Course10\_wk2\_MilestoneReport

SH

Sep 3, 2016

## Title: Course10 Milestone Exploratory Report

#### Author: SH

### Synopsis/Overview

The goal of this report is to show the progress made towards creating a natural language processing (NLP) prediction algorithm. Several text-processing packages e.g., "tm" and "nlp" were installed. Among the data downloaded, files in directory "en\_US" were loaded as a corpus, and file "en\_US.news.txt" was further explored. A basic summary statistics about the data was created and a few findings were noted. Finally, a rough plan for creating a prediction algorithm and Shiny app was presented for classmate feedback.

### Data Loading

The following code loads the data as well as likely-used libraries for text processing.

rm(list=ls())  
  
  
library(knitr)  
library(data.table)  
library(stringr)  
library(tm)

## Loading required package: NLP

library(rJava)  
library(openNLP)  
library(ngram)  
library(tokenizers)

##   
## Attaching package: 'tokenizers'

## The following object is masked from 'package:tm':  
##   
## stopwords

# to look at all files in directory "en\_US"  
txt1 <- "en\_US"  
en\_US <- VCorpus(DirSource(txt1, encoding = "UTF-8"), readerControl = list(language = "en"))

Table 1 gives the summary of the file collection in directory "en\_US".

kable(summary(en\_US))

|  |  |  |  |
| --- | --- | --- | --- |
|  | Length | Class | Mode |
| en\_US.blogs.txt | 2 | PlainTextDocument | list |
| en\_US.news.txt | 2 | PlainTextDocument | list |
| en\_US.twitter.txt | 2 | PlainTextDocument | list |

Table 1. Summary of the file collection (i.e., corpus)

Following code gives word count, line count, and file size.

require(stringi)

## Loading required package: stringi

require(ngram)  
require(knitr)  
  
rm(list="en\_US")  
  
tw <- readLines(con <- file("en\_US/en\_US.twitter.txt"), encoding = "UTF-8", skipNul = TRUE)  
close(con)  
blog <- readLines(con <- file("en\_US/en\_US.blogs.txt"), encoding = "UTF-8", skipNul = TRUE)  
close(con)  
news <- readLines(con <- file("en\_US/en\_US.news.txt"), encoding = "UTF-8", skipNul = TRUE)

## Warning in readLines(con <- file("en\_US/en\_US.news.txt"), encoding =  
## "UTF-8", : incomplete final line found on 'en\_US/en\_US.news.txt'

close(con)  
  
# number of lines  
tw\_len <- length(tw) # 2360148  
blog\_len <- length(blog) # 899288  
news\_len <- length(news) # 77259  
  
# length of the longest line  
tw\_len\_long <- max(stri\_length(tw)) # 140  
blog\_len\_long <- max(stri\_length(blog)) # 40833  
news\_len\_long <- max(stri\_length(news)) # 5760  
  
# make table  
fName <- c("en\_US.twitter.txt","en\_US.blogs.txt", "en\_US.news.txt")  
lineCnt <- c(tw\_len, blog\_len, news\_len)  
maxLen <- c(tw\_len\_long, blog\_len\_long, news\_len\_long)  
fSize <- c("163,189 KB","205,235 KB", "200,989 KB")  
a <- data.table(fName,lineCnt,maxLen,fSize)  
  
kable(a)

|  |  |  |  |
| --- | --- | --- | --- |
| fName | lineCnt | maxLen | fSize |
| en\_US.twitter.txt | 2360148 | 140 | 163,189 KB |
| en\_US.blogs.txt | 899288 | 40833 | 205,235 KB |
| en\_US.news.txt | 77259 | 5760 | 200,989 KB |

Table 2. Line count, line maximum length of line, and file size.

### Data Preprossing

We focus on file "en\_US.news.txt" due to difficulty of handling file collection all together. Data has gone through several transformation steps such as converting to lowercase and removing puncuation marks. In addition, steming was also imposed hoping to reduce unique words but it was decided not to use this process at this point.

