Week 2 - Milestone Report

Ioannis Petridis

7th of January, 2019

Introduction and Preprocessing

For this assignment, I first loaded the necessary packages and the data. I also provided some basic information about the english dataset, like size in megabytes, number of lines, longest line in each file and number of words per line.

```
# Read the data in R
con <- file("./final/en_US/en_US.twitter.txt", "r")
twitter<-readLines(con,skipNul=TRUE,encoding="UTF-8")
close(con)

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

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

# some info about the dataset
# rounded size of files in megabytes
twitter_size <- round(file.info("./final/en_US/en_US.twitter.txt")$size / (1024^2))
blogs_size <- round(file.info("./final/en_US/en_US.blogs.txt")$size / (1024^2))
news_size <- round(file.info("./final/en_US/en_US.news.txt")$size / (1024^2))

# lines of files
#length(twitter)
#length(blogs)
#length(news)

# longest line in three files
#max(nchar(twitter))
#max(nchar(blogs))
#max(nchar(news))

# create a dataframe consisting of size, number of lines, max characters
df<-data.frame(file = c("twitter", "blogs", "news"),
               size_MB = c(twitter_size, blogs_size, news_size),
               num_lines = c(length(twitter), length(blogs), length(news)),
               longest_line_chars = c(max(nchar(twitter)), max(nchar(blogs)), max(nchar(news))),
               number_words = c(sum(stri_count_words(twitter)),sum(stri_count_words(blogs)),sum(stri_co
              )

df
```

```
##      file size_MB num_lines longest_line_chars number_words
## 1 twitter     159   2360148                140     30093413
## 2   blogs     200    899288              40833     37546239
## 3    news     196   1010242              11384     34762395
```

Exploratory Analysis

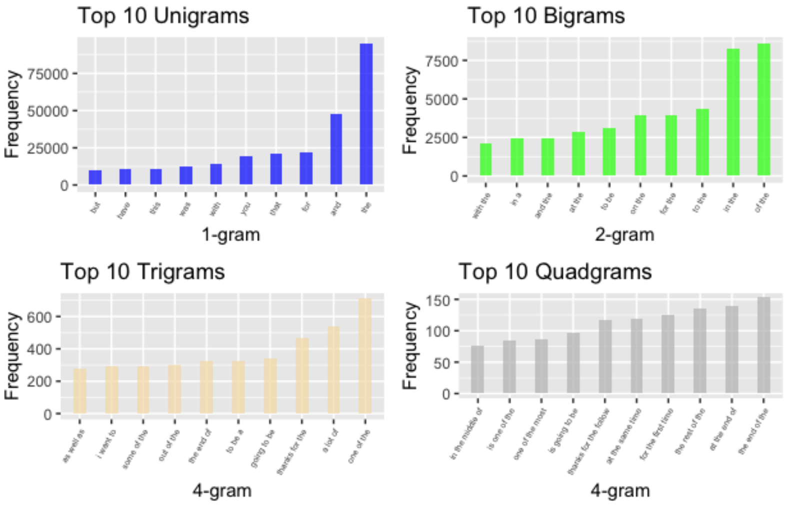
To continue, I proceeded with some exploratory analysis. I sampled 2% of the total data, cleaned it and plot the most frequent n-grams in histograms.

```r
# Exploratory analysis to find the most frequent n-grams in a sample of data
# I sampled 2% of the data
set.seed(12345)
twitter_sample <- sample(twitter, 0.02 * length(twitter))
blogs_sample <- sample(blogs, 0.02 * length(blogs))
news_sample <- sample(news, 0.02 * length(news))
###$$$$$$$$$$$$############
twitter_sample <- sample(twitter, size=10000, replace=TRUE)
blogs_sample <- sample(blogs, size=10000, replace=TRUE)
news_sample <- sample(news, size=10000, replace=TRUE)

# The data should be cleaned. Remove alpha-numeric characters and create a dataframe containing the sam
df_sample <- c(twitter_sample, blogs_sample, news_sample)
df_sample <- gsub("[^[:alnum:]']", " ",df_sample)

corpus_sample  <- Corpus(VectorSource(df_sample))
corpus_sample <- tm_map(corpus_sample, tolower)
corpus_sample <- tm_map(corpus_sample, removePunctuation)
corpus_sample <- tm_map(corpus_sample, removeNumbers)
corpus_sample <- tm_map(corpus_sample, stripWhitespace)
corpus_sample <- tm_map(corpus_sample, PlainTextDocument)

# n-grams
uniGramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max = 1))
biGramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max = 2))
triGramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3, max = 3))
fourGramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 4, max = 4))

uniGrams <- TermDocumentMatrix(corpus_sample, control = list(tokenize = uniGramTokenizer))
biGrams <- TermDocumentMatrix(corpus_sample, control = list(tokenize = biGramTokenizer))
triGrams <- TermDocumentMatrix(corpus_sample, control = list(tokenize = triGramTokenizer))
fourGrams <- TermDocumentMatrix(corpus_sample, control = list(tokenize = fourGramTokenizer))


#####
bigram <- NGramTokenizer(corpus_sample, Weka_control(min = 2, max = 2,delimiters = " \\r\\n\\t.,;:\"()?
bigram <- data.frame(table(bigram))
bigram <- bigram[order(bigram$Freq,decreasing = TRUE),]
names(bigram) <- c("words","freq")
head(bigram)
bigram$words <- as.character(bigram$words)
str2 <- strsplit(bigram$words,split=" ")
bigram <- transform(bigram,
                    one = sapply(str2,"[[",1),
                    two = sapply(str2,"[[",2))
bigram <- data.frame(word1 = bigram$one,word2 = bigram$two,freq = bigram$freq,stringsAsFactors=FALSE)
## saving files
```

