What is the most frequent word used - a milestone report for the capstone project

Averina Masha

May 22 2020

Introduction

This is a milestone report for the capestone project of Data Science Specialization.The goal of this report is to show that we've been familiar with the data. We will do some exploratory analysis and figure out our plan for the eventual app and algorithm. The major tasks of this report are: 1. download the data and load it in. 2. create a basic report of summary statistics about the data. 3. use 1-grams, 2-grams, and 3-grams to understand frequencies of words and word pairs. 4. discuss plans for creating a prediction algorithm and Shiny app.

Data loading

We first set our working directory, download the data and unzip it. The data was collected in four languanges: English, Russian, German, and Finnish. Since the data sources are newspapers and magazines, blogs, and Twitter, there are three text files (blogs, news, and twitter) for each language. In this report, we are only interested in the files in english.

```
setwd("C:/Users/Bell/Desktop/coursera/capstone")
fileUrl <-"https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
if (!file.exists("Coursera-SwiftKey.zip")){
  download.file(fileUrl, destfile = "./Coursera-SwiftKey.zip", method="curl")
}
unzip("./Coursera-SwiftKey.zip")
```

Summary of data statistics

We first check the file size.

```
fz1 <- file.info("final/en_US/en_US.blogs.txt")$size / 1024^2
fz2 <- file.info("final/en_US/en_US.news.txt")$size / 1024^2
fz3 <- file.info("final/en_US/en_US.twitter.txt")$size / 1024^2
filesizeMB <- rbind(fz1, fz2, fz3)
```

Then, we load the three interested text files by useing readLines function.

```
con1 <- file("final/en_US/en_US.blogs.txt", "rb")
blogs <- readLines(con1, encoding = "UTF-8")
close(con1)
#-------
con2 <- file("final/en_US/en_US.news.txt", "rb")
news <- readLines(con2, encoding = "UTF-8")
close(con2)
#------
con3 <- file("final/en_US/en_US.twitter.txt", "rb")
twitter <- readLines(con3, encoding = "UTF-8")
close(con3)
```

Next, we count the lines and the words for each file.

```
ls1 <- length(blogs)
ls2 <- length(news)
ls3 <- length(twitter)
linecount <- rbind(ls1, ls2, ls3)
#word count of each file
library(stringi)
ws1 <- sum(stri_count_words(blogs))
ws2 <- sum(stri_count_words(news))
ws3 <- sum(stri_count_words(twitter))
wordcount <- rbind(ws1, ws2, ws3)
```

Then, we get the file names and combine all the information together.

```
filenames<- Sys.glob("final/en_US/*.txt")
r <- data.frame(filenames,filesizeMB, linecount, wordcount)
rownames(r) <- c(1,2,3)
print(r)
```

```
##                       filenames filesizeMB linecount wordcount
## 1   final/en_US/en_US.blogs.txt   200.4242    899288  37546246
## 2    final/en_US/en_US.news.txt   196.2775   1010242  34762395
## 3 final/en_US/en_US.twitter.txt   159.3641   2360148  30093369
```

From the above simple data summary, we can see that these text files are very large files with size 200MB, 196MB,and 159MB respectively. They also contain huge number of lines and words. It is too time consuming to analyze the full dataset. Therefore, we will randomly sample 5% of lines in each text file. The sampled three files are combined together and then be written to a separate directory for further analysis.

```
set.seed(12345)
s_blogs <- sample(blogs, ls1*0.05)
s_news <- sample(news, ls2*0.05)
s_twitter <- sample(twitter, ls3*0.05)
s_Data <- c(s_blogs,s_news,s_twitter)
writeLines(s_Data, "sample/sampleData.txt")
rm(blogs, news, twitter, s_blogs, s_news, s_twitter, s_Data)
```

Data cleaning and preprocessing

Now, we are ready for data cleaning and tokenization. We first need to construct a corpus from our overall sampled text file.

```
library(NLP)
library(tm)
library(RWeka)
cname <- file.path("C:/Users/Bell", "Desktop", "Coursera", "capstone","sample")
docs <- Corpus(DirSource(cname))
```

The raw text file contains some characters, symbols, and words, that may not provide useful information. Therefore, we need to clean the data first. we will remove such things as numbers, punctuation, white space, etc. We will also remove the stopwords in english. Although we randomly sampled the original dataset, out corpus is still around 40Mb. Thus, We will simplify our R codes with pipes (%>%) to save some running time.

```
library(magrittr)
library(SnowballC)
docsf <- docs %>%
      tm_map(tolower) %>%  #transfer to lowercase
      tm_map(removePunctuation) %>% #remove punctuation
      tm_map(removeNumbers) %>%     #remove numbers
      tm_map(stemDocument) %>%      #remove common word endings
      tm_map(stripWhitespace)%>%    #strip white spaces
      tm_map(removeWords, stopwords("en"))%>%
      tm_map(PlainTextDocument)     #transfer to plain text document
```

