Uni-Gram Analysis and Data Preprocessing

For a “Predictive Text Product”

Aliakbar Safilian[*]

May 9, 2019

Contents

1 Introduction

1.1 The project

Around the world, people are spending an increasing amount of time on their mobile devices for email, social networking, banking and a whole range of other activities. However, typing on mobile devices can be a serious pain.

In this project, we aim at building a predictive text product that makes it easier for people to type. To this end, we start with analyzing a large corpus of text documents to discover the structure in the data and see how words are put together. The process involves cleaning and analyzing text data, then building and sampling from a predictive text model. Finally, we will build a predictive text shiny app.

This project is the capstone project for the Data Sciene Specialization offered by John Hopkins University. The data for this project has been provided by the Swiftkey company. SwiftKey has built a smart keyboard for mobile devices. One cornerstone of their smart keyboard is predictive text models. The basic training data can be found here. The data is from a corpus called HC Corpora. We may need to collect/use other data during the project.

There are four different databases, each for one specific language. The languages include German, English, Finnish, and Russian. In this project, we deal with the English database. There are the following textual files in the English database:

[*]a.a.safilian@gmail.com

- `en_us.blogs.txt`
- `en_us.news.txt`
- `en_us.twitter.txt`

Some information of the raw data is represented in the following table:

Table 1: Raw Data - Information

	Data	Size (mg)	Lines	Words (million)	Min/Max Chars in Lines
data_us_blogs.txt	Blogs	210.16	899288	+37	1 - 40833
data_us_twitter.txt	Twitter	167.11	2360148	+30	2 - 213
data_us_news.txt	News	205.81	1010242	+34	1 - 11384

1.2 The current report

In the current report, we analyze the databases, and we remove unimportant lines and those that include some profanity expressions/words. Unimportant lines are the lines that, excluding separators, numbers, punctuations, and stopwords, are empty.

We tokenize/clean a given text data in several levels as follow (each of which with and without prfanity expressions):

1. Words excluding seperators, numbers, hyphens, punctuations, stopwords, and sympbols;
2. Profanity words/expressions;
3. Words excluding seperators, numbers, hyphens, punctuations, stopwords, symbols, and profanity expressions;
4. Stemming data.

We will do some exploratory analysis on the cleaned data as well.

The structure of the rest of the report is as follows: In Sec. 2, we tokenize our given text data in several levels. Then, we do some analysis on the tokens. Sec.3 does the cleaning tasks on the data. The scripts used in this report to generate results can be found in Appendix.

2 Tokens

In this section, we tokenize our given text data in several levels. In Sec 2.1, Sec 2.2, and Sec 2.3, we deal with the blogs, twitter, and news databases, respectively. In Sec 2.4, we consider the integrated data, i.e., the whole corpus. In each section, we first load the data and do some summary on the data. Then, we tokenize the data in several different levels, and analyze their frequency distributions. We then take a look at the profanity expressions in the data. Finally, we clean the stop-words, punctuations, hyphens, symbols, URLs, and profanities out the data. We store the tokenized clean data so that we can use it in next steps.

2.1 Blogs

The blogs dataset contains **899288** lines (observations), and the number of characters in a line ranges between **1** and **40833**. In the following, we summarize the result of tokenizing the data.

Excluding the *seperators, numbers, punctuations, hyphens, URLs, symbols, and stop-words*: (Note that we also *stem the words*.)

- The number of words in a line of blogs ranges between **0** and **3969**.
- There is only **1** line with 3969 number of words.

- About **0.32%** of the observations (**2889** lines) are empty lines.
- The number of profanities in a line ranges between **0** and **24**.[1]
- There is only **1** line with maximum number of profanity expressions.
- About **2.13%** of the observations (**19110** lines) include some profanities.
- The number of unique profanities appearing in blogs is **110**.[2]

A Word-Cloud for the blogs data, excluding profanities, hyphens, URLs, symbols, stop-words, and numbers, is represented in Fig. 1.

