Text Processing

"lyrics.csv" is a filtered corpus of 380,000+ song lyrics from from MetroLyrics. You can read more about it on Kaggle.

"artists.csv" provides the background information of all the artistis. These information are scraped from LyricsFreak.

In this R notebook, we process the raw textual data for our data analysis.

Step 0 - Load all the required libraries

From the packages' descriptions:

- `tm` is a framework for text mining applications within R;
- `data.table` is a package for fast aggregation of large data;
- `tidyverse` is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures;
- `tidytext` allows text mining using 'dplyr', 'ggplot2', and other tidy tools;
- `DT` provides an R interface to the JavaScript library DataTables.

```r
library(tm)
library(data.table)
library(tidytext)
library(tidyverse)
library(DT)
```

Step 1 - Load the data to be cleaned and processed

```r
# load lyrics data
load('../data/lyrics.RData')
```

Step 2 - Preliminary cleaning of text

We clean the text by converting all the letters to the lower case, and removing punctuation, numbers, empty words and extra white space.

```r
# function for removimg leading and trailing whitespace from character strings
leadingWhitespace <- content_transformer(function(x) str_trim(x, side = "both"))
# remove stop words
data("stop_words")
word <- c("lot", "today", "months", "month", "wanna", "wouldnt", "wasnt", "ha", "na", "ooh", "da",
        "gonna", "im", "dont", "aint", "wont", "yeah", "la", "oi", "nigga", "fuck",
          "hey", "year", "years", "last", "past", "feel")
stop_words <- c(stop_words$word, word)
# clean the data and make a corpus
corpus <- VCorpus(VectorSource(dt_lyrics$lyrics))%>%
  tm_map(content_transformer(tolower))%>%
  tm_map(removePunctuation)%>%
  tm_map(removeWords, character(0))%>%
  tm_map(removeWords, stop_words)%>%
  tm_map(removeNumbers)%>%
  tm_map(stripWhitespace)%>%
  tm_map(leadingWhitespace)
```

Step 3 - Stemming words and converting tm object to tidy object

Stemming reduces a word to its word *stem*. We stem the words here and then convert the "tm" object to a "tidy" object for much faster processing.

```
stemmed <- tm_map(corpus, stemDocument) %>%
  tidy() %>%
  select(text)
```

Step 4 - Creating tidy format of the dictionary to be used for completing stems

We also need a dictionary to look up the words corresponding to the stems.

```
dict <- tidy(corpus) %>%
  select(text) %>%
  unnest_tokens(dictionary, text)
```

Step 5 - Combining stems and dictionary into the same tibble

Here we combine the stems and the dictionary into the same "tidy" object.

```
completed <- stemmed %>%
  mutate(id = row_number()) %>%
  unnest_tokens(stems, text) %>%
  bind_cols(dict)
```

Step 6 - Stem completion

Lastly, we complete the stems by picking the corresponding word with the highest frequency.

```
completed <- completed %>%
  group_by(stems) %>%
  count(dictionary) %>%
  mutate(word = dictionary[which.max(n)]) %>%
  ungroup() %>%
  select(stems, word) %>%
  distinct() %>%
  right_join(completed) %>%
  select(-stems)
```

Step 8 - Pasting stem completed individual words into their respective lyrics

We want our processed words to resemble the structure of the original lyrics. So we paste the words together to form processed lyrics.

```
completed <- completed %>%
  group_by(id) %>%
  summarise(stemmedwords= str_c(word, collapse = " ")) %>%
  ungroup()
```

Step 9 - Keeping a track of the processed lyrics with their own ID

```
dt_lyrics <- dt_lyrics %>%
  mutate(id = row_number()) %>%
  inner_join(completed)
```

Exporting the processed text data into a CSV file

```
save(dt_lyrics, file="../output/processed_lyrics.RData")
```

The final processed data is ready to be used for any kind of analysis.