Data Science Capstone Project

Milestone Report

Introduction

This Report represents the MileStone Report for the SwiftKey Project, which is the Capstone Project for the Data Science Specialization on Coursera. This Milestone Report gives an overview of the Loading, Cleaning and descriptive/ explorative Data Analysis of the Data.

Loading the data

The following Code will load the required Packages and the data into R. The dataset is available in different languages. In this Report we will take a look at the English dataset. There are three files: Twitter, News, Blogs.

```
library(tm)
library(stringi)
library(ngram)
library(stringr)

twitter_con <- file("C:/Users/maier/Desktop/Datensätze/SwiftKey/en_US.twitter.txt",
"r")
twitter <- readLines(twitter_con, skipNul = T, warn = F)
close(twitter_con)

blogs_con <- file("C:/Users/maier/Desktop/Datensätze/SwiftKey/en_US.blogs.txt",
"r")
blogs <- readLines(blogs_con, skipNul = T, warn = F)
close(blogs_con)

news_con <- file("C:/Users/maier/Desktop/Datensätze/SwiftKey/en_US.news.txt", "r")
news <- readLines(news_con, skipNul = T, warn = F)
close(news_con)
```

Descriptive Data Analysis

In the following we will take a first look at the datasets. The following code outputs basic summaries about each of the three datasets:

```
library(stringi)
# Number of lines
basic_summaries_length <- data.frame(c("Twitter", "Blogs", "News"), c(length(twitte
r), length(blogs), length(news)))
names(basic_summaries_length) <- c("Dataset", "Length")
basic_summaries_length
```

```
##   Dataset  Length
## 1 Twitter 2360148
## 2   Blogs  899288
## 3    News   77259
```

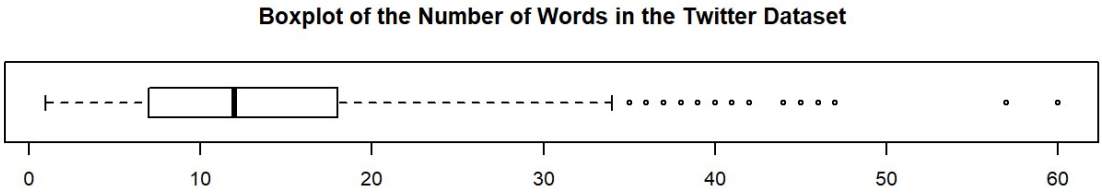
```
# Total Number of Words
twitter_words <- stri_count_words(twitter)
blogs_words <- stri_count_words(blogs)
news_words <- stri_count_words(news)
basic_summaries_sumwords <- data.frame(c("Twitter", "Blogs", "News"), c(sum(twitter
_words), sum(blogs_words), sum(news_words)))
names(basic_summaries_sumwords) <- c("Dataset", "Sum of Words")
basic_summaries_sumwords
```

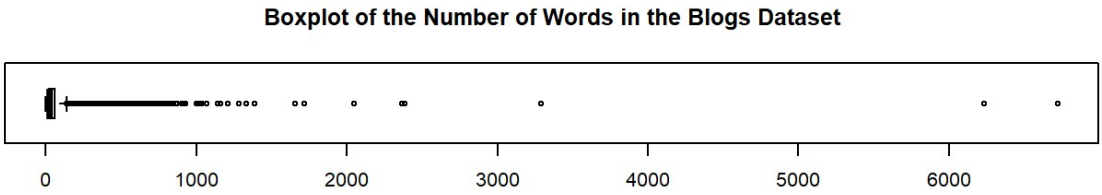
```
##   Dataset Sum of Words
## 1 Twitter     30218166
## 2   Blogs     38154238
## 3    News      2693898
```

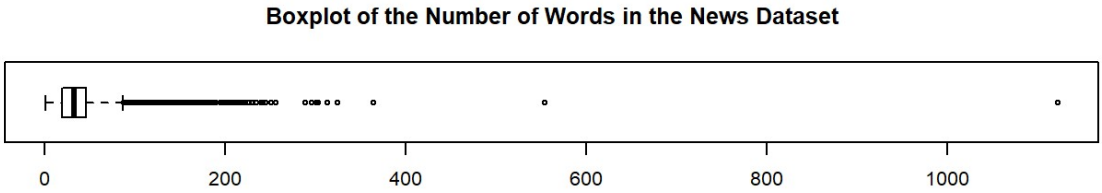
```
# Average Number of Words
basic_summaries_meanwords <- data.frame(c("Twitter", "Blogs", "News"), c(mean(twitt
er_words), mean(blogs_words), mean(news_words)))
names(basic_summaries_meanwords) <- c("Dataset", "Average Number of Words")
basic_summaries_meanwords
```

```
##   Dataset Average Number of Words
## 1 Twitter                12.80350
## 2   Blogs                42.42716
## 3    News                34.86840
```

```
par(mfrow = c(3, 1))
boxplot(twitter_words, main = "Boxplot of the Number of Words in the Twitter Datase
t",
        horizontal = T)
boxplot(blogs_words, main = "Boxplot of the Number of Words in the Blogs Dataset",
        horizontal = T)
boxplot(news_words, main = "Boxplot of the Number of Words in the News Dataset",
        horizontal = T)
```