Exploratory data analysis

After we clean and tokenize the data, we are ready to do some exploratory analysis. We will use 1-grams model to explore the most frequent words in the data.

```
UnigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min=1, max=1))
dtm_uni <- DocumentTermMatrix(docsf, control=list(tokenize=UnigramTokenizer))
dtm_uni_freq <-dtm_uni %>%
              removeSparseTerms(0.2) %>%
             as.matrix %>%
             colSums %>%
             sort(decreasing=TRUE)
dtm_uni_freq_d <- data.frame(word = names(dtm_uni_freq), freq = dtm_uni_freq)
head(dtm_uni_freq_d, 10)
```

```
##      word  freq
## will will 15527
## said said 15196
## just just 15091
## one   one 13682
## like like 13206
## can   can 12305
## get   get 11556
## time time  9615
## new   new  9582
## good good  8885
```

Then, we use the 2-grams model to determine the most frequent two-word phrases.

```
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min=2, max=2))
dtm_bi <- DocumentTermMatrix(docsf, control=list(tokenize=BigramTokenizer))
dtm_bi_freq <-dtm_bi %>%
  removeSparseTerms(0.2) %>%
  as.matrix %>%
  colSums %>%
  sort(decreasing=TRUE)
dtm_bi_freq_d <- data.frame(word = names(dtm_bi_freq), freq = dtm_bi_freq)
head(dtm_bi_freq_d, 10)
```

```
##                   word freq
## right now     right now 1220
## last year     last year  970
## new york       new york  970
## cant wait     cant wait  930
## dont know     dont know  778
## last night   last night  719
## high school high school  715
## years ago     years ago  690
## last week     last week  642
## feel like     feel like  621
```

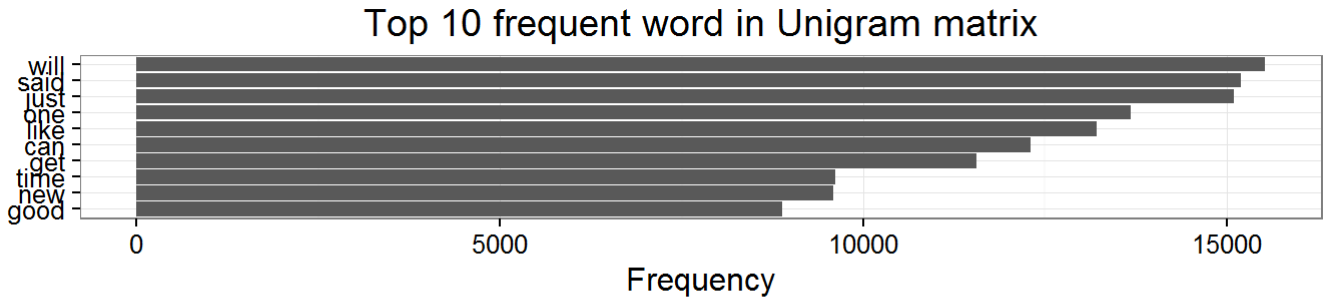
Next, we use 3-grams to look at the most frequent three-word phrases.

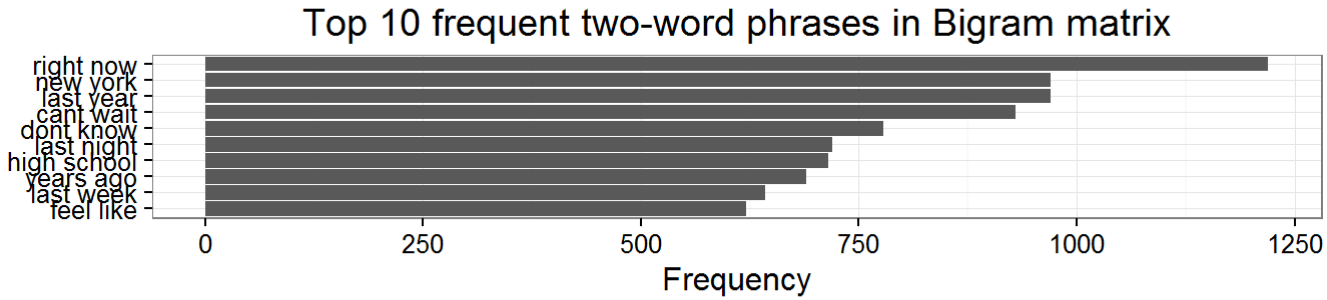
```
TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min=3, max=3))
dtm_tri <- DocumentTermMatrix(docsf, control=list(tokenize=TrigramTokenizer))
dtm_tri_freq <-dtm_tri %>%
  removeSparseTerms(0.2) %>%
  as.matrix %>%
  colSums %>%
  sort(decreasing=TRUE)
dtm_tri_freq_d <- data.frame(word = names(dtm_tri_freq), freq = dtm_tri_freq)
head(dtm_tri_freq_d, 10)
```

