

# Milestone\_Report\_Project\_Doc

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)
library(wordcloud)

## Loading required package: RColorBrewer

library(stringi)
library(ggplot2)

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##   annotate
```

## 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)
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

```
## Word counts
words_twitter<-sum(str_count_boundaries(twitter_raw, type="word"))
words_blog<-sum(str_count_boundaries(blogs_raw, type="word"))
words_news<-sum(str_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.

```
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)
```

## Tokenization

Tokenization was performed to build matrices of unigrams, bigrams and trigrams. Thus, work on the Shiny application will be carried out from the one (unigram), two (bigram) and three (trigram) 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))}
```

## 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))
```

## 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(uni_tdm)
bi_matrix <- as.matrix(bi_tdm)
tri_matrix <- as.matrix(tri_tdm)

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

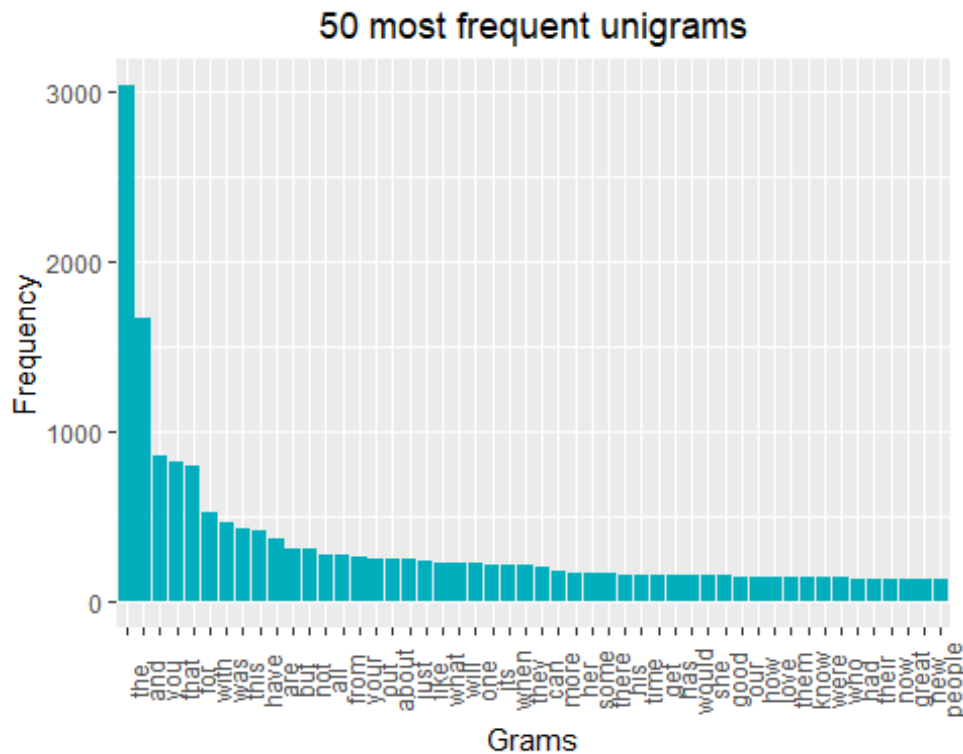
uni_matrix_df <- data.frame(word = names(uni_matrix),freq=uni_matrix,
row.names = 1:length(uni_matrix))
bi_matrix_df <- data.frame(word = names(bi_matrix),freq=bi_matrix,
row.names = 1:length(bi_matrix))
tri_matrix_df <- data.frame(word = names(tri_matrix),freq=tri_matrix,
row.names = 1:length(tri_matrix))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

## Plotting the data with frequencies

Histograms with the 50 most frequent n-grams

```
Barplot_uni <- ggplot(data=uni_matrix_df[1:50,],aes(x=reorder(word, -
freq),y=freq)) + geom_bar(stat = "identity", fill = "#00AFBB") +
theme(plot.title = element_text(hjust = 0.5))
Barplot_uni <- Barplot_uni + labs(x="Grams", y="Frequency", title="50
most frequent unigrams")
Barplot_uni <- Barplot_uni + theme(axis.text.x=element_text(angle=90))

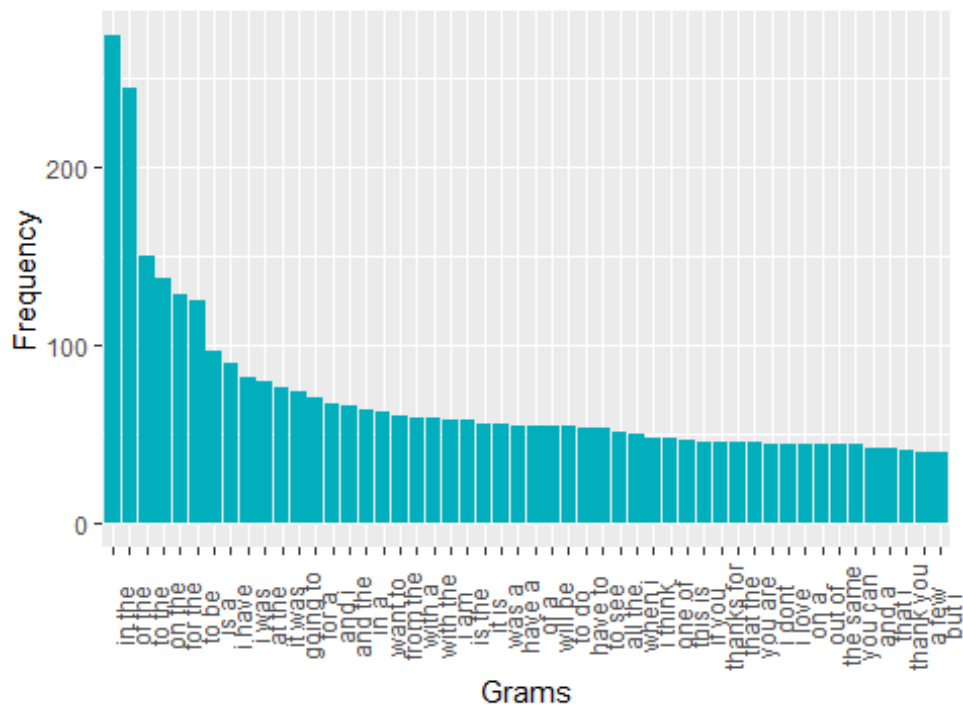
Barplot_uni
```



```
Barplot_bi <- ggplot(data=bi_matrix_df[1:50,],aes(x=reorder(word, -
freq),y=freq)) + geom_bar(stat = "identity", fill = "#00AFBB") +
theme(plot.title = element_text(hjust = 0.5))
Barplot_bi <- Barplot_bi + labs(x="Grams", y="Frequency", title="50 most
frequent bigrams")
Barplot_bi <- Barplot_bi + theme(axis.text.x=element_text(angle=90))
```

