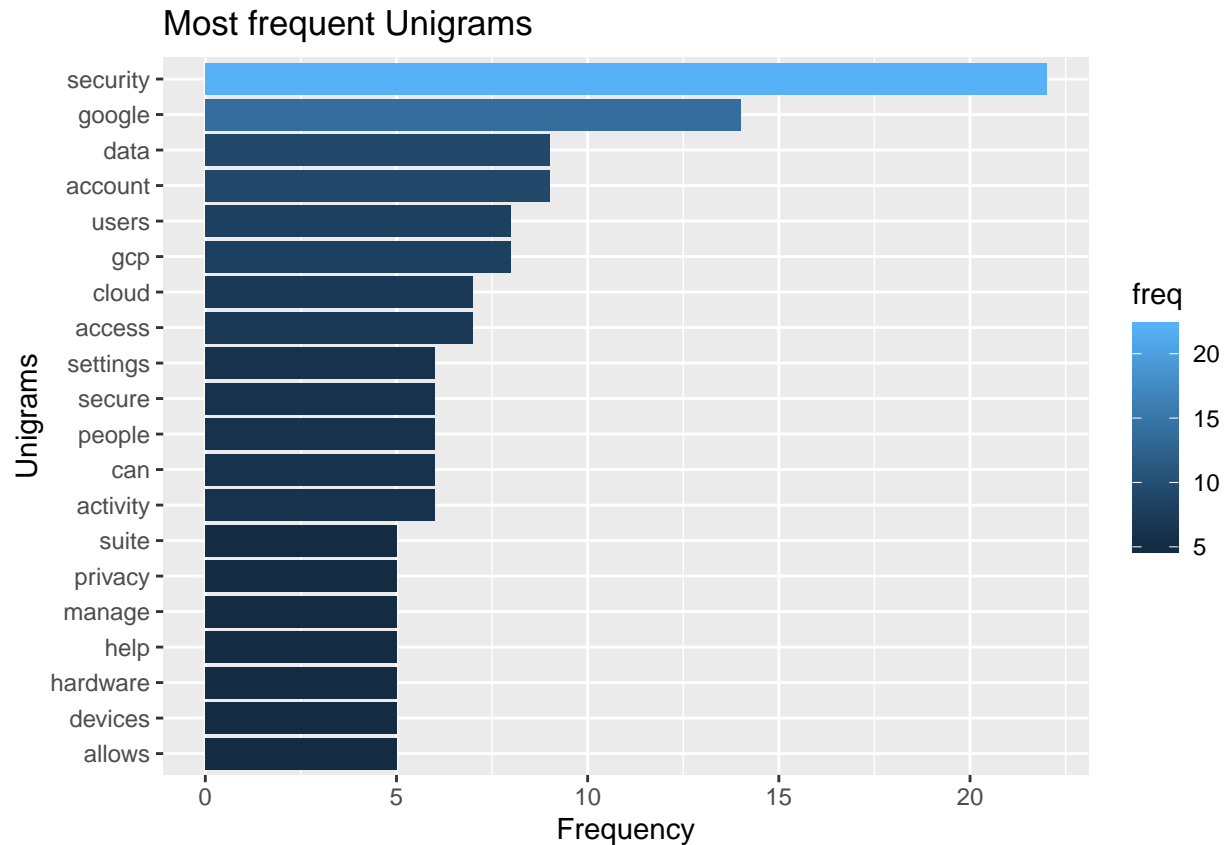
freq <-sort(rowSums(as.matrix(tdm)),decreasing = TRUE)
freq.df <- data.frame(word=names(freq), freq=freq)
head(freq.df, 20)

##           word freq
## security security 22
## google    google 14
## account   account 9
## data      data   9
## users     users   8
## gcp       gcp     8
## access    access  7
## cloud     cloud   7
## activity  activity 6
## can       can     6
## people    people  6
## secure    secure   6
## settings  settings 6
## help      help     5
## manage    manage   5
## privacy   privacy  5
## allows    allows   5
## devices   devices  5
## suite     suite    5
## hardware  hardware 5

# Plotting the Bigram model

ggplot(head(freq.df,20), aes(reorder(word,freq), freq, fill=freq)) +
  geom_bar(stat="identity") + coord_flip() +
  xlab("Unigrams") + ylab("Frequency") + scale_fill_continuous(type="gradient")+
  ggtitle("Most frequent Unigrams")

```



N gram analysis

Bigram Tokenization

```
library(rJava)
library(RWeka)
library(ggplot2)

# Use Weka's n-gram tokenizer to create a TDM
# that uses as terms the bigrams that appear in the corpus.

Bigram_Tokenizer <- function(x){
  NGramTokenizer(x, Weka_control(min=2, max=2))
}

#create a matrix
tdm <- TermDocumentMatrix(real.corpus, control = list(tokenize = Bigram_Tokenizer))