Data Cleaning

In the next step we are going to combine the data into one dataset and only take a few samples from the three datasets because they are so big and we might run into memory or performance issues. Then we are going to clean the data, e.g. by removing unneccesary spaces or symbols.

```
library(tm)
```

```
## Loading required package: NLP
```

```
twitter_sample <- sample(twitter, length(twitter)*0.01)
blogs_sample <- sample(blogs, length(blogs)*0.01)
news_sample <- sample(news, length(news)*0.01)
dataset <- c(twitter_sample, blogs_sample, news_sample)
head(dataset)
```

```
## [1] "s/o to in da studio getting it in"
## [2] "I'm falling in love with diatonic talempong music from West Sumatra. Guess
I really am an"
## [3] "RT : Hey people that sit on planes not reading, listening to or watching an
ything: you look like serial killers."
## [4] "\": mike tyson fought abunch of nobodies\" lohh i have to teach u son"
## [5] "huh? Lol anyways he's gay."
## [6] "I could kinda get down to some Metallica right now... Weird."
```

```
length(dataset)
```

```
## [1] 33365
```

```
dataset_corpus <- VCorpus(VectorSource(dataset))
dataset_corpus <- tm_map(dataset_corpus, tolower)
dataset_corpus <- tm_map(dataset_corpus, removeWords, stopwords("en"))
dataset_corpus <- tm_map(dataset_corpus, removePunctuation)
dataset_corpus <- tm_map(dataset_corpus, removeNumbers)
dataset_corpus <- tm_map(dataset_corpus, stripWhitespace)
dataset_corpus <- tm_map(dataset_corpus, PlainTextDocument)
```

