NLP - Assignment 2

In this assignment you will...

- learn how to do tokenization using the tidytext package.

Load text into R

The first task of this assignment consists of reloading your book and extracting the main text from the book.

1) Load your book using `read_file()`.
2) Rerun all of the steps up to and including the step, in which you extract the main text from the document (task 4 in the *Tokenize* section of the previous assignment).

Tokenize using `tidytext`

1) Create a `tibble` from you text using the code below.

```
# create tibble
text_tbl <- tibble(text = main_text)
```

2) Use the pipe operator `%>%` to compute the number of characters in the string using the code below.

```
# compute the number of characters using the pipe
text_tbl %>% nchar()
```

The above example illustrates a different way of passing on an (the first) argument to a function. While this may not yet seem very practical now, you will soon see how this style of coding makes it easy to create efficient analysis *pipe*lines.

3) Use `unnest_tokens()` function of the `tidytext` package (don't foget `library(tidytext)`) to tokenize the text. The function takes three main inputs the data (`tbl`), a name for variable that should contain the tokens (`output`, e.g., word), and the variable that contains the text to be tokenized (`input`). Using the pipe, specify the latter two arguments and tokenize your text.

```
# tokenize the text
text_tbl %>%
  unnest_tokens(output = XX,
                input = YY)
```

4) `unnest_tokens()` makes tokenization into words really easy. It even allows tokenization into sentences using the `token` argument (see `?unnest_tokens`. Tokenize into sentences rather words using the template below.

```
# tokenize the text
text_tbl %>%
  unnest_tokens(output = XX,
                input = YY,
                token = "ZZ")
```

5) Take a look at the result. Has `unnest_tokens()` done its job?

6) Insert a new step into the analysis pipeline that creates a new variable containing indices for the different sentences. See below. Now the usefulness of pipes should become clear.

```
# tokenize the text
text_tbl %>%
  unnest_tokens(output = XX,
                input = YY,
                token = "ZZ") %>%
  mutate(sent_ind = 1:n())
```

7) Now use `unnest_token()` another time to tokenize the sentences into words. The results of this should be a `tibble()` containing two variables, one coding the sentence from which the words came from and another coding the actual words. Store the `tibble` in an object called `token_tbl`.

This is it, for now. Next, session we will pick up from here to compare word vectors and conduct semantic analyses.