# focus on data "en\_US.news.txt"  
raw <- paste(readLines("en\_US/en\_US.news.txt"),collapse=" ")

## Warning in readLines("en\_US/en\_US.news.txt"): incomplete final line found  
## on 'en\_US/en\_US.news.txt'

# remove problematic characters  
news <- iconv(raw, "latin1", "ASCII", sub="")  
  
rm(list="raw")  
  
# convert to corpus  
newsC <- Corpus(VectorSource(news))  
rm(list="news")  
# transformation  
t <- tm\_map(newsC, content\_transformer(tolower))  
# t <- tm\_map(newsC,FUN=tolower)  
t <- tm\_map(t,removePunctuation)  
t <-tm\_map(t,removeNumbers)  
t <- tm\_map(t,stripWhitespace)  
# save tmp file in case R crashes  
# writeCorpus(t, filenames="newsC\_cleaned\_tmp.txt")  
  
rm(list="newsC")

### Exploratory Analysis

#### Unigram

Frequency of words, sorted by frequency in descending order and displayed with an interval of 1000, in the data file is shown in Figure 1.

require(data.table)  
require(tokenizers)  
  
str <- toString(t$content)  
# load("str.RData")  
t1a <- tokenize\_words(str)  
  
# find and remove any words other alphatic  
# idx <- grep('^[A-Za-z]+$', t1a )  
# str(idx) # int(0)  
# t1a <- t1a[idx]  
  
# uni <- ngram(str, n=1)  
# save(uni, file="uni.RData")  
  
dt <- data.table(table(t1a))  
rm(list="t1a")  
dt <- setorder(dt, -N)  
dt$rank <- seq(1,nrow(dt))  
dt$freq <- dt$N/sum(dt$N)  
  
# kable(dt[seq(1,nrow(dt),1000),])  
  
# histogram  
barplot(dt$freq[1:100], xlab="Word", ylab="Frequency (%)")

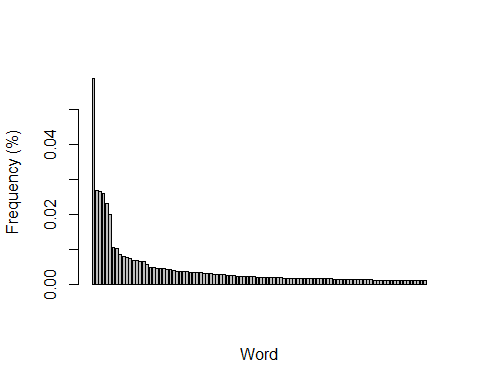
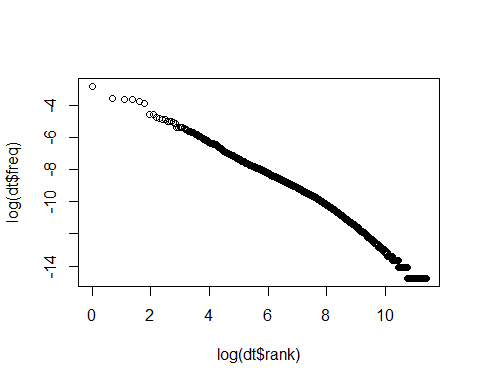


Figure 1. Histogram of of top 100 most-frequently-occuring words.

If we plot log(frequency) against log(rank), it appears that frequency x rank is approximately constant (Figure 2). This demonstrates Zipf's Law (see "text\_preprocessing.pdf" by X.Zhu).

plot(log(dt$rank),log(dt$freq))



rm(list=c("dt","t"))

Figure 1. Log-log plot of frequency and rank of unique words

#### Bigram

Bigram is explored by using the following code.

bi <- ngram(str, n = 2, sep = " ")  
# save(bi, file="bi.RData")  
# save(str, file="str.RData")  
  
t2a <- get.ngrams(bi)  
dt <- data.table(table(t2a))  
rm(list="t2a")  
dt <- setorder(dt, -N)  
unique(dt$N)  
  
rm(list=c("bi","dt"))

It appears that there is no repeat bigrams in the data.

Trigram can be produced in similar fashion.

### Next Step

Next step is to use the Ngram method to build a predictive model that predict the next word that is likely to occur. This model will be the brain of a Shiny app for word prediction. The words typed in will be tokenized and the last few words will be inputted to the model for next word prediction.

### Help Needed

Two issues have been encountered at this stage. 1. Only word document can be knitted due to Upgrading from R3.2.5 to R3.3.1. 2. My R session keeps crashing due to memory issue. Any suggestion is welcome to circumvent these problems. Since there is no mechanism to load the knitted word document to RPubs, it was uploaded to github.