```
#before running, select FILE, save with encoding , UTP-8 please.
#1 load in data set
#create a new working directory that is directly pointing to the folder contains
#blogs, news and tweets( english data)

rm(list=ls())
getwd()
data_path<-paste(getwd(),"/final/en_US", sep="")
setwd(data_path)
#package I use to do this projectï¼
library(ggplot2)
library(tm)#text mining
library(NLP) # for natural language processing
library(stringr) # package for handling string in R
library(R.utils) # ultils to count lines
library(SnowballC) # for steming words.
library(RWeka) #for n-gram model
library(ngram) # for n-grams model
library(qdap) # count word
library(stringi) # use to count lines fast
library(pryr) # to see file size with command object_size
library(wordcloud) # for visualization
#read in the data and read several lines of data:
con_twitts<- file("en_US.twitter.txt",open="rb")
con_news<- file("en_US.news.txt", open="rb")
con_blogs<- file("en_US.blogs.txt", open="rb")

twitts<-readLines(con_twitts,encoding="UTF-8",warn=FALSE)
news<-readLines(con_news,encoding="UTF-8")
```

```
blogs<-readLines(con_blogs,encoding="UTF-8")
#close connection:
close(con_twitts)
close(con_news)
close(con_blogs)
##length(readLines(twitts, encoding="UTF-8"))
#length(readLines(news,encoding="UTF-8"))
#length(readLines(blogs,encoding="UTF-8"))

#2 summary of data

#see how many lines of data(run bash command in R script)
#below is a way to count lines ( slower)
#twitts_path<-"~/en_US.twitter.txt"
#R.utils::countLines(twitts_path)
#news_path<-"~/en_US.news.txt"
#R.utils::countLines(news_path)
#blogs_path<-"~/en_US.blogs.txt"
#R.utils::countLines(blogs_path)
#This below is a faster way to count lines:
stri_stats_general(twitts)[1]
stri_stats_general(news)[1]
stri_stats_general(blogs)[1]
object_size(twitts)
object_size(news)
object_size(blogs)
#count and sum words in each data:
sum(sapply(gregexpr("\\W+", twitts), length))
sum(sapply(gregexpr("\\W+", news), length))
```

```
sum(sapply(gregexpr("\\W+", blogs), length))

#3  Preprocessing data:



#random sampling:
#Since we don't need to load in and use all of the data, I would like to just read
#several lines of each data and conbine them into one data.
#ramdom select several lines in each data and combine them into one data -all.
set.seed(1233)
ran_twitts<-sample(twitts, 2000, replace=FALSE)
ran_news<-sample(news, 2000, replace=FALSE)
ran_blogs<-sample(blogs, 2000, replace=FALSE)
#all is the combined data by twitts_part news_part and blogs_part.
all<-paste(ran_twitts, ran_news, ran_blogs)
#count how many words in data:
stri_stats_general(all) #it should be 2000.

sum(sapply(gregexpr("\\W+", all), length))

library(tm)
#tokenization function:
#create corpus
my_corp<-Corpus(VectorSource(all), readerControl=list(language="lat"))
#write corpus on hard drive:
writeCorpus(my_corp)
#find more about meta data:
inspect(my_corp[1:2])
#do transformation on corpus I made:
```

```
my_corp<-tm_map(my_corp, PlainTextDocument)
my_corp<-tm_map(my_corp, removePunctuation)
my_corp<-tm_map(my_corp, stripWhitespace)
my_corp<-tm_map(my_corp, tolower)
my_corp<-tm_map(my_corp, removeNumbers)
my_corp<-tm_map(my_corp, stemDocument)
my_corp<-tm_map(my_corp,tolower) #convert to lower case
my_corp<-tm_map(my_corp, removeWords, stopwords("english"))
profane_path<-paste(getwd(), "/profane.txt",sep="")
my_corp<-tm_map(my_corp, removeWords, profane_path)
corp_clean<-my_corp
corp_clean<-Corpus(VectorSource(corp_clean))
```

```
#generate a document term matrix:
dtm<-DocumentTermMatrix(corp_clean)
inspect(dtm[1:20, 100:110])
#clustering:
findFreqTerms(x=dtm, lowfreq=5)# find terms that occurs at least 5 times;
findAssocs(x=dtm, term="god", corlimit=0.4)
inspect(removeSparseTerms(dtm, 0.4))
```

```
#4 visualization :
my_dtm<-as.matrix(dtm)
order<-sort(colSums(my_dtm), decreasing=TRUE)[1:100]
order_name <-names(order)
word_freq<-data.frame(order)
df<-data.frame(as.character(rownames(word_freq)), word_freq)
colnames(df)[1]="word_names"
names(df)<-c("word_names","frequency")
rownames(df)<-c(1:nrow(df))
#transformed to a frequency table for plotting easily.
head(df)
#order frequency as decreasing and save to a data frame called df_order
df_order<-df[order(-df$frequency), ]
#df_freq<-as.vector(rep(df_order$word_names, df_order$frequency))
#df_new<-data.frame(df_freq)
#head(df_new)

g<-ggplot(df_order,aes(x=word_names, y=frequency)) +
      geom_bar(stat="identity",colour="yellow", fill="pink") +
      labs(x="frequency of each word", y="word names") +
      ggtitle("histogram of word frequencies") +
      coord_flip() +
      theme_bw() +
      geom_text(aes(label=frequency))

print(g)

#use word cloud.
```

```
v<-sort(colSums(my_dtm), decreasing=TRUE)
words<-names(v)
d<-data.frame(word=words, freq=v)
wordcloud(d$word,d$freq,max.words=150,colors=brewer.pal(5,"Set1"),random.order=FALSE)



#draw a graph:
#source("http://bioconductor.org/biocLite.R")
#biocLite("graph")
library(graph)
#source("http://bioconductor.org/biocLite.R")
#biocLite("Rgraphviz")
library(Rgraphviz)
freq<-findFreqTerms(dtm, lowfreq=15)
plot(dtm, term=freq, corThreshold=0.3, weighting=T)

#cluster:
dtm_rst<-removeSparseTerms(dtm, sparse=0.95)
df_dtm<-as.data.frame(inspect(dtm_rst))
d<-dist(scale(df_dtm), method="euclidean")
fit<-hclust(dist_mat,method="ward.D")
plot(fit)
rect.hclust(fit, k=5)



#5 N-gram modeling:
library(tau)
```

```
#str<-concat(all)
#ng<-ngram(str, n=3)
#ng
#print(ng, full=TRUE)
#str(ng)
#get.ngrams(ng)
#get.string(ng)
#get.nextwords(ng)



ngram<-function(x, n) {
      ngram_toke<- NGramTokenizer(x, Weka_control(min = n, max = n))









}




tdm <- TermDocumentMatrix(a, control = list(tokenize = TrigramTokenizer))
tdm <- removeSparseTerms(tdm, 0.75)
inspect(tdm[1:5,1:5])
```

#6 prediction :

#conclusion :

#reference:

#http://gastonsanchez.com/Handling_and_Processing_Strings_in_R.pdf

#http://cran.r-project.org/web/views/NaturalLanguageProcessing.html

#http://www.jstatsoft.org/v25/i05/

#brewer.pal:

#http://www.datavis.ca/sasmac/brewerpal.html

#appendix: