Analysis Report: A Shiny Predictive Text Product

Aliakbar Safilian

May 13, 2019

Contents

1 Introduction

1.1 What?

In this project, we are going to build a *predictive text Shiny app*, i.e., a web-based version of products like Google Search and SwiftKey Keyboards. To this end, we start with analyzing a large corpus of text documents to discover the relationship between words. The process involves cleaning and analyzing text data, then building and sampling from a predictive text model. Finally, we will build our shiny app.

The data for this project has been provided by the Swiftkey company.[1] There are four different databases, each for one specific language. The languages include German, English, Finnish, and Russian. In this project, we deal only with the English database. There are the following textual files in the English database:

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

1.2 How?

In the current report, we preprocess and analyze the data. Our original corpus includes more than *4 million lines*, and over *100 million words*. Due to computational resource limitations, we random sub *sample* the data and get a fraction (15%) of the data.

Next, we perform the following *preprossing steps* on the sample data:

- lower-case conversion
- removing hyphens
- removing twitter and other symbols
- removing separators (white-spaces)
- removing punctuations

[1] 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.

- removing numbers
- removing profanities
- removing non-English words

We then extract *uni-grams* (words), *bi-grams* (two consecutive words), and *tri-grams* (three consecutive words) from the clean data, and represent several interesting results analyzing them. We perform some exploratory analysis to understand the distributions of word frequencies.

The structure of the rest of the report is as follows: In Sec. 2,we tokenize and clean the data. Sec. 3, we analyze the bi- and tri-grams extracted from the clean data. The scripts used in this report to generate results can be found in Appendix.

2 Preprocessing and Uni-Grams

We first load the data, and get a general picture of the data:

Table 1: Raw Data - Information

Data	Size (mg)	Lines	Words	Range nchars	Avg nchars
Blogs	210.16	899288	~37,000,000	1 - 40833	229.99
Twitter	167.11	2360148	~30,000,000	2 - 213	68.8
News	205.81	1010242	~34,000,000	1 - 11384	201.16
Corpus	583.08	4269678	~102,000,000	1 - 40833	499.95

Note:
'Words': approximate in million
'Range nchars': range and avg number of chars in lines
'Avg nchars': range and avgerage number of chars in lines

As we see in the above table, the original corpus includes over 70 million words and over 100 million sentences. The main constraint in this project is the computational resource (i.e., time and memory). To alleviate this issue, we get a 10% random sample fraction of the data. The following table represents some information about the sample. Since the average number of characters have not changed much, the sample looks a reasonable one.

Table 2: Random Sampled Raw Data - Information

Data	Lines	Words	Range nchars	Avg nchars
Blogs	89928	~4,000,000	1 - 40833	231.09
Twitter	236014	~3,000,000	3 - 154	68.69
News	101024	~3,000,000	2 - 3555	201.76
Corpus	426966	~10,000,000	1 - 40833	501.54

Now, we extract the unigrams *excluding numbers, hyphens, URLs, separators, punctuations*, and (twitter) *symbols*. Moreover, we convert the words to lower-case. We also extract the *profanities* and *non-english* words in the corpuse. A summary of the results is represented in the following table.[2]

As we see above, about 0.14% and 0.19% of the words in the sample corpus is bad and non-English words, respecitvely. We clean the profanities and non-english words out the corpus.

The range of the frequency of words in our clean data is now between 1 and 0.48 million. We categorize them into 5 groups based on their frequencies in the corpus. See the following table, where it is shown that the most of the words are in the first category, i.e., the least frequent one.

[2]We have used https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words as a reference of profanities, whcih contains 376 items.

