Week 2

C. Kalinowski

3/1/2021

Task 2 - Exploratory Data Analysis

Exploratory Analysis and Word Frequencies

Based on tokenization, we can explore the words in the corpus. A possible function would be to create a clean tokenized list of vectors, unlist the contents, and return a frequency table as a data table.

```
tokenFreq<-function(x){
      ## creates data table of tokens from input text line x
      ## runs input x through cleanToken to get token list
      ## unlists token list to get single list
      ## creates data.table and calculates frequencies
      ## returns data.table

   words<-cleanToken(x)
   wordslist<-as.data.table(unlist(words))
   wordslist<-wordslist[,.(.N),keyby=wordslist]
   colnames(wordslist)<-c("token","freq")
   wordslist
}

sample1<-sampleReader("blogs",5)
sample1[4]
```

```
[1] "so anyways, i am going to share some home decor inspiration that i have been storing in my folder
```

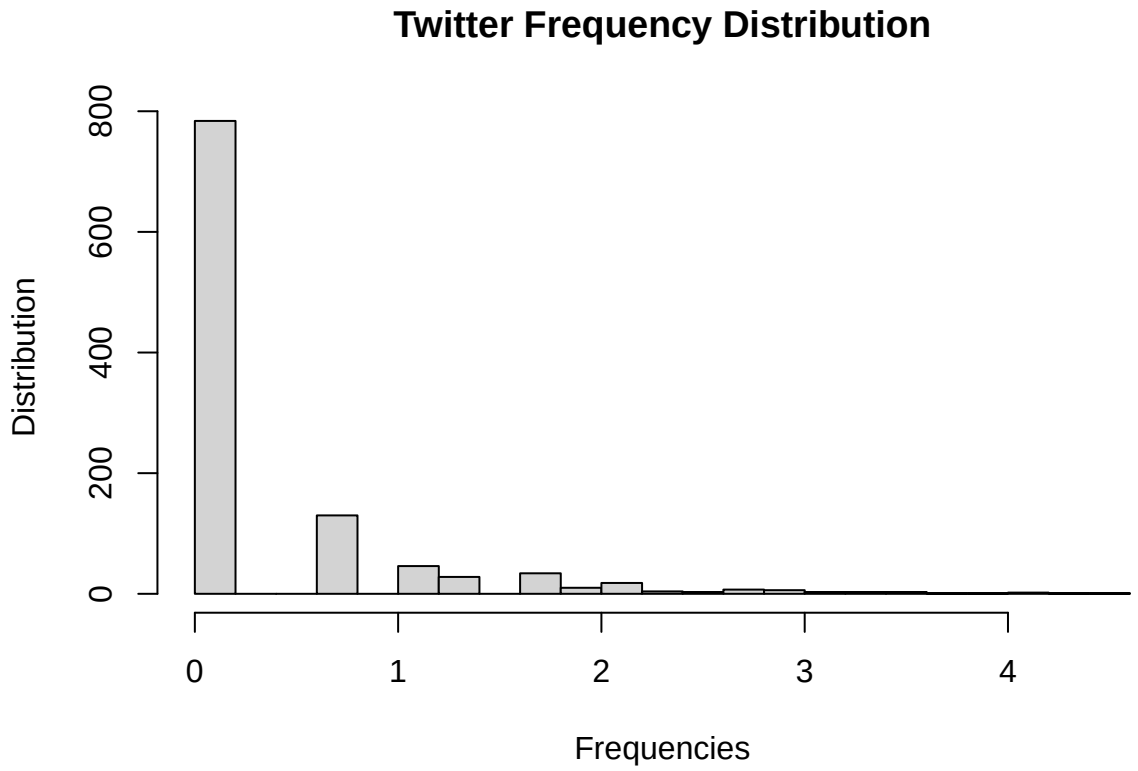
```
tokens1<-tokenFreq(sample1)
head(tokens1)
```

```
    token freq
1:      a    3
2:  after    2
3:    all    3
4: almost    1
5:   also    1
6:     am    1
```

