Swiftkey Word Prediction - Exploratory Data Analysis

Suresh Subramaniam

February 22, 2018

Exploratory Data Analysis - Swiftkey Dataset

The projects consists of building a prediction alogirithm to predict the next word when a user types in a word or multiple words.

This report is the first part and explains the exploratory data analysis of the data provided and the goals for the app and algorithm.

Tasks to accomplish

The main objectives of this report are as follows:

- demonstrate that the data has been downloaded successfully and loaded into R,
- present a basic summary statistics report on the datasets,
- present interesting characteristics of the datasets, and finally
- outline the plan for building the prediction algorithm.

Loading the data

The dataset has been downloaded from the Coursera instructions and the file is unzipped to reveal 3 datasets, one for News, the second for Twitter and third for Blogs. This data is loaded into R as below and a random sample of 15000 lines are selected from each dataset.

```
library(tm)
library(NLP)
#Load the Twitter dataset, and take a sample of 15000 lines
twitter <- readLines('en_US.twitter.txt', skipNul = TRUE)
twitterSample <- sample(twitter[-1], 15000, replace = FALSE)

#Load the News dataset and take sample of 15000 lines
news <- readLines('en_US.news.txt', skipNul = TRUE)
newsSample <- sample(news[-1], 15000, replace = FALSE)

# Load the Blogs dataset and take sample of 15000 lines
blogs <- readLines('en_US.blogs.txt', skipNul = TRUE)
blogsSample <- sample(blogs[-1], 15000, replace = FALSE)

#Append all samples into one file
fullText <- append(append(twitterSample, newsSample), blogsSample)

#Cleanup the text
#Create corpus
textCorpus <- VCorpus(VectorSource(fullText))

#Convert all characters to lowercase
textCorpus <- tm_map(textCorpus, content_transformer(tolower))
#Remove puctuation
textCorpus <- tm_map(textCorpus, removePunctuation)
#Remove numbers
textCorpus <- tm_map(textCorpus, removeNumbers)
#Remove white space
textCorpus <- tm_map(textCorpus, stripWhitespace)
```

Exploratory Data Analysis

Calculate some statistics of the text files

```
library(stringi)
# Number of words
stri_stats_latex(blogs)
```

```
##     CharsWord CharsCmdEnvir    CharsWhite         Words          Cmds
##     163325412             9      43302825      37865888             3
##        Envirs
##             0
```

```
stri_stats_latex(twitter)
```

```
##     CharsWord CharsCmdEnvir    CharsWhite         Words          Cmds
##     125769474          3033      36047952      30578933           963
##        Envirs
##             0
```

```
stri_stats_latex(news)
```

```
##     CharsWord CharsCmdEnvir    CharsWhite         Words          Cmds 
##      12502954             0       3114374       2665742             0 
##        Envirs 
##             0
```

```
#Number of characters in longest line
df <- as.data.frame(blogs)
df$chars <- nchar(as.character(df$blogs))
max(df$chars)
```

```
## [1] 40835
```

```
df <- as.data.frame(news)
df$chars <- nchar(as.character(df$news))
max(df$chars)
```