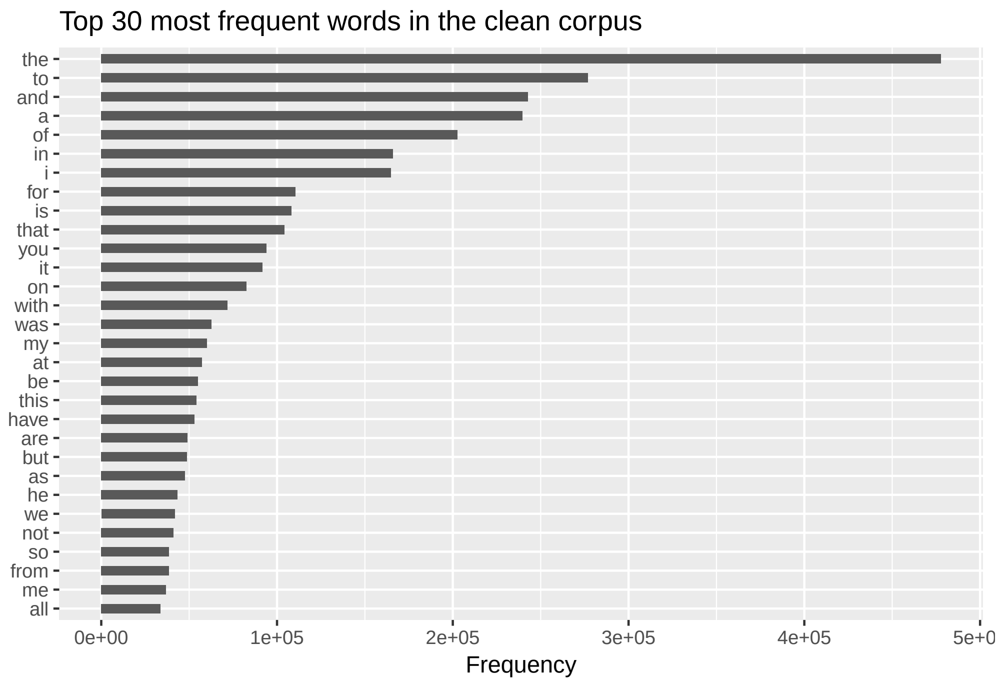
Table 3: Unigrams (Words, Profanities, Non-English Words) in the Corpus

Words	Unique Words	Profanities	Profanity	Non-English	Non-English
10096322	182467	14068	0.14 %	19622	0.19 %

Table 4: Frequency Groups - Unigrams

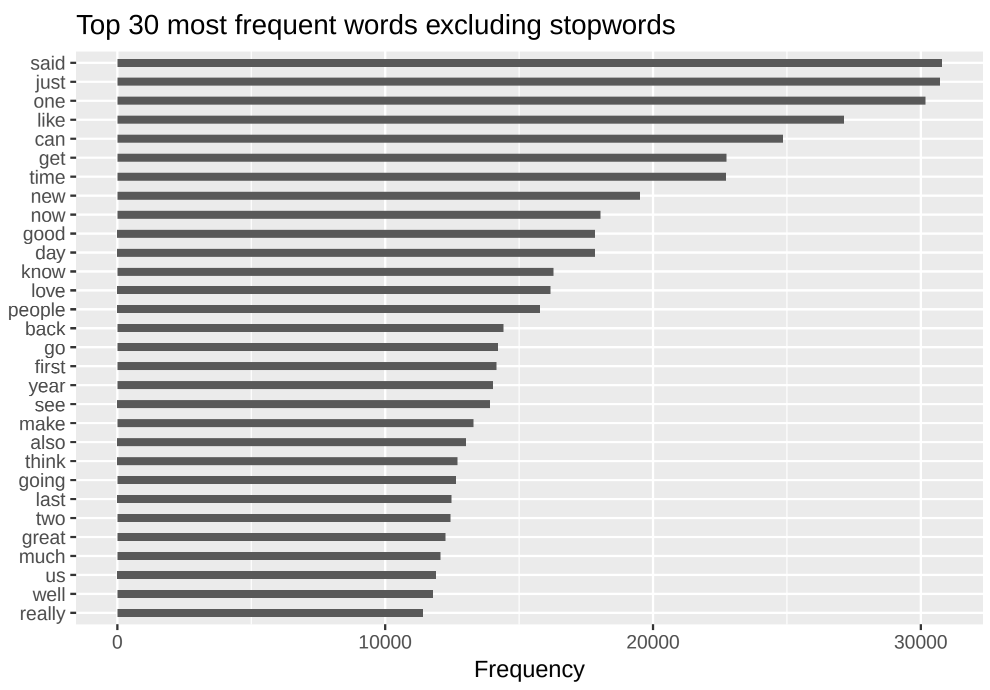
Category	Min Frequency	Max Frequency	Average Frequency	Words	Instances
1	1	94068	44.94	177317	7968523
2	104383	165987	130844.20	5	654221
3	202639	276809	240529.00	4	962116
5	477772	477772	477772.00	1	477772

The following figure represents the top 30 most frequenct words in the *clean* corpus with their frequencies.