# Extract the frequency of each bigram and analyse the twenty most frequent ones.

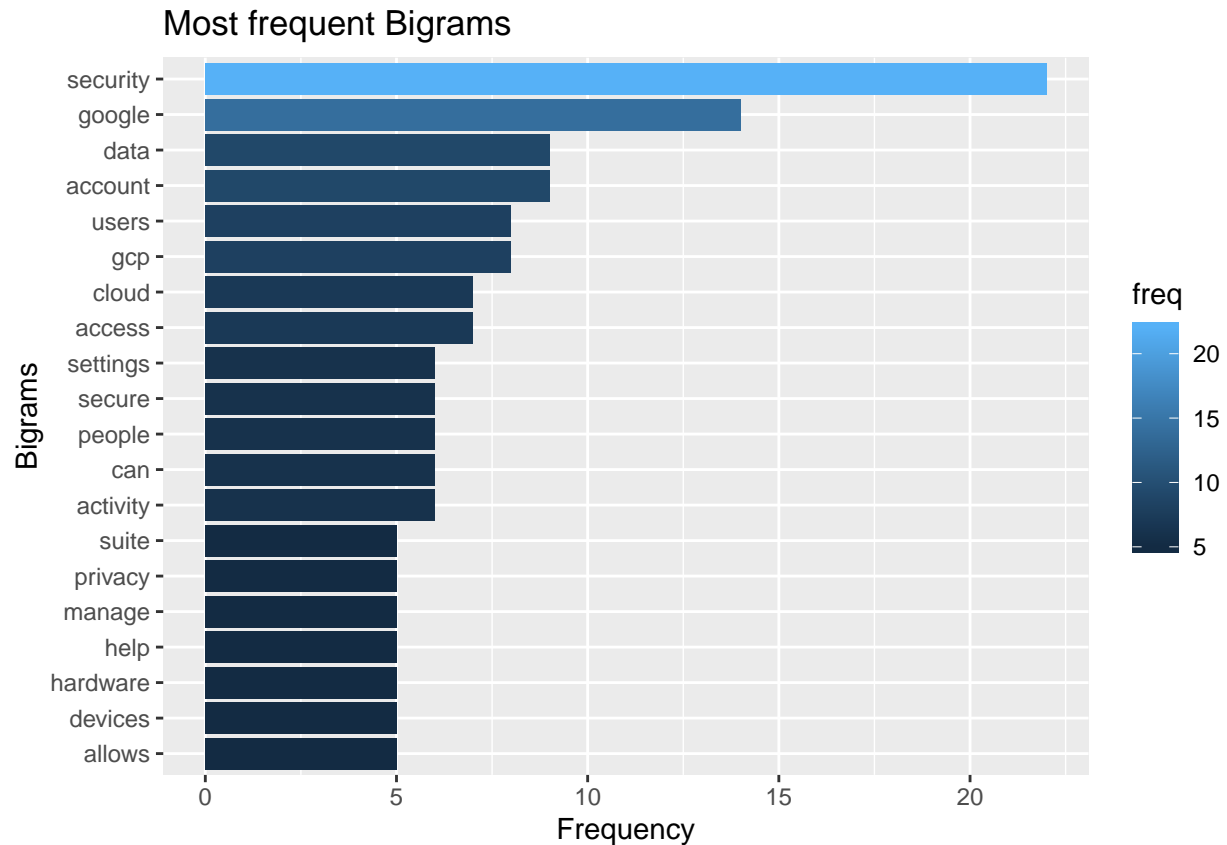
freq <- sort(rowSums(as.matrix(tdm)), decreasing = TRUE)
freq.df <- data.frame(word=names(freq), freq=freq)
```

```
head(freq.df, 20)
```

```
##           word freq
## security security 22
## google    google 14
## account   account 9
## data      data    9
## users     users   8
## gcp       gcp     8
## access    access  7
## cloud     cloud   7
## activity  activity 6
## can       can     6
## people    people  6
## secure    secure  6
## settings  settings 6
## help      help    5
## manage    manage  5
## privacy   privacy 5
## allows    allows  5
## devices   devices 5
## suite     suite   5
## hardware  hardware 5
```

```
# Plotting the Bigram model
```

```
ggplot(head(freq.df,20), aes(reorder(word,freq), freq, fill=freq)) +  
  geom_bar(stat="identity") + coord_flip() +  
  xlab("Bigrams") + ylab("Frequency") + scale_fill_continuous(type="gradient")+  
  scale_colour_hue(h = c(90, 180))+  
  ggtitle("Most frequent Bigrams")
```



Trigram Tokenizer

```
# Use Weka's n-gram tokenizer to create a TDM
# that uses as terms the trigrams that appear in the corpus.