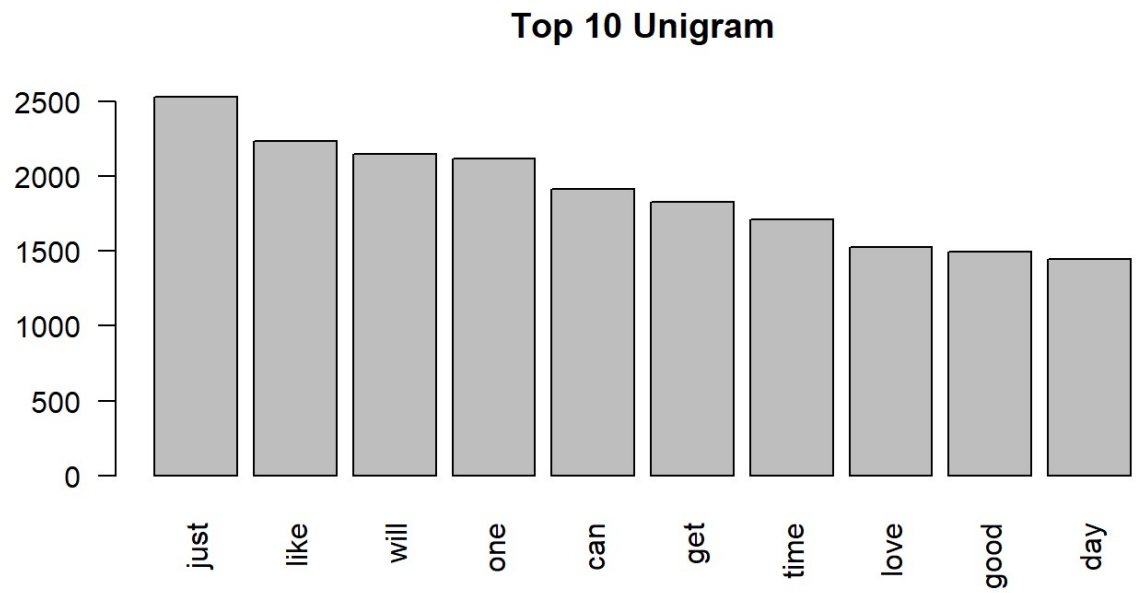
Modeling

In this chapter we are going to look at 1-, 2 and 3-gram models. With these models we will look at the most frequently occuring words or words combinations.

```
library(ngram)
library(stringr)

dataset_ngram <- unlist(sapply(dataset_corpus, `[`, "content"))
dataset_ngram <- dataset_ngram[dataset_ngram != " "]
dataset_ngram <- dataset_ngram[dataset_ngram != ""]
dataset_ngram <- dataset_ngram[sapply(dataset_ngram, wordcount) > 0]
dataset_ngram <- str_trim(dataset_ngram, "both")

par(mar=c(10,4,4,1)+.1)

ngram_1 <- ngram(dataset_ngram, n = 1)
ngram_1_table <- get.phrasetable(ngram_1)
ngram_1_table <- ngram_1_table[ngram_1_table[[1]] != "", ]
head(ngram_1_table)
```

```
##   ngrams freq        prop
## 1  just  2533 0.006756847
## 2  like  2236 0.005964591
## 3  will  2153 0.005743186
## 4   one  2119 0.005652491
## 5   can  1913 0.005102980
## 6   get  1828 0.004876240
```

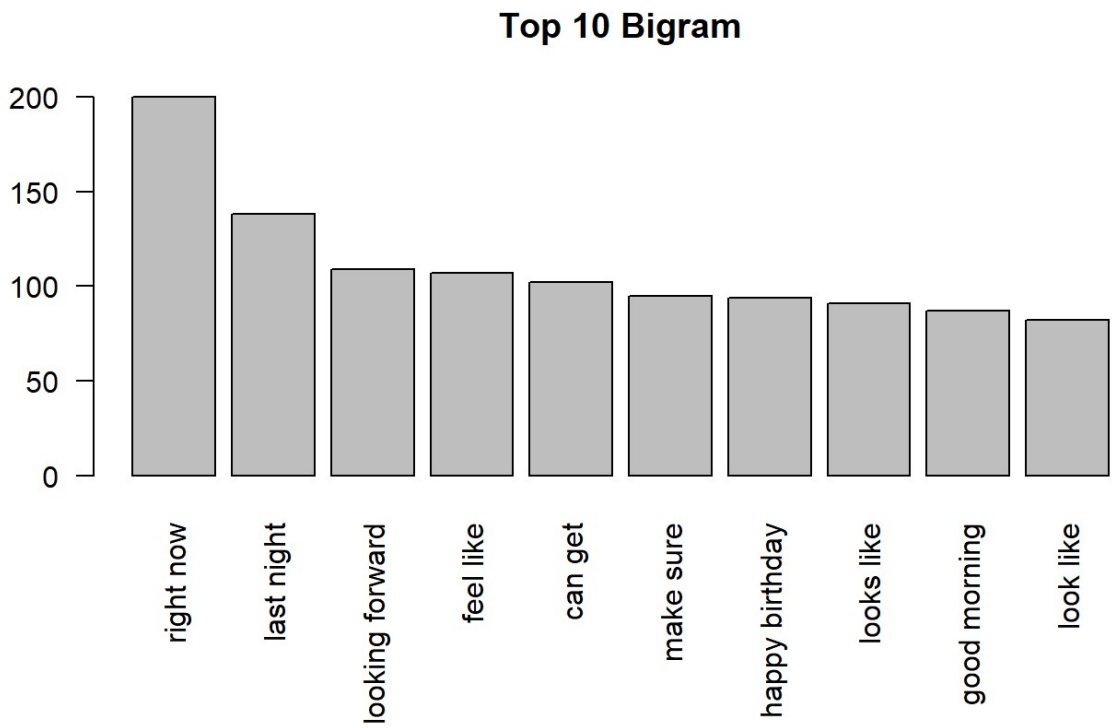
```
barplot(ngram_1_table[1:10, 2], names.arg = ngram_1_table[1:10, 1], main = "Top 10
Unigram", las = 2)
```




```
dataset_ngram <- dataset_ngram[sapply(dataset_ngram, wordcount) > 1]
ngram_2 <- ngram(dataset_ngram, n = 2)
ngram_2_table <- get.phrasetable(ngram_2)
ngram_2_table <- ngram_2_table[ngram_2_table[[1]] != " ", ]
head(ngram_2_table)
```

```
##             ngrams freq         prop
## 1       right now   200 0.0005854681
## 2      last night   138 0.0004039730
## 3 looking forward   109 0.0003190801
## 4       feel like   107 0.0003132254
## 5         can get   102 0.0002985887
## 6       make sure    95 0.0002780973
```