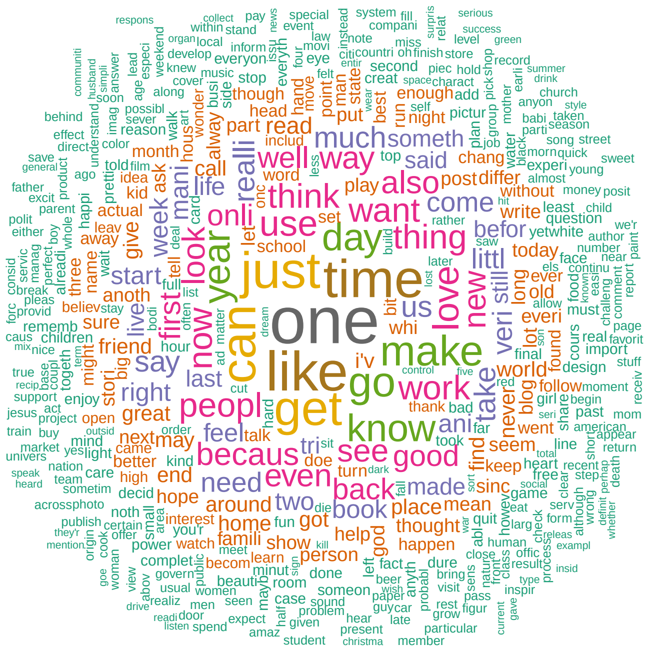

Figure 1: A Word-Cloud for the blogs dataset (excl. stopwords, numbers, profanities)

Fig. 2 represents the top 30 most-frequent words appearing in the blogs data set.

[1]We have used https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words as a reference of profanities.

[2]Our dataset of profanities include 376 items.

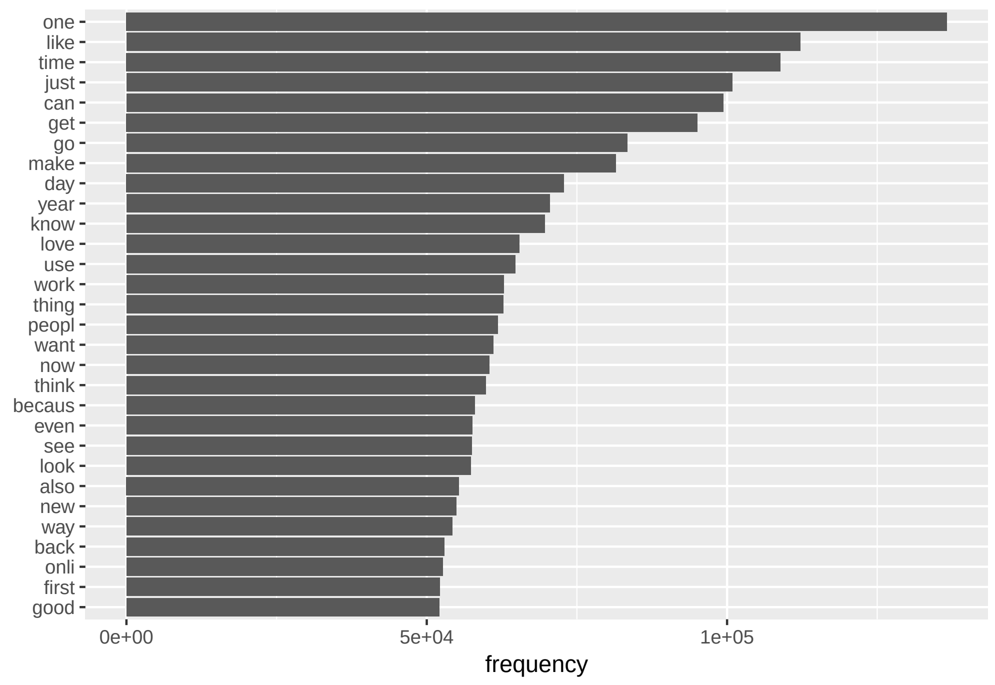

Figure 2: Top 30 frequent words in the blogs data (excl. stop-words, numbers, profanities)

2.2 Twitter

The twitter dataset contains **2360148** lines, and the number of characters in a line ranges between **2** and **213**. In the following, we summarize the result of tokenizing the data.

Excluding the *seperators, numbers, punctuations, hyphens, URLs, symbols, and stop-words*: (Note again that we also *stem the words*.)

- The number of words in a line ranges between **0** and **48**.
- There is only **1** line with 48 number of words.
- About **0.32%** of the observations (**7464** lines) are empty lines.
- The number of profanities in a line ranges between **0** and **35**.
- There is only **1** line with maximum number of profanities.
- About **3.79%** of the observations **89362** lines include some profanities.
- The number of unique profanities appearing in the twitter dataset is **185**.

A Word-Cloud for the twitter data, excluding profanities, stop-words, hyphens, URLs, symbols, and numbers, is represented in Fig. 3.

