# Natural Language Processing Project

## Initialization

The inital step that loads the required libraries and downloads the data sets ifff not all read on file.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.0.3
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## Warning: package 'tibble' was built under R version 4.0.3
```

```
## Warning: package 'readr' was built under R version 4.0.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 4.0.3
```

```
#downloads the corpus files, profanity filter and English dictionary

url <- "https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip"
url2 <- "https://www.freewebheaders.com/download/files/facebook-bad-words-list_comma-separated-text-file
url3 <- "https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt"
if(dir.exists("~/R/Capestone/data/") == FALSE){
       dir.create("~/R/Capestone/data/")}

if(file.exists("~/R/Capestone/data/data.zip") == FALSE|
   file.exists("~/R/Capestone/data/prof.zip")==FALSE|
   file.exists("~/R/Capestone/data/diction.txt")==FALSE){
        download.file(url,destfile = "~/R/Capestone/data/data.zip")
        download.file(url2,destfile = "~/R/Capestone/data/prof.zip")
        download.file(url3,destfile = "~/R/Capestone/data/diction.txt")
        setwd("~/R/Capestone/data/")
        unzip("~/R/Capestone/data/prof.zip")
        unzip("~/R/Capestone/data/data.zip")
        setwd("~/R/Capestone")
}
```

## Create corpus

At this stage the files are open and joined to create a corpus for the project. The Corpus is so large and requires some much ram that a sample of 20% is taken.

```
blog <- read_lines("~/R/Capestone/data/final/en_US/en_US.blogs.txt")
news <- read_lines("~/R/Capestone/data/final/en_US/en_US.news.txt")
twitter <- read_lines("~/R/Capestone/data/final/en_US/en_US.twitter.txt")
blog <- tibble(text = blog)
news <- tibble(text = news)
twitter <- tibble(text = twitter)

set.seed(90210)
corpus <- full_join(blog,twitter) %>% full_join(news) %>%
        slice_sample(prop = 0.20) %>%
        mutate(line = row_number(), .before = "text")
```

```
## Joining, by = "text"
## Joining, by = "text"
```

## Corpus filtering

Here the corpus filter is created to remove profanity and any word that is not in the English dictionary.

```
prof <- read_lines("~/R/Capestone/data/facebook-bad-words-list_comma-separated-text-file_2021_01_18.txt
prof <- prof %>% str_split(", ") %>% flatten %>% unlist
prof <- tibble("word" = prof)

english <- read_lines("~/R/Capestone/data/diction.txt")
english <- tibble("word" = english[!english==""])
```

## Vocabulary

A vocabulary of words is created from the unique words with the applied filters

```
#clean up ram
rm(blog,news,twitter)
voc <- english %>% anti_join(prof)
```

```
## Joining, by = "word"
```

```
unigram <- corpus %>% unnest_tokens(ngram, text, token = "ngrams", n = 1) %>%
        semi_join(voc, by = c("ngram"="word")) %>% count(ngram)
#decreases the voc size
voc <- tibble(word = unigram$ngram)
```

## Out of Vocabulary

To model out of vocabulary words we take a sampling of the least frequent unigrams and change them to the character “”. If a word is tested that isn’t in the vocabulary for the corpus, the quantity will be converted to “”.

```
#OOV 1% of the least likely unigrams
unks <- unigram[unigram$n==1,] %>% slice_sample(prop = 0.01)
unigram[unigram$ngram %in% unks$ngram,]$ngram <- "<unk>"
unigram <- count(unigram, ngram)
```