As we see in the above frequency plot, the most frequent words in the sample corpus are *stop-words*. The following plot represents the 30 top frequenct words excluding the stop-words[3] A Word-Cloud for the data, excluding profanities, hyphens, URLs, symbols, stop-words, and numbers, follows it.

[3]However, note that we do not clean the stop-words out the unigrams.



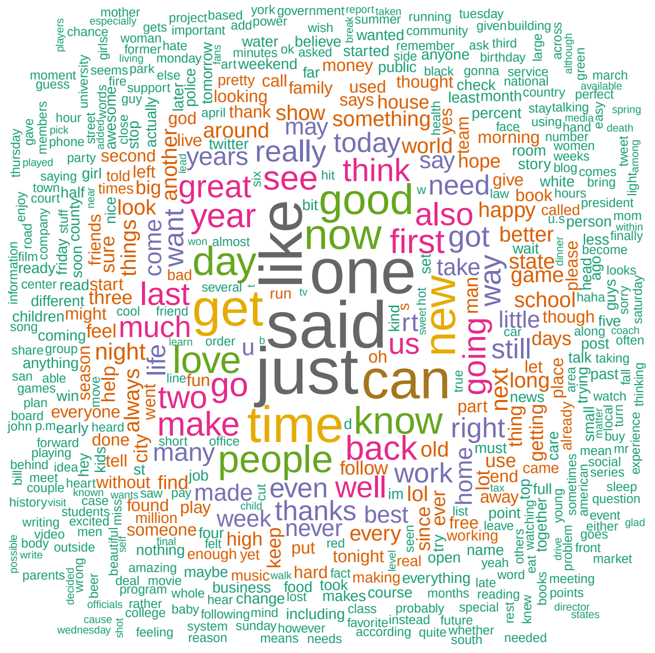
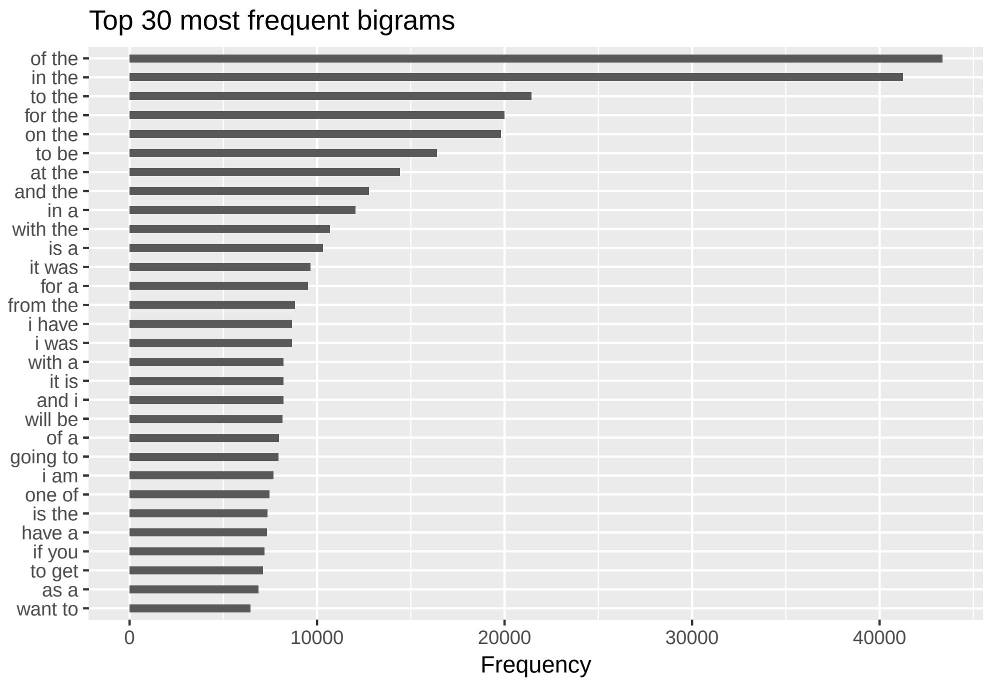


Figure 1: A Word-Cloud for unigrams (excl. stopwords, numbers, profanities, URLs, symbols)

3 Bi- and Tri-Grams

In this section, we extract the *bi-grams* and *tri-grams* from clean unigrams, and we analyze their frequencies. The following figure represents the 30 most frequent bigrams in our corpus.



The following table represents some information about the categorized version of bigrams based on their frequencies. Like uni-grams, the most of the bigrams are in the least frequent one.