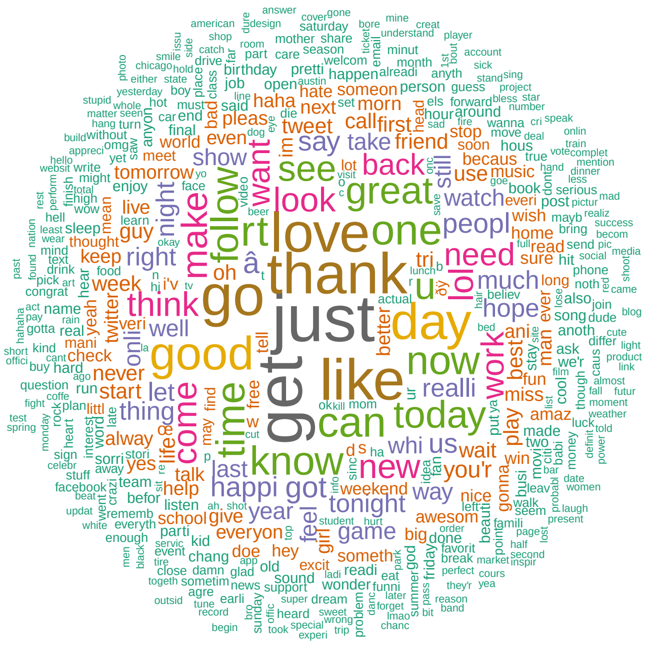

Figure 3: A Word-Cloud for the twitter dataset (excl. stopwords, numbers, profanities)

Fig. 4 represents the top 30 most-frequent words appearing in the twitter data set.

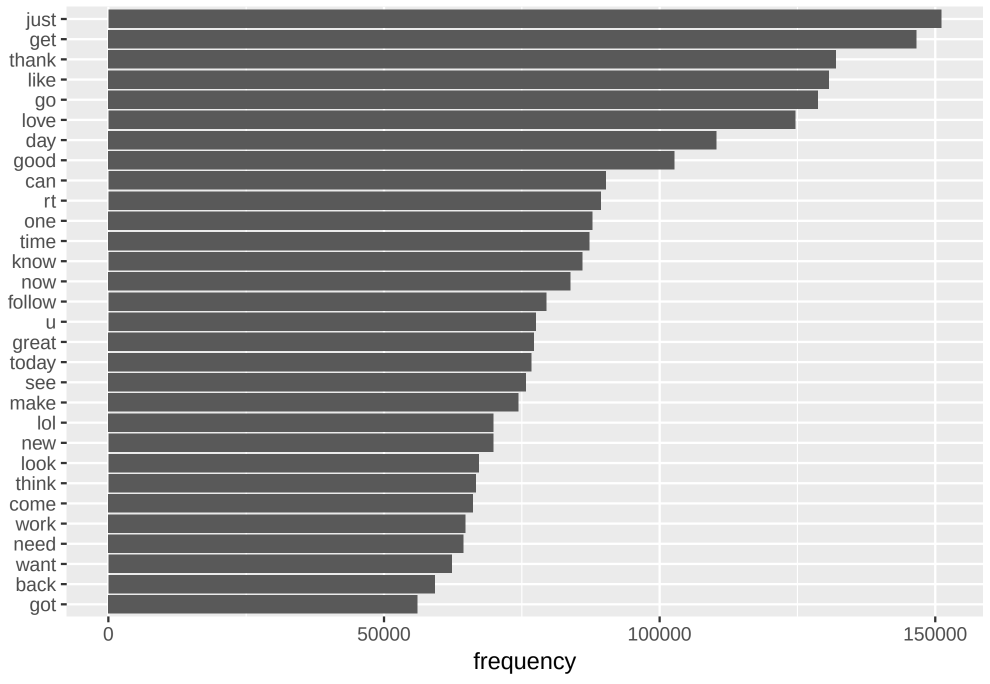

Figure 4: Top 30 frequent words in the twitter data (excl. stop-words, numbers, profanities)

2.3 News

The news dataset contains **1010242** lines, and the number of characters in a line ranges between **1** and **11384**. In the following, we summarize the result of tokenizing the data.

Excluding the *seperators, numbers, punctuations, hyphens, URLs, symbols, and stop-words*: (Note again that we work on *stemmed* tokens.)

- The number of words in a line of news ranges between **0** and **1315**.
- There is only **1** line with 1315 number of words.
- About **0.13%** of the observations (**1314** lines) are empty lines.
- The number of profanity expressions/words in a line ranges between **0** and **12**.
- There is only **1** line with maximum number of profanity expressions.
- About **0.89%** of the observations (**9035** lines) include some profanity.
- The number of unique profanities appearing in news is **130**.

A Word-Cloud for the news data, excluding profanities, stop-words, hyphens, URLs, symbols, and numbers, is represented in Fig. 5.

