Capstone_Final_Project

ETSulato

14/11/2020

Synopsis

This project presents an algorithm model for forecasting next word in the Shiny application, using the SwiftKey database, made available by Coursera during the Data Science Capstone course.

Loading packages

```
library(tm)

## Loading required package: NLP

library(RWeka)
library(SnowballC) # important for the wordcloud package use
library(wordcloud)

## Loading required package: RColorBrewer

library (stringi) # string/text manipulation
library(rvest) # reading html

## Loading required package: xml2
```

Loading data files

```
if(!file.exists("./final/en_US/en_US.blogs.txt") &&
   !file.exists("./final/en_US/en_US.news.txt") &&
    !file.exists("./final/en_US/en_US.twitter.txt")){
  URL <-
"https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-
SwiftKey.zip"
  download.file(URL, destfile="Coursera-SwiftKey.zip")
  unzip(zipfile="Coursera-SwiftKey.zip")
}
```

Reading twitter, news, and blogs txt data files

```
## twitter
con_twitter <- file("./final/en_US/en_US.twitter.txt")
twitter_raw <- readLines(con_twitter, encoding = "UTF-8", skipNul = TRUE)
```

```r
close(con_twitter)

## news
con_news <- file("./final/en_US/en_US.news.txt",open="r")
news_raw <- readLines(con_news, encoding = "UTF-8", skipNul = TRUE)
close(con_news)

## blogs
con_blogs<-file("./final/en_US/en_US.blogs.txt", open="r")
blogs_raw <- readLines(con_blogs, encoding = "UTF-8", skipNul = TRUE)
close(con_blogs)

rm(con_blogs,con_news,con_twitter)
```

Summary of the files

```r
## Word counts
words_twitter<-sum(stri_count_boundaries(twitter_raw, type="word"))
words_blog<-sum(stri_count_boundaries(blogs_raw, type="word"))
words_news<-sum(stri_count_boundaries(news_raw, type="word"))

# Summary of the files (lines and words counts)
files_summary<- data.frame(files=c("twitter","blogs", "news"),
lines=c(length(twitter_raw),
length(blogs_raw),length(news_raw)),
words=c(words_twitter,words_blog,words_news))

files_summary

##     files   lines    words
## 1 twitter 2360148 65264908
## 2   blogs  899288 79779789
## 3    news   77259  5718223
```

Data processing

Removing unwanted characters from converting Latin codepage to ASCII.

```r
twitter_clean <- iconv(twitter_raw, 'UTF-8', 'ASCII', "byte")
blogs_clean<- iconv(blogs_raw, 'UTF-8', 'ASCII', "byte")
news_clean <- iconv(news_raw, 'UTF-8', 'ASCII', "byte")
```

Data selection

A total of 0.1% of the data in each file was selected. Subsequently, the selected data were unified and converted into corpus (natural language).

```
set.seed(333)

twitter_sample <- sample(twitter_clean, length(twitter_clean)*0.001)

blogs_sample <- sample(blogs_clean, length(blogs_clean)*0.001)

news_sample <- sample(news_clean, length(news_clean)*0.001)

all <- c(twitter_sample,blogs_sample,news_sample)
all_corpus <- VCorpus(VectorSource(all))

rm(twitter_clean,twitter_raw,twitter_sample)
rm(blogs_clean,blogs_raw,blogs_sample)
rm(news_clean,news_raw,news_sample)
```

Text cleaning

All characters that can’t aggregate any meaning for the Natural Language Processing that the corpus might contain must be cleaned.

```
all_corpus <- tm_map(all_corpus, content_transformer(tolower))
all_corpus <- tm_map(all_corpus, removePunctuation)
all_corpus <- tm_map(all_corpus, removeNumbers)
all_corpus <- tm_map(all_corpus, stripWhitespace)
```

Tokeninzation

Tokenization was performed to build matrices of bigrams, trigrams, and quadgrams. Thus, work on the Shiny application will be carried out from the two (bigram), three (trigram), and four (quadgram) previous words. For this, the RWeka and NGramTokenizer packages were used.

```
bi_tokenizer <- function(x){
                    NGramTokenizer(x, Weka_control(min = 2, max = 2))}
tri_tokenizer <-function(x){
                    NGramTokenizer(x, Weka_control(min = 3, max = 3))}
quad_tokenizer <-function(x){
                    NGramTokenizer(x, Weka_control(min = 4, max = 4))}
```

Create Term Document Matrices

Constructs or coerces to a term-document matrix or a document-term matrix.

```
uni_tdm <- TermDocumentMatrix(all_corpus)
bi_tdm <- TermDocumentMatrix(all_corpus, control = list(tokenize =
bi_tokenizer))
tri_tdm <-TermDocumentMatrix(all_corpus, control = list(tokenize =
tri_tokenizer))
quad_tdm <-TermDocumentMatrix(all_corpus, control = list(tokenize =
quad_tokenizer))
```

Frequency of words

The counting of the frequencies was performed to sort them in decreasing order. Then, the results were stored into a data frame.

```
uni_matrix <- as.matrix(removeSparseTerms(uni_tdm, sparse = 0.999))
bi_matrix <- as.matrix(removeSparseTerms(bi_tdm, sparse = 0.999))
tri_matrix <- as.matrix(removeSparseTerms(tri_tdm, sparse = 0.9995))
quad_matrix <- as.matrix(removeSparseTerms(quad_tdm, sparse = 0.999))

uni_matrix <- sort(rowSums(uni_matrix),decreasing=TRUE)
bi_matrix <- sort(rowSums(bi_matrix),decreasing=TRUE)
tri_matrix <- sort(rowSums(tri_matrix),decreasing=TRUE)
quad_matrix <- sort(rowSums(quad_matrix),decreasing=TRUE)

uni_matrix_df <- data.frame(word = names(uni_matrix),freq=uni_matrix,
row.names = 1:length(uni_matrix))
bi_matrix_df <- data.frame(word = names(bi_matrix),freq=bi_matrix,
row.names = 1:length(bi_matrix))
tri_matrix_df <- data.frame(word = names(tri_matrix),freq=tri_matrix,
row.names = 1:length(tri_matrix))
quad_matrix_df <- data.frame(word = names(quad_matrix),freq=quad_matrix,
row.names = 1:length(quad_matrix))
```

Save data frames into r-compressed files

```
#1-grams
write.csv(uni_matrix_df[uni_matrix_df$freq >
0,],"unigram.csv",row.names=F)
unigram <- read.csv("unigram.csv",stringsAsFactors = F)
saveRDS(unigram,"unigram.RData")



#2-grams
write.csv(bi_matrix_df[bi_matrix_df$freq > 1,],"bigram.csv",row.names=F)
bigram <- read.csv("bigram.csv",stringsAsFactors = F)
saveRDS(bigram,"bigram.RData")

#3-grams
write.csv(tri_matrix_df[tri_matrix_df$freq >
1,],"trigram.csv",row.names=F)
trigram <- read.csv("trigram.csv",stringsAsFactors = F)
saveRDS(trigram,"trigram.RData")
```

```r
#4-gram
write.csv(quad_matrix_df[quad_matrix_df$freq >
1,],"quadgram.csv",row.names=F)
quadgram <- read.csv("quadgram.csv",stringsAsFactors = F)
saveRDS(quadgram,"quadgram.RData")
```

End