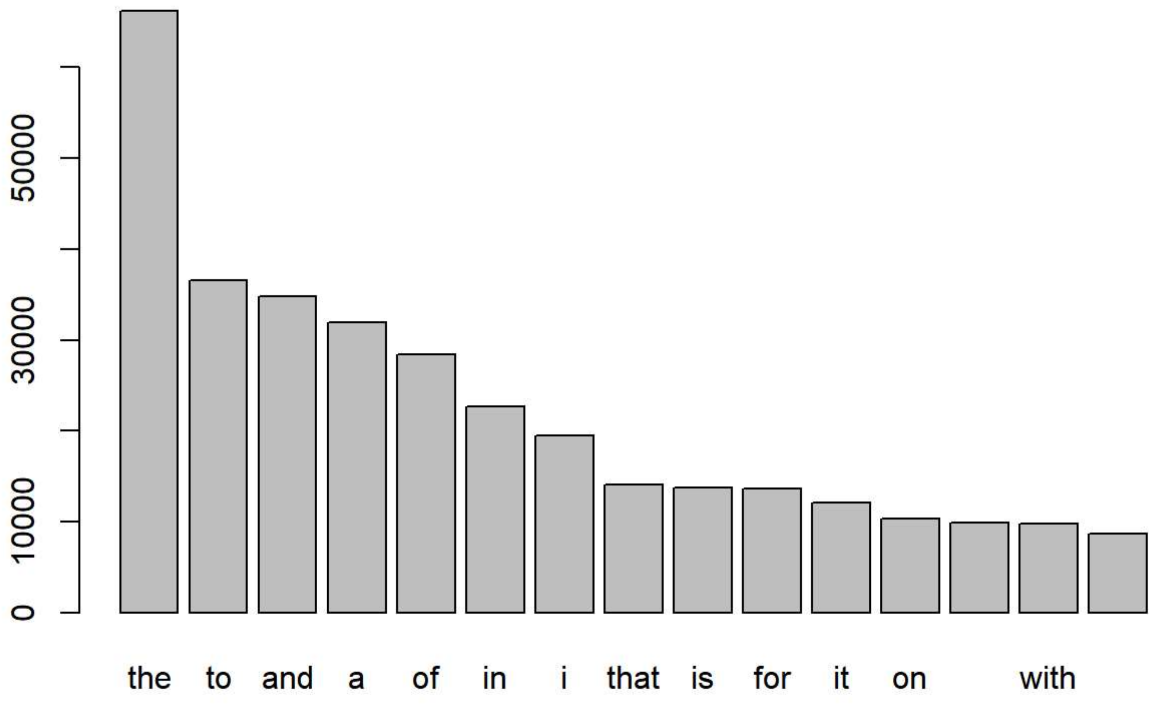
```
## [1] 5760
```

```
df <- as.data.frame(twitter)
df$chars <- nchar(as.character(df$twitter))
max(df$chars)
```

```
## [1] 213
```

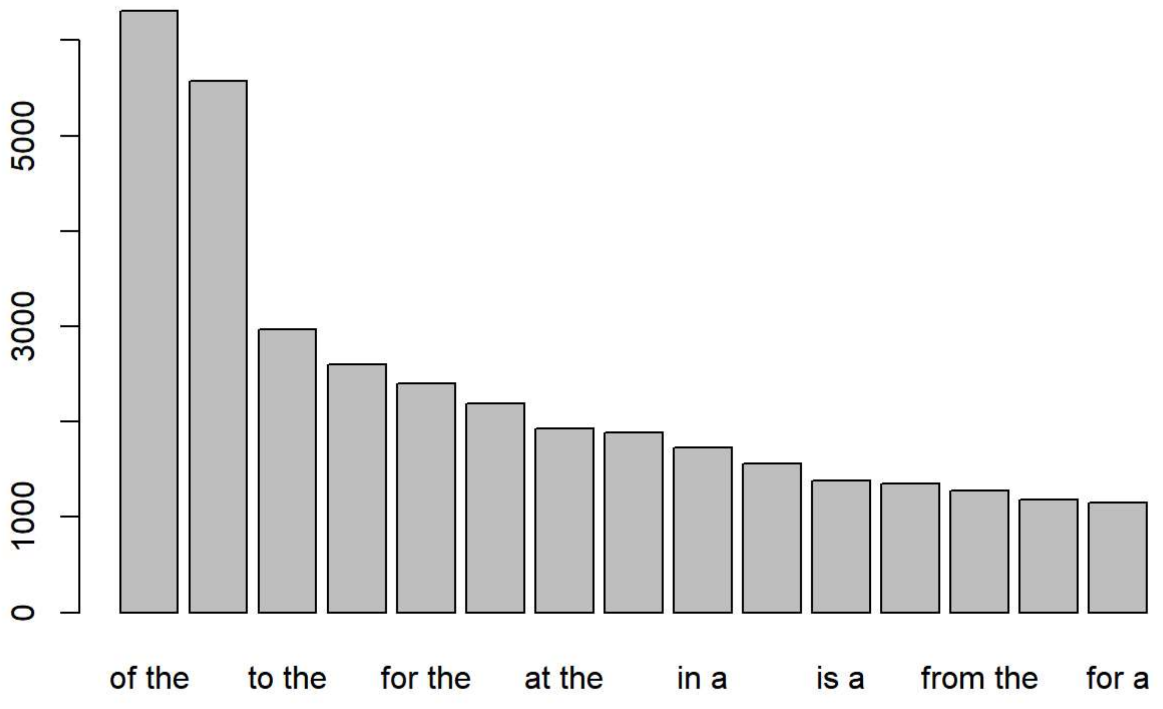
Word frequencies

```
library(tidytext)
library(dplyr)
#Convert the sample text into a dataframe
fullTextDf <- as.data.frame(fullText)
names(fullTextDf) <- "text"
#Toeknize into words
wordsDf <- unnest_tokens(fullTextDf, words, text, token='ngrams', n=1)

#Get count of single words
wordCounts <- count(wordsDf, words, sort=TRUE)

#plot histogram of 15 of the most frequent words
wordCounts <- head(wordCounts, 15)
barplot(wordCounts$n, names.arg=wordCounts$words)
```