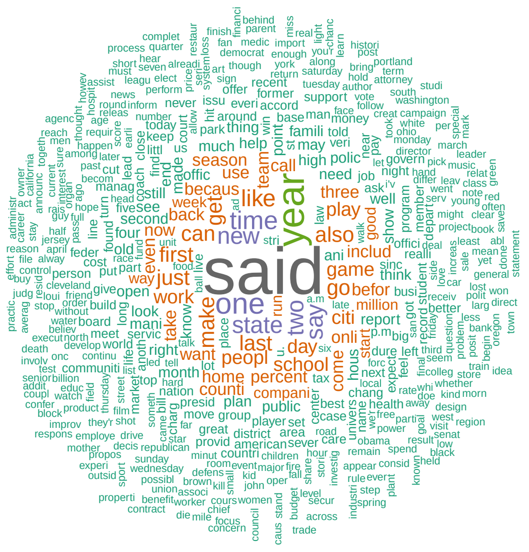

Figure 5: A Word-Cloud for the news dataset (excl. stopwords, numbers, profanities)

Fig. 6 represents the top 30 most-frequent words appearing in the news data set.

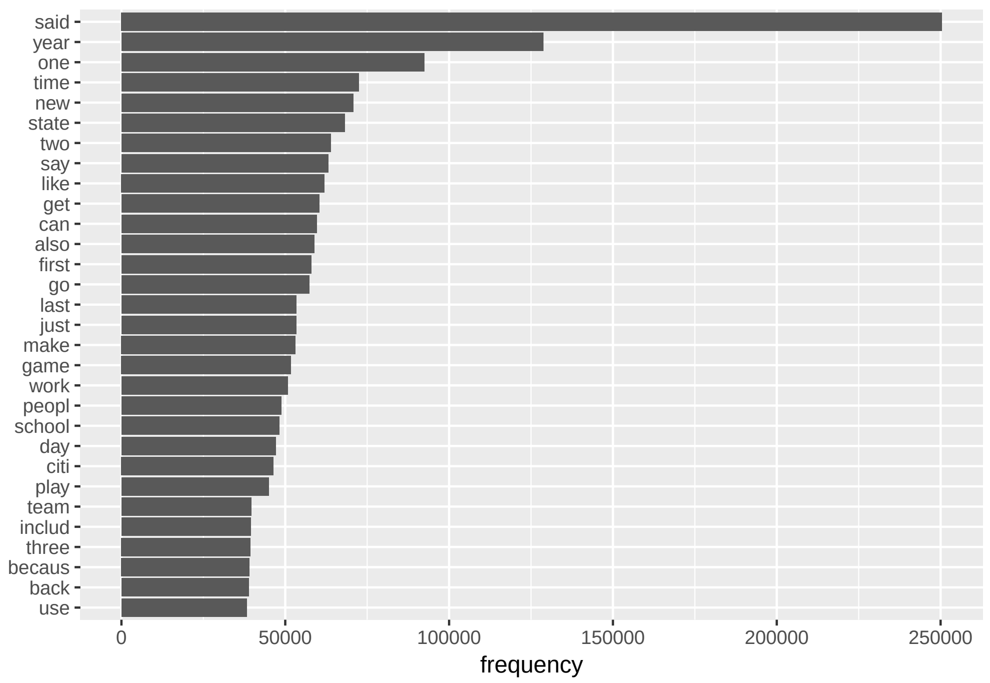

Figure 6: Top 30 frequent words in the news data (excl. stop-words, numbers, profanities)

2.4 The Curpos

In this section, we consider the integrated data. Excluding the *seperators, numbers, punctuations, URLs, hyphens, symbols, and stop-words*, the summary of the data is as follows: (Note again that we *stem the words.*)

- The number of words in a line of the integrated data ranges between **0** and **3969**.
- There is only **1** line with 3969 number of words.
- About **0.27%** (**11667** lines) of the observations are empty lines.
- The number of profanities in a line of the corpus ranges between **0** and **35**.
- There is only **1** line with maximum number of profanity expressions.
- About **2.76%** of the observations (i.e., **117880** lines) include some profanities.
- The number of unique profanities appearing in the dataset is **203**.

A Word-Cloud for the integrated data, excluding profanities, stop-words, URLs, symbols, and numbers, is represented in Fig. 7.

