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()`.

```
# load text
text <- read_file('grimm.txt')
```

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).

```
# define regex
regex <- '\\*{3}[:print:]*\\*{3}'

# cut text into sections
text_split = str_split(text, '\\*{3}[:print:]*\\*{3}')

# get sections
sections <- text_split[[1]]

# select main text
main_text <- sections[2]
```

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()
```

```
##   text
## 530064
```

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)
```

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

```
## # A tibble: 101,660 x 1
##    word
##    <chr>
##  1 produced
##  2 by
##  3 emma
##  4 dudding
##  5 john
##  6 bickers
##  7 and
##  8 dagny
##  9 fairy
## 10 tales
## # ... with 101,650 more rows
```

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")
```

```
# tokenize the text
text_tbl %>%
  unnest_tokens(output = word,
                input = text,
                token = "sentences")
```

```
## # A tibble: 4,537 x 1
##    word
##    <chr>
##  1 produced by emma dudding, john bickers, and dagny            fairy tale~
##  2 contents:          the golden bird       hans in luck       jorinda and ~
##  3 how they went to the mountains to eat nuts         2.
```

```
##  4 how chanticleer and partlet went to visit mr korbes       rapunzel     ~
##  5 the turnip       clever hans       the three languages       the fox an~
##  6 these apples were always counted, and about  the time when they began t~
##  7 the king became very angry at this, and ordered the  gardener to keep w~
##  8 the gardener set his  eldest son to watch; but about twelve o'clock he ~
##  9 then the second son was  ordered to watch; and at midnight he too fell ~
## 10 then the third son offered to keep watch; but  the gardener at first wo~
## # ... with 4,527 more rows
```

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())
```

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

```
## # A tibble: 4,537 x 2
##    word                                                        sent_ind
##    <chr>                                                          <int>
##  1 produced by emma dudding, john bickers, and dagny            f~        1
##  2 contents:          the golden bird       hans in luck       jor~        2
##  3 how they went to the mountains to eat nuts         2.                  3
##  4 how chanticleer and partlet went to visit mr korbes       rapu~        4
##  5 the turnip       clever hans       the three languages       t~        5
##  6 these apples were always counted, and about  the time when the~        6
##  7 the king became very angry at this, and ordered the  gardener ~        7
##  8 the gardener set his  eldest son to watch; but about twelve o'~        8
##  9 then the second son was  ordered to watch; and at midnight he ~        9
## 10 then the third son offered to keep watch; but  the gardener at~       10
## # ... with 4,527 more rows
```

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`.

```
# define regex
token_tbl = text_tbl %>%
  unnest_tokens(sentence, text, token = "sentences") %>%
  mutate(sentence_ind = as.character(1:n())) %>%
  unnest_tokens(word, "sentence")
```

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