Table 5: Frequency Groups - Bigrams

Category	Min Frequency	Max Frequency	Average Frequency	Words	Instances
1	1	8658	3.58	166676	9385280
2	8819	16395	11620.00	9	104580
3	19798	21438	20407.00	3	61221
5	41237	43349	42293.00	2	84586

The following figure represents the 30 most frequenct trigrams in our corpus. It is followed by a table which represents some information about the trigrams categorized into 10 groups based on their frequencies.

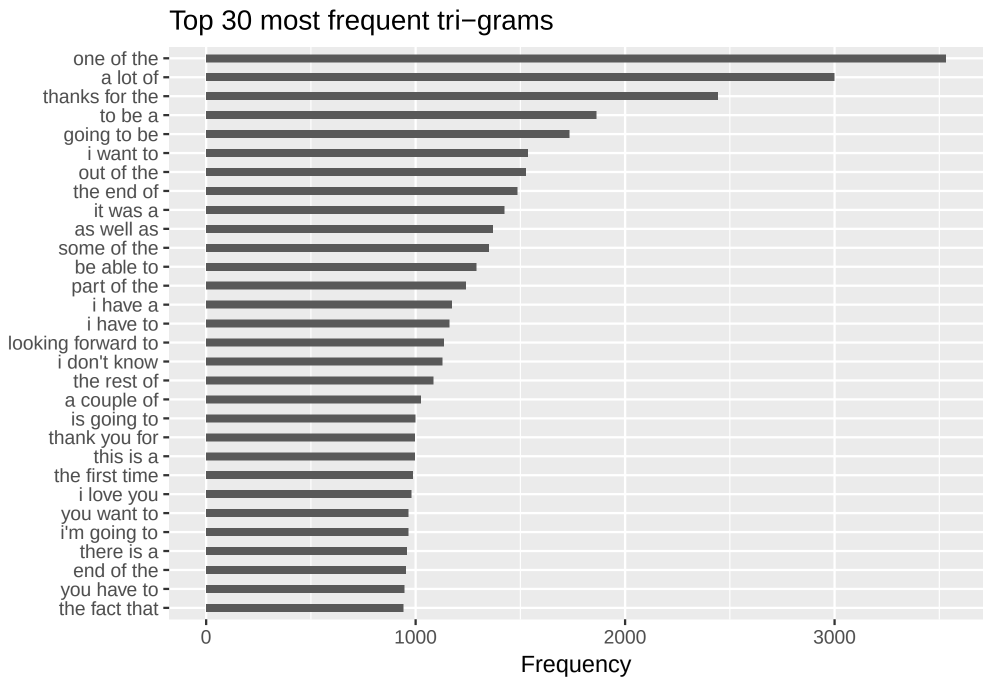


Table 6: Frequency Groups - Trigrams

Category	Min Frequency	Max Frequency	Average Frequency	Words	Instances
1	1	704	1.47	160859	9152835
2	714	1368	937.79	26	40325
3	1425	1864	1594.83	6	9569
4	2443	2443	2443.00	1	2443
5	2999	3532	3265.50	2	6531

Now that we have an impression of uni-, bi-, and tri-grams, let us see how many unique terms (uni-, bi-, tri-grams) we need in a frequency sorted way to cover 50%, 90%, and 99% of all term instances in the sampled corpus. This is shown in Table. 7.

Table 7: Coverage Table

	Terms	50% Coverage	90% Coverage	99% Coverage
Uni-Grams	177327	143	7414	81119
Bi-Grams	2621538	38710	1657953	2525180
Tri-Grams	6242375	1636501	5321201	6150258

Appendix

In this section, we can find the scripts and some mathematical details used in the main section to create the contents.

```
#----- Required Libraries ----#

require(quanteda)
require(readr)
```

```r
require(stopwords)
require(tidytext)
require(dplyr)
require(ngram)
require(knitr)
require(kableExtra)
require(stringr)
require(ggplot2)
require(data.table)
```

```r
#---- Some Auxiliary Functions -----#

'%!in%' <- function(x,y)!('%in%'(x,y))

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

Preprocessing & Uni-Grams

```r
#--- Loading the Data (Blogs, Twitter, News) ---#

blogs <- read_lines("rawData/en_US/en_US.blogs.txt")
twitter <- readLines("rawData/en_US/en_US.twitter.txt", skipNul = TRUE)
news <- read_lines("rawData/en_US/en_US.news.txt")
```