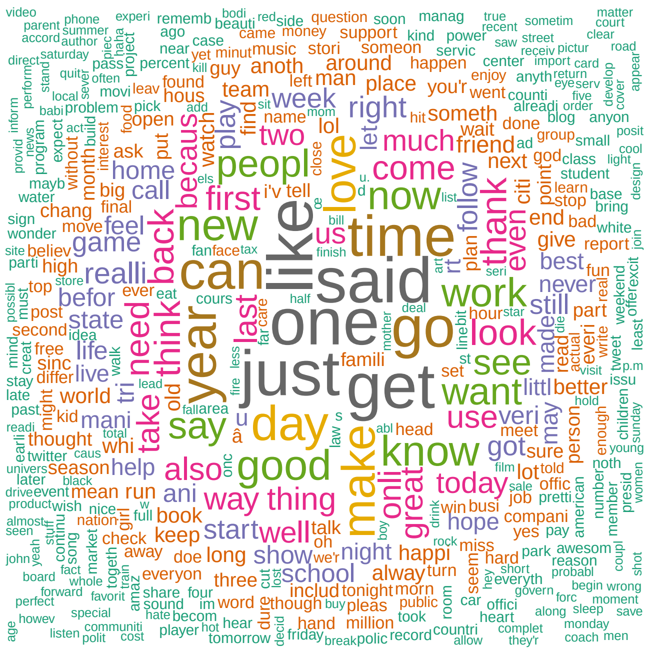

Figure 7: A Word-Cloud for corpus

Fig. 8 represents the top 30 most-frequent words appearing in the corpus.

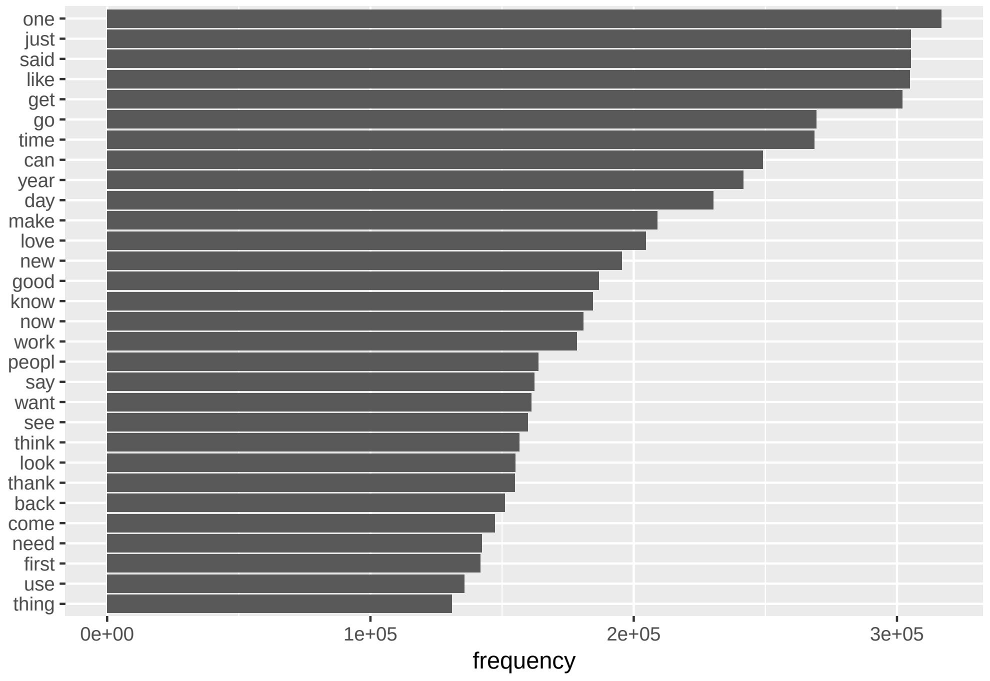

Figure 8: Top 30 frequent words in the integrated data (excl. stop-words, numbers, profanities)

According to the document frequencies of the integrated data:

- **589** unique words (out of 519127 features) cover 50% of all word instances in the integrated data.
- **7069** unique words (out of 519127 features) cover 90% of all word instances in the integrated data.
- $\mathbf{1.5785 \times 10^4}$ unique words (out of 519127 features) cover 95% of all word instances in the integrated data.
- $\mathbf{1.14575 \times 10^5}$ unique words (out of 519127 features) cover 99% of all word instances in the integrated data.
- There are **424223** words with less than 5 frequencies in the corpus, including 297946 words with only 1 frequency.

We are going to clean the words with less frequency out of the data. That is we keep at least the top **7069** features/words to cover at least 90% of all word instances in the data. (See The whole Corpus for the corresponding scripts.)

3 Conclusion

We started with a very large corpus with millions of words. We tokenized the data and performed some frequency analysis on the words. We cleaned out *profanities, punctuations, numbers, symbols, urls, separators, and words with less frequency* from the data. Moreover, we stemmed the words in the data. Also, we stored the corresponding cleaned tokenized data to be used in the next steps.

The next step is to do analysis on *n-grams*.