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

trigram <- NGramTokenizer(corpus_sample, Weka_control(min = 3, max = 3,delimiters = " \\r\\n\\t.,;:\"()
trigram <- data.frame(table(trigram))
trigram <- trigram[order(trigram$Freq,decreasing = TRUE),]
names(trigram) <- c("words","freq")
head(trigram)
#####################
trigram$words <- as.character(trigram$words)
str3 <- strsplit(trigram$words,split=" ")
trigram <- transform(trigram,
                     one = sapply(str3,"[[",1),
                     two = sapply(str3,"[[",2),
                     three = sapply(str3,"[[",3))
# trigram$words <- NULL
trigram <- data.frame(word1 = trigram$one,word2 = trigram$two,
                      word3 = trigram$three, freq = trigram$freq,stringsAsFactors=FALSE)
# saving files
#write.csv(trigram[trigram$freq > 1,],"trigram.csv",row.names=F)
#trigram <- read.csv("trigram.csv",stringsAsFactors = F)
saveRDS(trigram,"trigram.RData")
#####



quadgram <- NGramTokenizer(corpus_sample, Weka_control(min = 4, max = 4,delimiters = " \\r\\n\\t.,;:\"(
quadgram <- data.frame(table(quadgram))
quadgram <- quadgram[order(quadgram$Freq,decreasing = TRUE),]
names(quadgram) <- c("words","freq")
quadgram$words <- as.character(quadgram$words)
str4 <- strsplit(quadgram$words,split=" ")
quadgram <- transform(quadgram,
                      one = sapply(str4,"[[",1),
                      two = sapply(str4,"[[",2),
                      three = sapply(str4,"[[",3),
                      four = sapply(str4,"[[",4))
# quadgram$words <- NULL
quadgram <- data.frame(word1 = quadgram$one,
                       word2 = quadgram$two,
                       word3 = quadgram$three,
                       word4 = quadgram$four,
                       freq = quadgram$freq, stringsAsFactors=FALSE)
# saving files
#write.csv(quadgram[quadgram$freq > 1,],"quadgram.csv",row.names=F)
#quadgram <- read.csv("quadgram.csv",stringsAsFactors = F)
saveRDS(quadgram,"quadgram.RData")

saveRDS(uniGrams,file="uniGrams.RData")
saveRDS(biGrams,file="biGrams.RData")
saveRDS(triGrams,file="triGrams.RData")
saveRDS(fourGrams,file="fourGrams.RData")
```