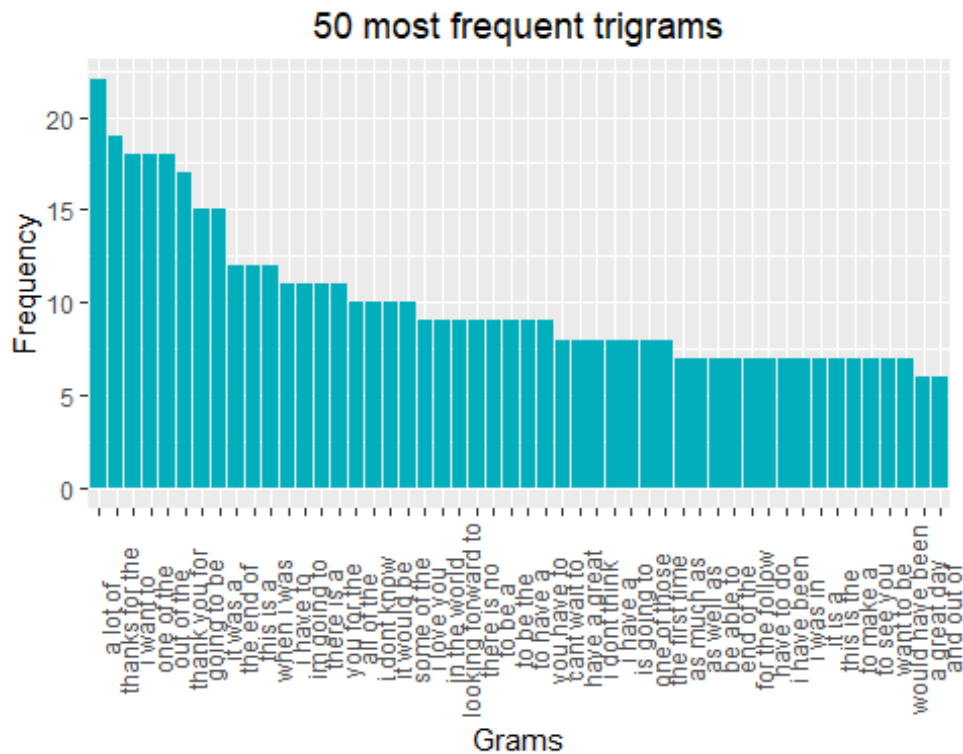
Barplot\_bi

### 50 most frequent bigrams



```
Barplot_tri <- ggplot(data=tri_matrix_df[1:50,],aes(x=reorder(word, -
freq),y=freq)) + geom_bar(stat = "identity", fill = "#00AFBB") +
theme(plot.title = element_text(hjust = 0.5))
Barplot_tri <- Barplot_tri + labs(x="Grams", y="Frequency", title="50
most frequent trigrams")
Barplot_tri <- Barplot_tri + theme(axis.text.x=element_text(angle=90))
```

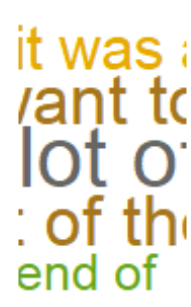
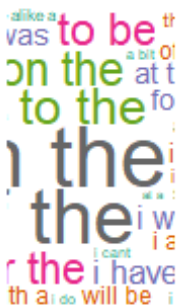
Barplot\_tri



## Wordcloud

The wordclouds present the main words, varying in size according to frequency.

```
thetwords<-list(uni_matrix_df,bi_matrix_df,tri_matrix_df)
par(mfrow=c(1,3))
for (i in 1:3) {
  wordcloud(thetwords[[i]]$word, thetwords[[i]]$freq, scale = c(4,0.4),
max.words=200, random.order=FALSE, rot.per=0, fixed.asp = TRUE,
use.r.layout = FALSE, colors=brewer.pal(8, "Dark2"))
}
```



According to the results mentioned above, the model for predicting words can be used in the construction of the Shiny application.