Appendix

Packages/Sources/Functions

```r
require(quanteda)
require(readr)
require(stopwords)
require(tidytext)
require(dplyr)
require(ngram)
library(knitr)
library(kableExtra)
require("ggplot2")
source("loading.R")
```

```r
# a helper function for renaming items
nameKeys <- function(i, vec, txt){
        vec_sub <- substr(vec, i, sapply(vec, nchar))
        vec <- paste(txt, vec_sub, sep = "")
}

# find the
findMany <- function(df, p){
        cri <- sum(df$frequency) * p
        com <- 0
        ind <- 1
        for(i in 1:dim(df)[1]){
                com <- com + df$frequency[i]
                if(com >= cri){
                        break
                }
                else{
                        ind <- ind + 1
                }
        }
        ind
}
```

Table: Raw Data Info

```
# Raw data info
blogs <- loadtext("blogs")
twitter <- loadtext("twitter")
news <- loadtext("news")

size_blogs <- round((file.info("rawData/en_US/en_US.blogs.txt")$size)/1000000, 2)
size_twitter <- round((file.info("rawData/en_US/en_US.twitter.txt")$size)/1000000, 2)
size_news <- round((file.info("rawData/en_US/en_US.news.txt")$size)/1000000, 2)

lines_blogs <- length(blogs)
lines_twitter <- length(twitter)
lines_news <- length(news)

words_blogs <- round(wordcount(blogs)/1000000, 0)
words_twitter <- round(wordcount(twitter)/1000000, 0)
words_news <- round(wordcount(news)/1000000, 0)

length_blogs <- sapply(blogs, nchar)
max_length_blogs <- max(length_blogs)
min_length_blogs <- min(length_blogs)
min_max_chars_blogs <- paste(as.character(min_length_blogs), " - ",
                             as.character(max_length_blogs), sep = "")

length_twitter <- sapply(twitter, nchar)
max_length_twitter <- max(length_twitter)
min_length_twitter <- min(length_twitter)
min_max_chars_twitter <- paste(as.character(min_length_twitter), " - ",
                               as.character(max_length_twitter), sep = "")

length_news <- sapply(news, nchar)
max_length_news <- max(length_news)
min_length_news <- min(length_news)
min_max_chars_news <- paste(as.character(min_length_news), " - ",
                            as.character(max_length_news), sep = "")

info_blogs <- c("Blogs", size_blogs, lines_blogs,
                paste("+", as.character(words_blogs), sep = ""),
                min_max_chars_blogs)
```

```r
info_twitter <- c("Twitter", size_twitter, lines_twitter,
                  paste("+", as.character(words_twitter), sep = ""),
                  min_max_chars_twitter)

info_news <- c("News", size_news, lines_news,
               paste("+", as.character(words_news), sep = ""),
               min_max_chars_news)

raw_info <- as.data.frame(rbind(info_blogs, info_twitter, info_news))
colnames(raw_info) = c("Data", "Size (mg)", "Lines", "Words (million)", "Min/Max Chars in Lines")
rownames(raw_info) = c("data_us_blogs.txt", "data_us_twitter.txt", "data_us_news.txt")


kable(raw_info, "latex", booktabs = T,
      caption = "Raw Data - Information") %>%
        kable_styling(latex_options = "hold_position")
```

Blogs

```r
# Tokenizing Blogs
tokens_blogs <- tokens(blogs,
                       remove_numbers = TRUE,
                       remove_hyphens = TRUE,
                       remove_url = TRUE,
                       remove_symbols = TRUE,
                       remove_separators = TRUE,
                       remove_punct = TRUE,
                       remove_twitter = TRUE)

names(tokens_blogs) <- nameKeys(i = 5,
                                vec = names(tokens_blogs),
                                txt = "blog")

tokens_blogs <- tokens_tolower(tokens_blogs)

tokens_blogs <- tokens_wordstem(tokens_blogs, language = "english")

tokens_wostp_blogs <- tokens_remove(tokens_blogs,
                                    pattern = stopwords('en'))

num_tokens_wostp_blogs <- sapply(tokens_wostp_blogs, length)
#Maximum number of tokens w/o seps, nums, punctuations, stops in a line
max_num_tokens_wostp_blogs <- max(num_tokens_wostp_blogs)
#Minimum number of tokens w/o seps, nums, punctuations, stops in a line
min_num_tokens_wostp_blogs <- min(num_tokens_wostp_blogs)
```

```r
# Profanity expressions in blogs
bad_words <- read_lines("rawData/bad_words.txt")

tokens_bad_blogs <- tokens_select(tokens_wostp_blogs,
                                  pattern = bad_words,
                                  selection = "keep")
```

```r
num_tokens_bad_blogs <- sapply(tokens_bad_blogs, length)
#Maximum number of bad tokens in a line
max_num_tokens_bad_blogs <- max(num_tokens_bad_blogs)
#Minimum number of bad tokens in a line
min_num_tokens_bad_blogs <- min(num_tokens_bad_blogs)
blogs_bad_dfm <- dfm(tokens_bad_blogs)
```

```r
#tokens-wostpbad-blogs

tokens_wostp_blogs_clean <- tokens_remove(tokens_wostp_blogs,
                                          pattern = bad_words)
#Worldcloud - blogs - wosepnumpnct
set.seed(1000)
blogs_wostpbad_dfm <- dfm(tokens_wostp_blogs_clean)
textplot_wordcloud(blogs_wostpbad_dfm, random_order = FALSE,
                   rotation = .25,
                   color = RColorBrewer::brewer.pal(8,"Dark2"))
```

```r
# Frequency Plot - blogs
stat_blogs_wostpbad <- textstat_frequency(blogs_wostpbad_dfm)
# visualization with ggplot
ggplot(stat_blogs_wostpbad[1:30, ],
       aes(x = reorder(feature, frequency),
           y = frequency))  +
        geom_col() +
        xlab(NULL) +
        coord_flip()
```