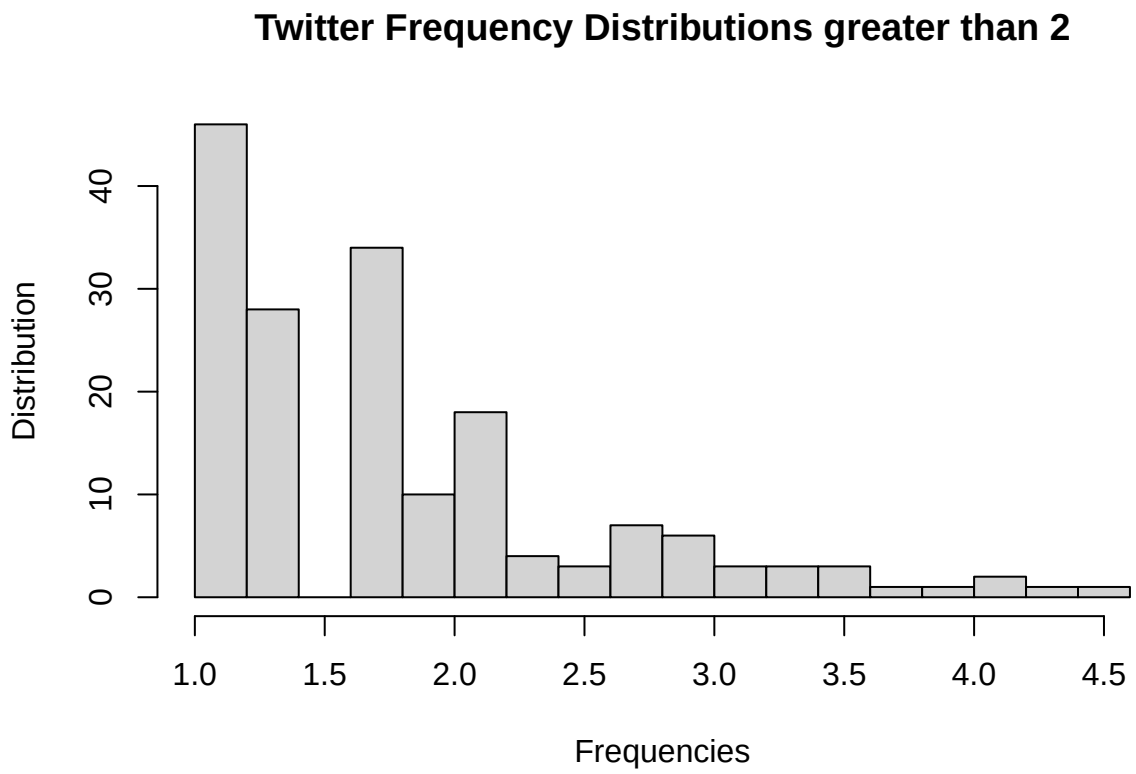
In each data set, a random sample of 200 lines has the following frequency distribution:

```
set.seed=322021
twitter<-sampleReader("twitter")
sampleTwitter<-sample(twitter,size=200,replace=F)
rm(twitter)
blog<-sampleReader("blogs")
sampleBlog<-sample(blog,size=200,replace=F)
```

```
rm(blog)
news<-sampleReader("news")
sampleNews<-sample(news,size=200,replace=F)
rm(news)
twitterToken<-tokenFreq(sampleTwitter)
blogToken<-tokenFreq(sampleBlog)
newsToken<-tokenFreq(sampleNews)
hist(log(twitterToken$freq),main="Twitter Frequency Distribution",xlab="Frequencies",
   ylab="Distribution",breaks=20)
```




```
hist(log(subset(twitterToken,freq>2)$freq),main="Twitter Frequency Distributions greater than 2",
   xlab="Frequencies",ylab="Distribution",breaks=20)
```



The words with the highest counts in each of the 3 English corpora are:

```
# BLOGS:
head(blogToken[order(-freq)])
```

```
   token freq
1:   the  389
2:   and  246
3:    to  234
4:     a  197
5:    of  177
6:     i  163
```

```
# NEWS:
head(newsToken[order(-freq)])
```

```
   token freq
1:   the  355
2:     a  164
3:   and  162
4:    to  150
5:    of  146
6:    in  124
```

```
# TWITTER:
head(twitterToken[order(-freq)])
```

```
   token freq
1:   the   99
```

```
2:     i   77
3:    to   64
4:   you   57
5:     a   51
6:   and   39
```