```r
# Getting the Size of the Corresponding Files
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)
size_corpus <- size_blogs + size_twitter + size_news

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

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

```r
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 = "")


max_length_corpus <- max(max_length_news, max_length_blogs, max_length_twitter)
min_length_corpus <- min(min_length_news, min_length_blogs, min_length_twitter)
min_max_chars_corpus <- paste(as.character(min_length_corpus), " - ",
                            as.character(max_length_corpus), sep = "")

avg_nchar_news <- round(mean(length_news), 2)
avg_nchar_twitter <- round(mean(length_twitter), 2)
avg_nchar_blogs <- round(mean(length_blogs), 2)
avg_nchar_corpus <- avg_nchar_news + avg_nchar_blogs + avg_nchar_twitter

info_blogs <- c("Blogs", size_blogs, lines_blogs,
                paste("~", as.character(words_blogs), ",000,000", sep = ""),
                min_max_chars_blogs,
                avg_nchar_blogs)

info_twitter <- c("Twitter", size_twitter, lines_twitter,
                  paste("~", as.character(words_twitter), ",000,000", sep = ""),
                  min_max_chars_twitter,
                  avg_nchar_twitter)

info_news <- c("News", size_news, lines_news,
               paste("~", as.character(words_news), ",000,000", sep = ""),
               min_max_chars_news,
               avg_nchar_news)

info_corpus <- c("Corpus", size_corpus, lines_corpus,
               paste("~", as.character(words_corpus), ",000,000", sep = ""),
               min_max_chars_corpus,
               avg_nchar_corpus)

raw_info <- as.data.frame(rbind(info_blogs, info_twitter, info_news, info_corpus))
colnames(raw_info) = c("Data",
                       "Size (mg)",
                       "Lines",
                       "Words",
                       "Range nchars",
                       "Avg nchars")
rownames(raw_info) = NULL
```

```
#---- A Table of Some Info about the Raw Data -----#

kable(raw_info, "latex", booktabs = T,
      caption = "Raw Data - Information") %>%
        kable_styling(latex_options = "hold_position") %>%
        footnote(general =  c("`Words`: approximate in million",
                              "`Range nchars`: range and avg number of chars in lines",
                              "`Avg nchars`: range and avgerage number of chars in lines"
                              ))
```

```
#--- Sampling the Data ---#

set.seed(2019)
blogs_ind <- sample(length(blogs), length(blogs) * 0.1)
twitter_ind <- sample(length(twitter), length(twitter) * 0.1)
news_ind <- sample(length(news), length(news) * 0.1)

blogs_sample <- blogs[blogs_ind]
twitter_sample <- twitter[twitter_ind]
news_sample <- news[news_ind]

corp_sample <- corpus(c(blogs_sample,
                          twitter_sample,
                          news_sample))
```

```
#--- Raw Sampled data info ----#


lines_blogs_sample <- length(blogs_sample)
lines_twitter_sample <- length(twitter_sample)
lines_news_sample <- length(news_sample)
lines_corpus_sample <- lines_blogs_sample + lines_twitter_sample + lines_news_sample

words_blogs_sample <- round(wordcount(blogs_sample)/1000000, 0)
words_twitter_sample <- round(wordcount(twitter_sample)/1000000, 0)
words_news_sample <- round(wordcount(news_sample)/1000000, 0)
words_corpus_sample <- words_blogs_sample + words_twitter_sample + words_news_sample

length_blogs_sample <- sapply(blogs_sample, nchar)
range_chars_blogs_sample <- paste(as.character(min(length_blogs_sample)), " - ",
                             as.character(max(length_blogs_sample)), sep = "")

length_twitter_sample <- sapply(twitter_sample, nchar)
range_chars_twitter_sample <- paste(as.character(min(length_twitter_sample)), " - ",
                               as.character(max(length_twitter_sample)), sep = "")

length_news_sample <- sapply(news_sample, nchar)
range_chars_news_sample <- paste(as.character(min(length_news_sample)), " - ",
                            as.character(max(length_news_sample)), sep = "")
```

```r
min_sample_corp <- min(min(length_blogs_sample),
                       min(length_twitter_sample),
                       min(length_news_sample))

max_sample_corp <- max(max(length_blogs_sample),
                       max(length_twitter_sample),
                       max(length_news_sample))

range_chars_corpus_sample <- paste(as.character(min_sample_corp), " - ",
                            as.character(max_sample_corp), sep = "")


avg_nchar_news_sample <- round(mean(length_news_sample), 2)
avg_nchar_twitter_sample <- round(mean(length_twitter_sample), 2)
avg_nchar_blogs_sample <- round(mean(length_blogs_sample), 2)
avg_nchar_corpus_sample <- avg_nchar_news_sample +
        avg_nchar_twitter_sample +
        avg_nchar_blogs_sample

info_blogs_sample <- c("Blogs", lines_blogs_sample,
                paste("~", as.character(words_blogs_sample), ",000,000", sep = ""),
                range_chars_blogs_sample,
                avg_nchar_blogs_sample)

info_twitter_sample <- c("Twitter", lines_twitter_sample,
                  paste("~", as.character(words_twitter_sample), ",000,000", sep = ""),
                  range_chars_twitter_sample,
                  avg_nchar_twitter_sample)

info_news_sample <- c("News", lines_news_sample,
               paste("~", as.character(words_news_sample), ",000,000", sep = ""),
               range_chars_news_sample,
               avg_nchar_news_sample)

info_corpus_sample <- c("Corpus", lines_corpus_sample,
               paste("~", as.character(words_corpus_sample), ",000,000", sep = ""),
               range_chars_corpus_sample,
               avg_nchar_corpus_sample)

raw_info_sample <- as.data.frame(rbind(info_blogs_sample,
                                       info_twitter_sample,
                                       info_news_sample,
                                       info_corpus_sample))
colnames(raw_info_sample) = c("Data",
                       "Lines",
                       "Words",
                       "Range nchars",
                       "Avg nchars")
rownames(raw_info_sample) = NULL

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