Twitter

```r
tokens_twitter <- tokens(twitter,
                         remove_numbers = TRUE,
                         remove_hyphens = TRUE,
                         remove_url = TRUE,
                         remove_symbols = TRUE,
                         remove_separators = TRUE,
                         remove_punct = TRUE,
                         remove_twitter = TRUE)

names(tokens_twitter) <- nameKeys(i = 5,
                                  vec = names(tokens_twitter),
                                  txt = "twitter")

tokens_twitter <- tokens_tolower(tokens_twitter)

tokens_twitter <- tokens_wordstem(tokens_twitter, language = "english")

tokens_wostp_twitter <- tokens_remove(tokens_twitter,
                                      pattern = stopwords('en'))

num_tokens_wostp_twitter <- sapply(tokens_wostp_twitter, length)
#Maximum number of tokens w/o seps, nums, punctuations, stops in a line
```

```r
max_num_tokens_wostp_twitter <- max(num_tokens_wostp_twitter)
#Minimum number of tokens w/o seps, nums, punctuations, stops in a line
min_num_tokens_wostp_twitter <- min(num_tokens_wostp_twitter)
```

```r
# Profanity expressions in twitter
tokens_bad_twitter <- tokens_select(tokens_wostp_twitter,
                                   pattern = bad_words,
                                   selection = "keep")
num_tokens_bad_twitter <- sapply(tokens_bad_twitter, length)
#Maximum number of bad tokens in a line
max_num_tokens_bad_twitter <- max(num_tokens_bad_twitter)
#Minimum number of bad tokens in a line
min_num_tokens_bad_twitter <- min(num_tokens_bad_twitter)
twitter_bad_dfm <- dfm(tokens_bad_twitter)
```

```r
#tokens-wostpbad-twitter
tokens_wostp_twitter_clean <- tokens_remove(tokens_wostp_twitter,
                                            pattern = bad_words)
#Worldcloud - twitter - wosepnumpnct
set.seed(1000)
twitter_wostpbad_dfm <- dfm(tokens_wostp_twitter_clean)
textplot_wordcloud(twitter_wostpbad_dfm, random_order = FALSE,
                   rotation = .25,
                   color = RColorBrewer::brewer.pal(8,"Dark2"))
```

```r
# Frequency Plot - twitter
stat_twitter_wostpbad <- textstat_frequency(twitter_wostpbad_dfm)
# Visualization with ggplot
ggplot(stat_twitter_wostpbad[1:30, ],
       aes(x = reorder(feature, frequency),
            y = frequency))  +
        geom_col() +
        xlab(NULL) +
        coord_flip()
```