Trigram_Tokenizer <- function(x){
  NGramTokenizer(x, Weka_control(min=3, max=3))
}

#create a matrix
tdm <- TermDocumentMatrix(real.corpus, control = list(tokenize = Trigram_Tokenizer))

# Extract the frequency of each trigram and analyse the twenty most frequent ones.

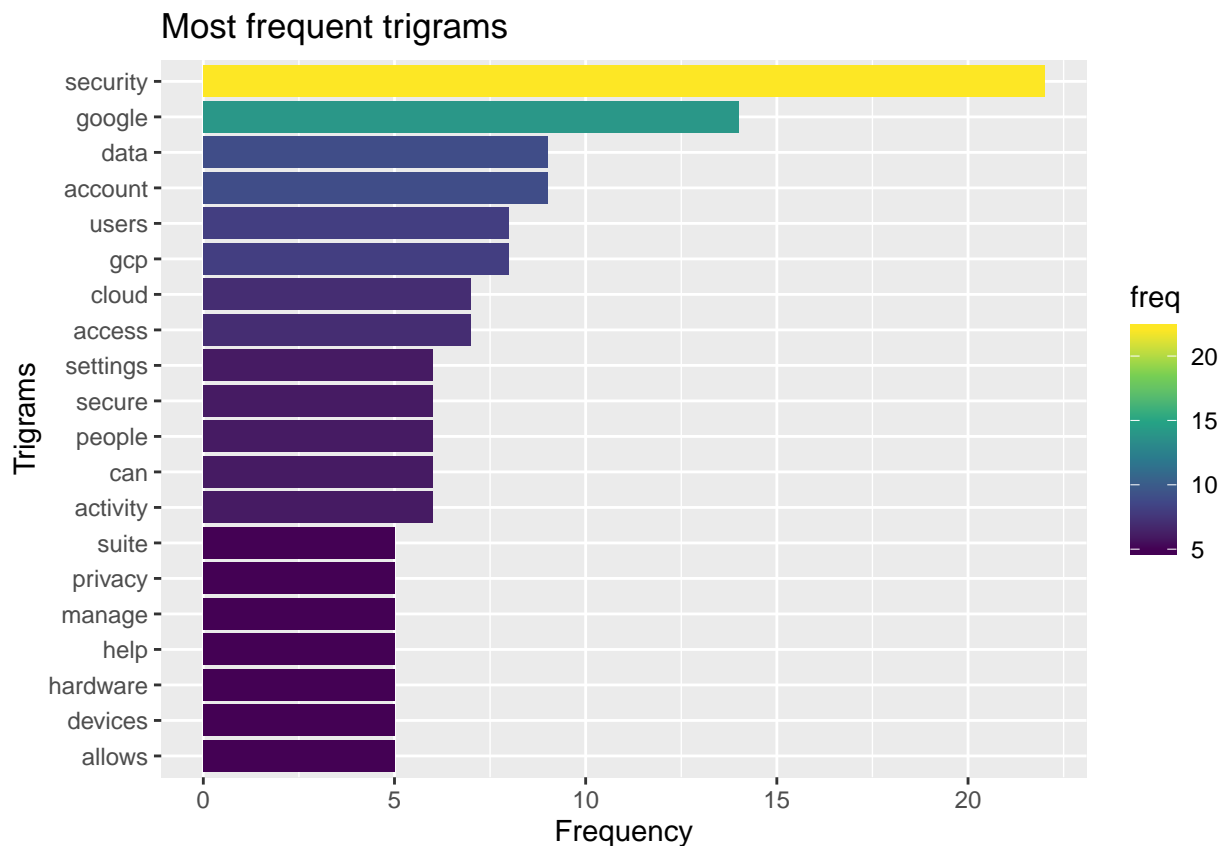
freq <- sort(rowSums(as.matrix(tdm)), decreasing = TRUE)
freq.df <- data.frame(word=names(freq), freq=freq)
head(freq.df, 20)

##           word freq
## security security 22
## google    google 14
## account   account  9
## data      data    9
```

```
## users      users      8
## gcp        gcp        8
## access     access     7
## cloud      cloud      7
## activity   activity   6
## can        can        6
## people     people     6
## secure     secure     6
## settings   settings   6
## help       help       5
## manage     manage     5
## privacy    privacy    5
## allows     allows     5
## devices    devices    5
## suite      suite      5
## hardware   hardware   5
```

```
# Plotting the Trigram model
```

```
ggplot(head(freq.df,20), aes(reorder(word,freq), freq,fill=freq)) +
  geom_bar(stat="identity") + coord_flip() +
  xlab("Trigrams") + ylab("Frequency") + scale_fill_continuous(type="viridis")+
  scale_colour_hue(h = c(90, 180))+
  ggtitle("Most frequent trigrams")
```



```
TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
tdm.trigram = TermDocumentMatrix(real.corpus,
```

```
control = list(tokenize = TrigramTokenizer))

freq = sort(rowSums(as.matrix(tdm.trigram)),decreasing = TRUE)
freq.df = data.frame(word=names(freq), freq=freq)
head(freq.df, 20)
```

```
##           word freq
## security security 22
## google    google 14
## account   account 9
## data      data   9
## users     users   8
## gcp       gcp     8
## access    access  7
## cloud     cloud   7
## activity  activity 6
## can       can     6
## people    people  6
## secure    secure  6
## settings  settings 6
## help      help    5
## manage    manage  5
## privacy   privacy 5
## allows    allows  5
## devices   devices 5
## suite     suite   5
## hardware  hardware 5
```