```r
#---- UniGrams ----#

unigrams <- tokens(corp_sample,
                       remove_numbers = TRUE,
                       remove_hyphens = TRUE,
                       remove_url = TRUE,
                       remove_symbols = TRUE,
                       remove_separators = TRUE,
                       remove_punct = TRUE,
                       remove_twitter = TRUE)
# to lower case
unigrams <- tokens_tolower(unigrams)
```

```r
#---- DFM of Unigrams ---#

# DFM
dfm_uni <- dfm(unigrams)
# Tidy the DFM of Unigrams
dt_uni <- tidy(dfm_uni)
```

```r
#----  Profanity Words/Expressions ----#

profanities <- read_lines("rawData/bad_words.txt")
# Extract the profanities from data
dfm_profanities <- dfm_select(dfm_uni, pattern = profanities,
                         selection = "keep")
# Tidy up
dt_profanities <- tidy(dfm_profanities)
```

```r
#----- Extracting Non-English Words -----#

noneng_ind <- grepl("[^\x01-\x7F]+", dt_uni$term)
# A data table with non-english words
dt_uni_neng <- dt_uni[which(noneng_ind), ]
```

```r
#----- Information Table for Unigrams ----#

# nummber of word instances in the corpus
num_words <- sum(dt_uni$count)
# number of words in the corpus
num_uniq_words <- length(unique(dt_uni$term))

# nummber of profanity instances in the corpus
num_profanities <- sum(dt_profanities$count)
# number of unique profanities in the corpus
num_uniq_profanities <- length(unique(dt_profanities$term))
# Portion - profanities in the corpus
profanities_perc <- round(num_profanities / num_words * 100, 2)

# number of non english instanes
num_neng_words <- sum(dt_uni_neng$count)
# number of unique non-english words in the corpus
num_uniq_neng_words <- length(dt_uni_neng$term)
# Portion - non english words in the corpus
neng_perc <- round((num_neng_words / num_words) * 100, 2)
```

```r
info_uni <- data.frame(num_words,
                       num_uniq_words,
                       num_profanities,
                       #num_uniq_profanities,
                       paste(as.character(profanities_perc), "%"),
                       num_neng_words,
                       #num_uniq_neng_words,
                       paste(as.character(neng_perc), "%"))
colnames(info_uni) <- c("Words",
                        "Unique Words",
                        "Profanities",
                        #"Unique Profanities",
                        "Profanity",
                        "Non-English",
                        #"Unique Non-English",
                        "Non-English")

kable(info_uni, "latex", booktabs = T,
      caption = "Unigrams (Words, Profanities, Non-English Words) in the Corpus") %>%
        kable_styling(latex_options = "hold_position")
```

```r
#--- Profanity Filtering on tokens ----#

unigrams_clean <- tokens_remove(unigrams, pattern = profanities)
# Non-english words Filtering
unigrams_clean <- tokens_select(unigrams_clean,
                                pattern = "[A-z]*[^\x01-\x7F]+[A-z]*",
                                selection = "remove", valuetype = "regex")
```