N-Gram Frequency

An easy way to create N-grams is to paste together token vectors.

```
ngrammer<-function(x,y){
      ## Creates list of y-grams from input tokenized list x
      ## Checks whether y is greater than length of x, else returns NULL
      ## Creates y vectors with stepped start and end points
      ## Vector 1 starts at 1 and ends at length of x - y
      ## Vector y starts at y and ends at end of x
      ## Binds vectors into data frame
      ## Pastes rows of data frame with space separator
      ## Returns list of pasted rows

   ngramMatrix<-NULL
   size<-length(x)
   if(size<=y){
      return()
   }

   ngramMatrix<-matrix(nrow=(size-y+1),ncol=0)
   for (i in (1:y)){
      tokenlist<-x[i:(size-y+i)]
      ngramMatrix<-cbind(ngramMatrix,tokenlist)
   }

   df_args <- c(as.data.table(ngramMatrix), sep=" ")
   df_args<-as.data.table(do.call(paste, df_args))
}


sample1[4]
```

```
[1] "so anyways, i am going to share some home decor inspiration that i have been storing in my folder
```

```
head(ngrammer(cleanToken(sample1[4]),4),10)
```

```
                          V1
 1:             so anyways i am
 2:          anyways i am going
 3:               i am going to
 4:           am going to share
 5:         going to share some
 6:          to share some home
 7:       share some home decor
 8: some home decor inspiration
 9: home decor inspiration that
10:    decor inspiration that i
```

Task 3 - Modeling

The first task for modeling is creating ngram frequencies from the test sets and assigning them probabilities. Each of the 3 English corpora were split 60-20-20 into training, validation, and test sets.

```
ngramtable<-function(x,y){
      ## Creates ngram frequency table of n-1 as rows and last word as column
      ## x is tokenized list, y is ngram number
      ## If y>2, send x to ngrammer function with y-1 for row generator
      ## If y=2, create vector using x from 1 to end-1 as first word generator
      ## Create vector using x from y to end as last word generator
      ## Create data frame with ngrammer or first word as first column
      ## and last word as second column
      ## If z, create and return frequency table of ngrammer by last word
      ## If not z, return data table

   if(y>2){
      ngrams<-ngrammer(x,(y-1))
   }
   else{
      ngrams<-x[1:(length(x)-1)]
   }
   singles<-x[y:(length(x))]
   ngramtable<-data.table(ngrams[1:length(singles)],
                  single = singles)

   ngramtable<-ngramtable[,.(freq=.N),keyby = .(phrase=V1,single)]
   ngramtable
}

sampletable<-ngramtable(cleanToken(sample1[4]),4)
head(sampletable[order(-freq)])
```

```
                  phrase single freq
1:     all these amazing images    1
2:           am going to  share    1
3: amazing images stored   away    1
4:          anyways i am  going    1
5:         away ready to   come    1
6:       been storing in     my    1
```

For multi-line corpora, using lapply gets the correct output

```
trainTwitter[1:2]
```

```
[1] "they've decided its more fun if I don't."
[2] "So Tired D; Played Lazer Tag & Ran A LOT D; Ughh Going To Sleep Like In 5 Minutes ;)"
```

```
token<-lapply(trainTwitter[1:2],cleanToken)
grams<-lapply(token,ngrammer,y=4)
gramDT<-lapply(token[1:2],ngramtable,y=4)
gramList<-do.call("rbind",gramDT)
gramList
```

```
             phrase  single freq
 1: decided its more     fun    1
 2:         fun if i     don    1
```

```
 3:         if i don       t    1
 4:     its more fun      if    1
 5:      more fun if       i    1
 6:  they ve decided     its    1
 7:   ve decided its    more    1
 8:          a lot d    ughh    1
 9:   d played lazer     tag    1
10:     d ughh going      to    1
11:   going to sleep    like    1
12:    lazer tag ran       a    1
13:       lot d ughh   going    1
14: played lazer tag     ran    1
15:        ran a lot       d    1
16:    sleep like in minutes    1
17:       so tired d  played    1
18:        tag ran a     lot    1
19:   tired d played   lazer    1
20:    to sleep like      in    1
21:    ughh going to   sleep    1
              phrase  single freq
```

Frequencies

Unigrams

```
# Clean and tokenize the twitter data
twitterClean<-lapply(trainTwitter,cleanToken)
# unlist the token list to create a single list
twitter1gramList<-unlist(twitterClean)
# create a frequency table
twitter1gramFreq<-table(twitter1gramList)
# create a probability table
twitter1gramProb<-round((twitter1gramFreq/length(twitter1gramList)*100),6)
```

Bigrams

```
library(dplyr)
# create 2gram dfs
twitter2grams<-lapply(twitterClean,ngramtable,y=2,z=FALSE)
# combine 2gram dfs
twitter2gramList<-do.call("rbind",twitter2grams)
twitter2gramList<-as.data.frame(twitter2gramList)
# create a frequency chart
twitter2gramFreq<-table(twitter2gramList[,1],twitter2gramList[,2])
```

**********normalize frequencies by unigrams********** frequency of “a b” / frequency of a