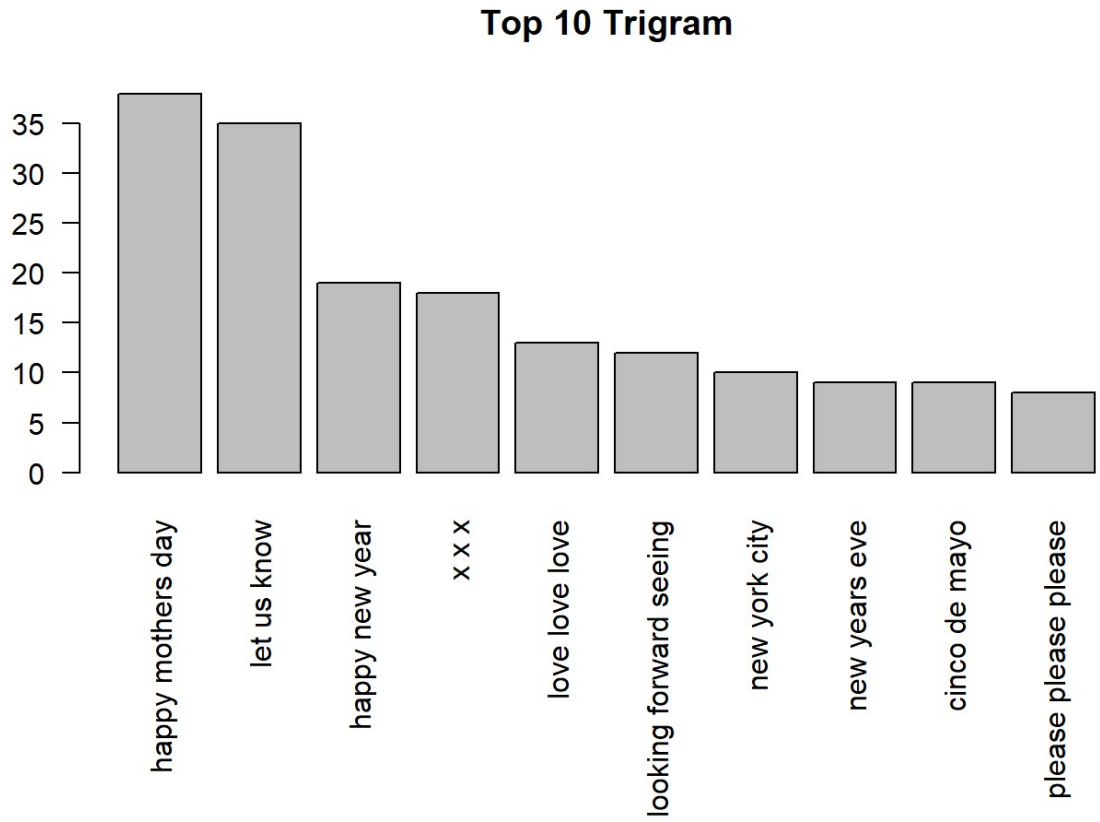
```
barplot(ngram_2_table[1:10, 2], names.arg = ngram_2_table[1:10, 1], main = "Top 10
Bigram", las = 2)
```




```
dataset_ngram <- dataset_ngram[sapply(dataset_ngram, wordcount) > 2]
ngram_3 <- ngram(dataset_ngram, n = 3)
ngram_3_table <- get.phrasetable(ngram_3)
ngram_3_table <- ngram_3_table[ngram_3_table[[1]] != "", ]
head(ngram_3_table)
```

```
##                     ngrams freq         prop
## 1      happy mothers day    38 1.228199e-04
## 2            let us know    35 1.131236e-04
## 3         happy new year    19 6.140997e-05
## 4                  x x x    18 5.817787e-05
## 5         love love love    13 4.201735e-05
## 6 looking forward seeing    12 3.878525e-05
```

```
barplot(ngram_3_table[1:10, 2], names.arg = ngram_3_table[1:10, 1], main = "Top 10
Trigram", las = 2)
```



Conclusion

In this report we showed the Loading and Cleaning of the dataset as well as some descriptive and explorative data analysis. We finished the report with some first and simple modeling. In the next step we will build the final prediction model and finally deploy the model on a shiny-app. The final model will consist of some n-gram model. It will take all prior known words up to a maximum number (like the mean word-count of a sentence) and use that number of words as the n for the n-gram-model. This way the algorithm would take into account the given number of words but doesn't use too many words (if the user already typed more sentences). The App will need an inpzt field so the user can input the text as well as afew label fields to show the predictions.