```r
#---- DFM of clean unigrams ----#

dfm_uni_clean <- dfm(unigrams_clean)
# The tidy data of clean unigrams
dt_uni_clean <- tidy(dfm_uni_clean)
```

```r
#---  Tidy Frequency - unigrams - summarize ---#

dt_uni_freq <- dt_uni_clean %>%
        group_by(term) %>%
        summarize(freq = sum(count)) %>%
        arrange(desc(freq))
```

```r
#---- Frequency Plot - unigrams ----#

ggplot(dt_uni_freq[1:30, ],
       aes(x = reorder(term, freq),
           y = freq))  +
        geom_col(width = 0.5) +
        xlab(NULL) +
        ylab("Frequency") +
        coord_flip() +
        ggtitle("Top 30 most frequent words in the clean corpus")
```

```r
#----- Frequency Groups - Unigrams ----#

dt_uni_freq_grp <- dt_uni_freq
dt_uni_freq_grp$bin <- cut(dt_uni_freq_grp$freq,
                                breaks = 5, labels = 1:5)
dt_uni_freq_grp <- dt_uni_freq_grp %>%
        group_by(bin) %>%
        summarize(min_freq = round(min(freq), 2),
                  max_freq = round(max(freq), 2),
                  avg_freq = round(mean(freq),2),
                  words = length(unique(term)),
                  frequency = sum(freq))

colnames(dt_uni_freq_grp) <- c("Category",
                               "Min Frequency",
                               "Max Frequency",
                               "Average Frequency",
                               "Words",
                               "Instances")

kable(dt_uni_freq_grp, "latex", booktabs = T,
      caption = "Frequency Groups - Unigrams") %>%
        kable_styling(latex_options = "hold_position")
```

```r
#---- DFM for unigrams excl. stop-words ---#

dfm_uni_clean_wostp <- dfm_remove(dfm_uni_clean, stopwords("english"))
# Tidy up
dt_uni_clean_wostp <- tidy(dfm_uni_clean_wostp)
# Frequency
dt_uni_freq_wostp <- dt_uni_clean_wostp %>%
        group_by(term) %>%
        summarize(freq = sum(count)) %>%
        arrange(desc(freq))
```

```r
# --- Frequency Plot - unigrams ----#

ggplot(dt_uni_freq_wostp[1:30, ],
       aes(x = reorder(term, freq),
            y = freq))  +
        geom_col(width = 0.4) +
        xlab(NULL) +
        ylab("Frequency") +
        coord_flip() +
        ggtitle("Top 30 most frequent words excluding stopwords")
```

```r
#--- Worldcloud - unigrams ---#

set.seed(1000)
textplot_wordcloud(dfm_uni_clean_wostp, random_order = FALSE,
                   rotation = .25,
                   color = RColorBrewer::brewer.pal(8,"Dark2")
                  )
```

Bi-Grams & Tri-Grams

```
#--- Bi-Grams extraction from clean unigrams ---#

bigrams <- tokens_ngrams(unigrams_clean, n = 2)
```

```
#---- DFM of Bigrams ----#

dfm_bi <- dfm(bigrams)
# Tidy the DFM of Bigrams
dt_bi <- tidy(dfm_bi)
```

```
#---- group, summarize, and order ----#

dt_bi_freq <- dt_bi %>%
        group_by(term) %>%
        summarize(freq = sum(count)) %>%
        arrange(desc(freq))
```

```
#----- Seperating Words -----#

dt_bi_freq_sep <- data.frame(
        word1 = sapply(strsplit(dt_bi_freq$term, "_", fixed = TRUE), '[[', 1),
        word2 = sapply(strsplit(dt_bi_freq$term, "_", fixed = TRUE), '[[', 2),
        freq = dt_bi_freq$freq)
```

```
dt_bi_freq_sep <- dt_bi_freq_sep[!duplicated(dt_bi_freq_sep %>%
                                                     select(word1, word2)), ]
```

```
#----- Frequency Plot - bigrams ----#

ggplot(dt_bi_freq_sep[1:30, ],
       aes(x = reorder(paste(word1, word2, sep = " "), freq),
            y = freq))  +
        geom_col(width = 0.4) +
        xlab(NULL) +
        ylab("Frequency") +
        coord_flip() +
        ggtitle("Top 30 most frequent bigrams")
```