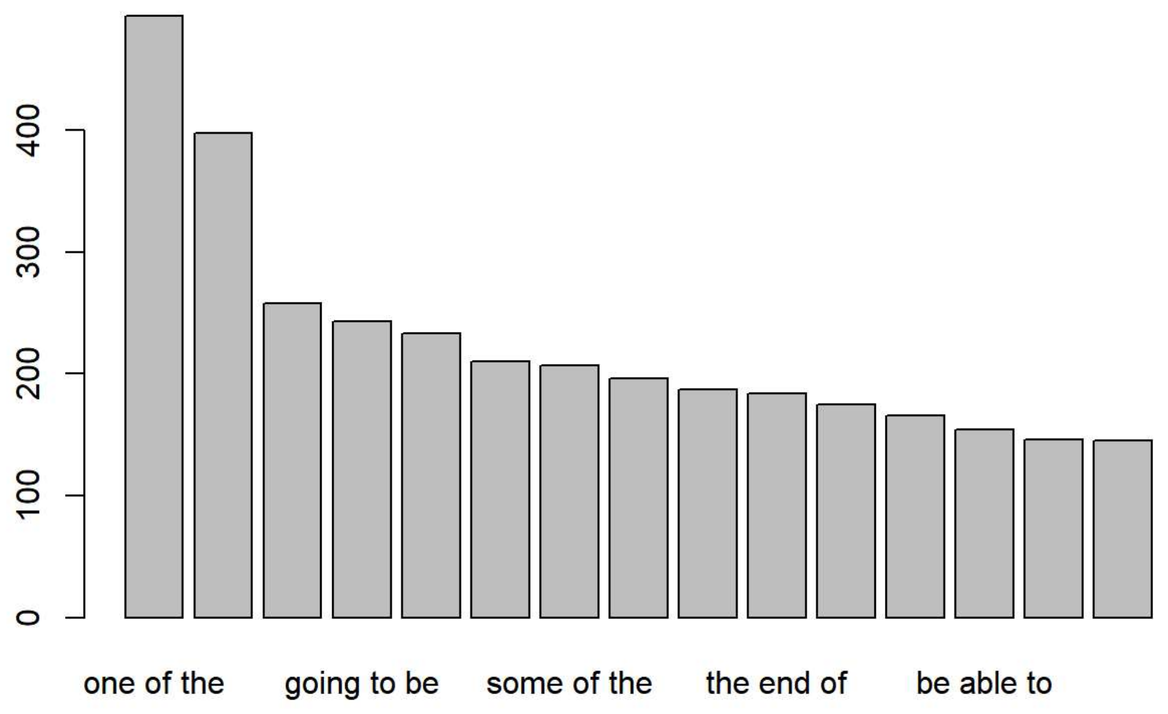
Bigram frequencies

```
#Toeknize into bigrams
bigramsDf <- unnest_tokens(fullTextDf, bigrams, text, token='ngrams', n=2)
#Get count of bigrams
bigramCounts <- count(bigramsDf, bigrams, sort=TRUE)

#plot histogram of 15 of the most frequent word pairs
bigramCounts <- head(bigramCounts, 15)
barplot(bigramCounts$n, names.arg=bigramCounts$bigrams)
```



Trigram frequencies

```
#Toeknize into trigrams
trigramsDf <- unnest_tokens(fullTextDf, trigrams, text, token='ngrams', n=3)
#Get count of trigrams
trigramCounts <- count(trigramsDf, trigrams, sort=TRUE)

#plot histogram of 15 of the most frequent trigrams
trigramCounts <- head(trigramCounts, 15)
barplot(trigramCounts$n, names.arg=trigramCounts$trigrams)
```



Building a word prediction model

‘Next word’ prediction will be done based on the bi-grams and tri-grams created above.

If one word is supplied by the user, the bi-grams list will be sorted on descending order of frequency and the second word of the first three rows will be offered as three choices for the next word.

If two words are supplied by the user, the tri-grams will be sorted on the descending order of their frequency and the third word of the top 3 rows will be offered as three choices for the next word.

If the user supplies more than two words, the last 2 words of the string supplied will be used to predict the next word.

In the above cases, if the word or words supplied are not found in the bi-gram and tri-gram list, a message will be displayed that there are not suggestions.

In addition to the above, a Shiny app will be developed to capture user input and display predictions.