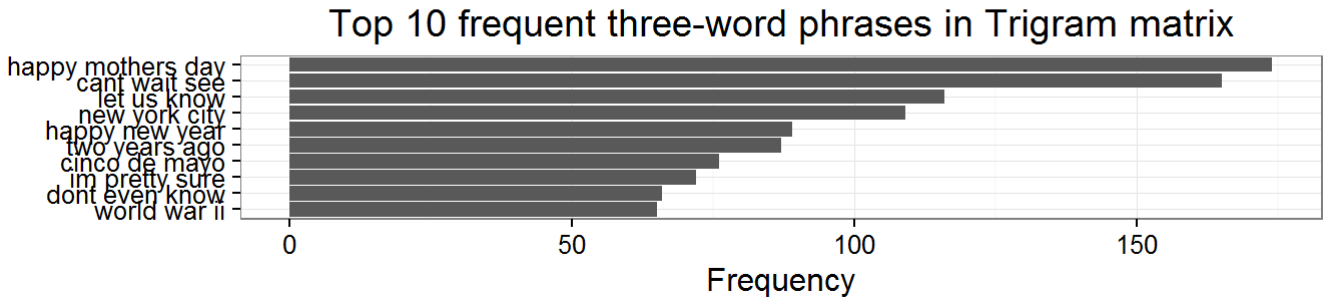
```
##                               word freq
## happy mothers day happy mothers day  174
## cant wait see         cant wait see  165
## let us know             let us know  116
## new york city         new york city  109
## happy new year       happy new year   89
## two years ago         two years ago   87
## cinco de mayo         cinco de mayo   76
## im pretty sure       im pretty sure   72
## dont even know       dont even know   66
## world war ii           world war ii   65
```

Finally, we show our data mining results in some histogram plots.

```
library(ggplot2)
library(gridExtra)
frequ10 <- head(dtm_uni_freq_d, 10)
freqb10 <- head(dtm_bi_freq_d, 10)
freqt10 <- head(dtm_tri_freq_d, 10)
up <- ggplot(frequ10, aes(x=reorder(word,freq), y=freq)) +
  geom_bar(stat="identity") +
  theme_bw() +
  coord_flip() +
  theme(axis.title.y = element_blank()) +
  labs(y="Frequency", title="Top 10 frequent word in Unigram matrix")
bp <-ggplot(freqb10, aes(x=reorder(word,freq), y=freq)) +
  geom_bar(stat="identity") +
  theme_bw() +
  coord_flip() +
  theme(axis.title.y = element_blank()) +
  labs(y="Frequency", title="Top 10 frequent two-word phrases in Bigram matrix")
tp <- ggplot(freqt10, aes(x=reorder(word,freq), y=freq)) +
  geom_bar(stat="identity") +
  theme_bw() +
  coord_flip() +
  theme(axis.title.y = element_blank()) +
  labs(y="Frequency", title="Top 10 frequent three-word phrases in Trigram matrix")
grid.arrange(up, bp, tp, ncol=1, nrow =3)
```







Interesting findings

It is really interesting to look at the most frequent word or phrases occuring in the documents, and we do find some interesting things. First, in the cleaning process, we have to remove the stopwords such as “the”, “and”,“are”, etc. These kind of words don’t provide useful information, but they occur very frequently. Initially, we did not remove them, we get the most frequent word “the” with 236543 frequency. Second, we realize that it is very important to look at the data with two-word phrases or three-word phrases. For our 3-grams model, we get the most frequent phrases as “happy mothers day”, “cant wait see”, “let us know”, “new york city”, etc. It is very interesting and useful information. This is also very important for prediction of the next word in a sentence.

Plans for prediction algorithm and Shiny app

We plan to use N-grams models to buid our prediction algorithm. We will use 2-grams and 3-grams to first understand the distribution of words and relationship between the words in the corpora. Then, we may use a 4-gram model to find the most likely next word. This strategy has not been finalized. For the Shiny app, we will creat a simple interface where the user can input a string of text. Then, our prediction model will give a list of the most likely suggested next words.

References

https://rstudio-pubs-static.s3.amazonaws.com/31867_8236987cf0a8444e962ccd2aec46d9c3.html