```
#----- Frequency Groups - Bigrams ----#

dt_bi_freq_grp <- dt_bi_freq_sep
dt_bi_freq_grp$bin <- cut(dt_bi_freq_grp$freq,
                                breaks = 5, labels = 1:5)
dt_bi_freq_grp <- dt_bi_freq_grp %>%
        group_by(bin) %>%
        summarize(min_freq = round(min(freq), 2),
                  max_freq = round(max(freq), 2),
                  avg_freq = round(mean(freq),2),
                  words = length(unique(word1)),
                  frequency = sum(freq))

colnames(dt_bi_freq_grp) <- c("Category",
                               "Min Frequency",
                                "Max Frequency",
                                "Average Frequency",
                                "Words",
                                "Instances")

kable(dt_bi_freq_grp, "latex", booktabs = T,
      caption = "Frequency Groups - Bigrams") %>%
        kable_styling(latex_options = "hold_position")
```

```
# --- Tri-Grams extraction from clean unigrams ----#

trigrams <- tokens_ngrams(unigrams_clean, n = 3)
```

```
#----- DFM of Trigrams ----#

dfm_tri <- dfm(trigrams)
# Tidy the DFM of Trigrams
dt_tri <- tidy(dfm_tri)
```

```
#---- group, summarize, and order ----#

dt_tri_freq <- dt_tri %>%
        group_by(term) %>%
        summarize(freq = sum(count)) %>%
        arrange(desc(freq))
```

```
#----- Separating Words - TriGrams ----#

dt_tri_freq_sep <- data.frame(
        word1 = sapply(strsplit(dt_tri_freq$term, "_", fixed = TRUE), '[[', 1),
        word2 = sapply(strsplit(dt_tri_freq$term, "_", fixed = TRUE), '[[', 2),
        word3 = sapply(strsplit(dt_tri_freq$term, "_", fixed = TRUE), '[[', 3),
        freq = dt_tri_freq$freq)
```

```
#----- Frequency Plot - trigrams ------#

ggplot(dt_tri_freq_sep[1:30, ],
       aes(x = reorder(paste(word1, word2, word3, sep = " "), freq),
           y = freq))  +
        geom_col(width = 0.4) +
        xlab(NULL) +
        ylab("Frequency") +
        coord_flip() +
        ggtitle("Top 30 most frequent tri-grams")
```

```
#----- Freq Table : Tri-Grams -----#

dt_tri_freq_grp <- dt_tri_freq_sep
dt_tri_freq_grp$bin <- cut(dt_tri_freq_grp$freq,
                                breaks = 5, labels = 1:5)
dt_tri_freq_grp <- dt_tri_freq_grp %>%
        group_by(bin) %>%
        summarize(min_freq = round(min(freq), 2),
                  max_freq = round(max(freq), 2),
                  avg_freq = round(mean(freq),2),
                   words = length(unique(word1)),
                   frequency = sum(freq))

colnames(dt_tri_freq_grp) <- c("Category",
                               "Min Frequency",
                               "Max Frequency",
                               "Average Frequency",
                               "Words",
                               "Instances")

kable(dt_tri_freq_grp, "latex", booktabs = T,
      caption = "Frequency Groups - Trigrams") %>%
        kable_styling(latex_options = "hold_position")
```

```
#----- Coverage Table -----#

dt_coverage <- data.frame(
        Terms = c(length(dt_uni_freq$term),
                  length(dt_bi_freq$term),
                  length(dt_tri_freq$term)),
        Cov_50 = c(findMany(dt_uni_freq, .5),
                   findMany(dt_bi_freq, .5),
                   findMany(dt_tri_freq, .5)),
        Cov_90 = c(findMany(dt_uni_freq, .9),
                   findMany(dt_bi_freq, .9),
                   findMany(dt_tri_freq, .9)),
        Cov_99 = c(findMany(dt_uni_freq, .99),
                   findMany(dt_bi_freq, .99),
                   findMany(dt_tri_freq, .99))
)
# Rename Column Names
colnames(dt_coverage) = c("Terms",
                          "50% Coverage", "90% Coverage", "99% Coverage")
rownames(dt_coverage) = c("Uni-Grams", "Bi-Grams", "Tri-Grams")

# Represent the Table
kable(dt_coverage, "latex", booktabs = T,
      caption = "Coverage Table") %>%
        kable_styling(latex_options = "hold_position")
```