News

```r
tokens_news <- tokens(news,
                      remove_numbers = TRUE,
                      remove_hyphens = TRUE,
                      remove_url = TRUE,
                      remove_symbols = TRUE,
                      remove_separators = TRUE,
                      remove_punct = TRUE,
                      remove_twitter = TRUE)

names(tokens_news) <- nameKeys(i = 5,
                               vec = names(tokens_news),
                               txt = "news")

tokens_news <- tokens_tolower(tokens_news)

tokens_news <- tokens_wordstem(tokens_news, language = "english")
```

```r
tokens_wostp_news <- tokens_remove(tokens_news,
                                    pattern = stopwords('en'))

num_tokens_wostp_news <- sapply(tokens_wostp_news, length)
#Maximum number of tokens w/o seps, nums, punctuations, stops in a line
max_num_tokens_wostp_news <- max(num_tokens_wostp_news)
#Minimum number of tokens w/o seps, nums, punctuations, stops in a line
min_num_tokens_wostp_news <- min(num_tokens_wostp_news)
```

```r
# Profanity expressions in news
tokens_bad_news <- tokens_select(tokens_wostp_news,
                                 pattern = bad_words,
                                 selection = "keep")
num_tokens_bad_news <- sapply(tokens_bad_news, length)
#Maximum number of bad tokens in a line
max_num_tokens_bad_news <- max(num_tokens_bad_news)
#Minimum number of bad tokens in a line
min_num_tokens_bad_news <- min(num_tokens_bad_news)
news_bad_dfm <- dfm(tokens_bad_news)
```

```r
#tokens-wostpbad-news
tokens_wostp_news_clean <- tokens_remove(tokens_wostp_news,
                                         pattern = bad_words)
#Worldcloud - news - wosepnumpnct
set.seed(1000)
news_wostpbad_dfm <- dfm(tokens_wostp_news_clean)
textplot_wordcloud(news_wostpbad_dfm, random_order = FALSE,
                   rotation = .25,
                   color = RColorBrewer::brewer.pal(8,"Dark2"))
```

```r
# Frequency Plot - news
stat_news_wostpbad <- textstat_frequency(news_wostpbad_dfm)
# Visualization with ggplot
ggplot(stat_news_wostpbad[1:30, ],
       aes(x = reorder(feature, frequency),
           y = frequency))  +
        geom_col() +
        xlab(NULL) +
        coord_flip()
```

The Corpus

```r
length_all <- length(blogs) + length(news) + length(twitter)

tokens_all <- append(append(tokens_blogs, tokens_twitter), tokens_news)


tokens_wostp_all <- append(append(tokens_wostp_blogs,
                                  tokens_wostp_twitter),
                           tokens_wostp_news)

num_tokens_wostp_all <- sapply(tokens_wostp_all, length)
```

```r
#Maximum number of tokens w/o seps, nums, punctuations, stops in a line
max_num_tokens_wostp_all <- max(num_tokens_wostp_all)
#Minimum number of tokens w/o seps, nums, punctuations, stops in a line
min_num_tokens_wostp_all <- min(num_tokens_wostp_all)
```

```r
# Profanity expressions in all
tokens_bad_all <- tokens_select(tokens_wostp_all,
                                pattern = bad_words,
                                selection = "keep")
num_tokens_bad_all <- sapply(tokens_bad_all, length)
#Maximum number of bad tokens in a line
max_num_tokens_bad_all <- max(num_tokens_bad_all)
#Minimum number of bad tokens in a line
min_num_tokens_bad_all <- min(num_tokens_bad_all)
all_bad_dfm <- dfm(tokens_bad_all)
```

```r
#tokens-wostpbad-all
tokens_wostp_all_clean <- tokens_remove(tokens_wostp_all,
                                             pattern = bad_words)
#Worldcloud - all - wosepnumpnct
set.seed(1000)
all_wostpbad_dfm <- dfm(tokens_wostp_all_clean)
textplot_wordcloud(all_wostpbad_dfm, random_order = FALSE,
                    rotation = .25,
                    color = RColorBrewer::brewer.pal(8,"Dark2"))
```

```r
# Frequency Plot - all
stat_all_wostpbad <- textstat_frequency(all_wostpbad_dfm)
# Visualization with ggplot
ggplot(stat_all_wostpbad[1:30, ],
       aes(x = reorder(feature, frequency),
            y = frequency))  +
        geom_col() +
        xlab(NULL) +
        coord_flip()
```

```r
# Getting frequency 50, 90, ...%
df_stat_all <- as.data.frame(stat_all_wostpbad)
freq1_all <- df_stat_all %>% filter(frequency <= 1)
freqless5_all <- df_stat_all %>% filter(frequency <= 5)
num_words_50per <- findMany(df_stat_all, 0.5)
num_words_90per <- findMany(df_stat_all, 0.9)
num_words_95per <- findMany(df_stat_all, 0.95)
num_words_99per <- findMany(df_stat_all, 0.99)
```

```r
# Profanity filtering --> tokens_clean

tokens_clean <- tokens_remove(tokens_all,
                              pattern = bad_words,
                              padding = TRUE)
```

```r
# getting features to be excluded --> tokens_clean

# frequency criteria
freq_cri <- df_stat_all[num_words_90per, ]$frequency
```

```r
# features to be excluded
exc_feats <- df_stat_all %>%
        filter(frequency < freq_cri) %>%
        select(feature)

# clean less frequent tokens
tokens_clean <- tokens_remove(tokens_clean,
                              pattern = exc_feats,
                              padding = TRUE)
```

```r
# save cleaned all
saveRDS(tokens_clean,
        "cleanData/tokens/tokens_clean.rds")
```