```r
# plots together
# 1-grams
frequent_terms <- findFreqTerms(uniGrams, lowfreq = 50)
frequency_terms <- rowSums(as.matrix(uniGrams[frequent_terms,]))
frequency_terms <- data.frame(unigram=names(frequency_terms), frequency=frequency_terms)
frequency_terms <- frequency_terms[order(-frequency_terms$frequency),][1:10,]

g1 <- ggplot(frequency_terms, aes(x=reorder(unigram, frequency), y=frequency)) +
  geom_bar(width=0.4,stat = "identity", fill = "blue", alpha=0.7) + xlab("1-gram") + ylab("Frequency") +
  theme(axis.text.x = element_text(angle = 60, size = 5, hjust = 1)) +
  labs(title = "Top 10 Unigrams")


# 2-grams
frequent_terms <- findFreqTerms(biGrams, lowfreq = 50)
frequency_terms <- rowSums(as.matrix(biGrams[frequent_terms,]))
frequency_terms <- data.frame(bigram=names(frequency_terms), frequency=frequency_terms)
frequency_terms <- frequency_terms[order(-frequency_terms$frequency),][1:10,]

g2 <- ggplot(frequency_terms, aes(x=reorder(bigram, frequency), y=frequency)) +
  geom_bar(width=0.4,stat = "identity", fill = "green", alpha=0.7) + xlab("2-gram") + ylab("Frequency")
  theme(axis.text.x = element_text(angle = 60, size = 5, hjust = 1)) +
  labs(title = "Top 10 Bigrams")


# 3-grams
frequent_terms <- findFreqTerms(triGrams, lowfreq = 50)
frequency_terms <- rowSums(as.matrix(triGrams[frequent_terms,]))
frequency_terms <- data.frame(trigram=names(frequency_terms), frequency=frequency_terms)
frequency_terms <- frequency_terms[order(-frequency_terms$frequency),][1:10,]

g3 <- ggplot(frequency_terms, aes(x=reorder(trigram, frequency), y=frequency)) +
  geom_bar(width=0.4,stat = "identity", fill = "wheat", alpha=0.7)+
  theme(axis.text.x = element_text(angle = 60, size = 5, hjust = 1)) + xlab("4-gram") + ylab("Frequency
  labs(title = "Top 10 Trigrams")


# 4-grams
frequent_terms <- findFreqTerms(fourGrams, lowfreq = 50)
frequency_terms <- rowSums(as.matrix(fourGrams[frequent_terms,]))
frequency_terms <- data.frame(fourgram=names(frequency_terms), frequency=frequency_terms)
frequency_terms <- frequency_terms[order(-frequency_terms$frequency),][1:10,]

g4 <- ggplot(frequency_terms, aes(x=reorder(fourgram, frequency), y=frequency)) +
  geom_bar(width=0.4,stat = "identity", fill = "grey", alpha=0.7) +
  theme(axis.text.x = element_text(angle = 60, size = 5, hjust = 1)) +
  xlab("4-gram") + ylab("Frequency") +
  labs(title = "Top 10 Quadgrams")

grid.arrange(g1, g2, g3, g4, nrow = 2)
```

```r
### The calculations has already been made and saved to a png file. Therefore I loaded just the png fil
img <- load.image("/Users/iopetrid/Desktop/Coursera/Data Science/10_Capstone/Rplot.png")
plot(img,axes=FALSE)
```



Plans for the future

The following steps would be to create a shiny app that will predict the next word based on the word that has given as input.