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.

explains your exploratory analysis and your goals for the eventual app and algorithm. This document should be concise and explain only the major features of the data you have identified and briefly summarize your plans for creating the prediction algorithm and Shiny app in a way that would be understandable to a non-data scientist manager. You should make use of tables and plots to illustrate important summaries of the data set. The motivation for this project is to: 1. Demonstrate that you've downloaded the data and have successfully loaded it in.2. Create a basic report of summary statistics about the data sets.3. Report any interesting findings that you amassed so far.4. Get feedback on your plans for creating a prediction algorithm and Shiny app.

### Data Loading

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

rm(list=ls())  
  
# install.packages("knitr")  
# install.packages("rmarkdown")  
# install.packages("tm")  
# install.packages("R.utils")  
# install.packages("tau")  
# install.packages("rJava")  
# install.packages("openNLP")  
# install.packages("ngram")  
# install.packages("tokenizers")  
# install.packages("RWeka")  
  
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

# library(ggplot2)  
# library(RWeka)  
  
# 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 |

rm(list="en\_US")

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

### 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.

# 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")

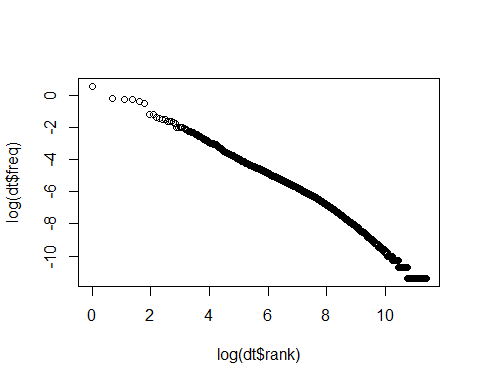
### Exploratory Analysis

#### Unigram

Frequency of words, sorted by frequency in descending order and displayed with an interval of 1000, in the data file is listed in Table 1. If we plot log(frequency) against log(rank), it appears that frequency x rank is approximately constant (Figure 1). This demonstrates Zipf's Law (see "text\_preprocessing.pdf" by X.Zhu).

# library(data.table)  
  
str <- toString(t$content)  
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/nrow(dt)  
  
# kable(dt[seq(1,nrow(dt),1000),])

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

My R session keeps quitting due to memory issue. Any suggestion is welcome to circumvent the problem.