Eksploracja tekstu i wyszukiwanie informacji w mediach społecznościowych

LABORATORIUM 2

- Schludny tekst
- Inwokacja!
- Zliczanie słów
- Słowa funkcyjne
- Pakiet gutenbergr

Schludny tekst (tidy text)

Tidy text oraz **tidy data**, czyli w wolnym tłumaczeniu **schludny lub czysty tekst** i **czyste dane** to pewnego rodzaju paradygmat programistyczny związany z pakietem **R**, wprowadzany i propagowany przez **Hadleya Wickhama** (Chief Scientist w RStudio). Założenia struktur, na których pracuje się w tidy text sprowadzają się do następujących punktów:

- każda zmienna jest kolumną,
- każda obserwacja jest rzędem,
- każda jednostka obserwacyjna jest tabelą.

W efekcie schludny tekst jest zdefiniowany jako **tabela z pojedynczym tokenem na rząd**, przy czym słowo token będziemy najczęsciej rozumieć jako wyraz lub słowo.

Warto też wiedzieć, że podstawowa forma danych związana z tidy text to tzw. **tibble**. Jest to pewna wariacja na temat ramki danych, posiadająca wygodne listowanie, pomijająca nazwy rzędów. Dodatkowo w tibble nie następuje konwersja z typu char do typu factor (co jest prawdziwą zmorą dla początkujących, jeśli chodzi o ramki danych).

Inwokacja!

Aby prześledzić jak wygląda przekształcanie zwykłego tekstu na tidy text posłużymy się pierwszymi czterema wersami Pana Tadeusza:

```
# PRZYKŁAD 2.1
mr.ted <- c("  Litwo! Ojczyzno moja! ty jesteś jak zdrowie;",
            "Ile cię trzeba cenić, ten tylko się dowie",
            "Kto cię stracił. Dziś piękność twą w całéj ozdobie",
            "Widzę i opisuję, bo tęsknię po tobie.")

mr.ted
```

```
## [1] "  Litwo! Ojczyzno moja! ty jesteś jak zdrowie;"
## [2] "Ile cię trzeba cenić, ten tylko się dowie"
## [3] "Kto cię stracił. Dziś piękność twą w całéj ozdobie"
## [4] "Widzę i opisuję, bo tęsknię po tobie."
```

Aby otrzymać tibble wykorzystujemy eponimiczną funkcję z biblioteki **dplyr** (w zasadzie z biblioteki **tibble**), dodatkowo podając numery linii:

```
# PRZYKŁAD 2.2

library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df <- tibble(lines = 1:length(mr.ted), text = mr.ted)
df
```

```
## # A tibble: 4 x 2
##   lines text
##   <int> <chr>
## 1     1 "  Litwo! Ojczyzno moja! ty jesteś jak zdrowie;"
## 2     2 Ile cię trzeba cenić, ten tylko się dowie
## 3     3 Kto cię stracił. Dziś piękność twą w całéj ozdobie
## 4     4 Widzę i opisuję, bo tęsknię po tobie.
```

Sęk w tym, taka postać do końca spełnia "wymogi" czystego tekstu - powinniśmy przekształcić tę strukturę w taką tabelę, w której w każdym rzędzie znajdzie się słowo. Przydatna do tego będzie funkcja **unnest_tokens()** z pakietu **tidytext**. Funkcja przyjmuje jako argumenty strukture z danymi, nazwę wynikowej kolumny, w której mają być przetworzone dane oraz nazwę kolumny, z której ma korzystać przy przekształcaniu danych. Domyślnie ustawione są opcje **to_lower=TRUE** oraz **drop=TRUE**, czyli w wyrazach zmieniane są wielkie litey na małe oraz usuwa się z wejściowe dane.

```
# PRZYKŁAD 2.3

library(tidytext)
df.words <- df %>%
  unnest_tokens(word, text)
df.words
```

```
## # A tibble: 31 x 2
##    lines word
##    <int> <chr>
##  1     1 litwo
##  2     1 ojczyzno
##  3     1 moja
##  4     1 ty
##  5     1 jesteś
##  6     1 jak
##  7     1 zdrowie
##  8     2 ile
##  9     2 cię
## 10     2 trzeba
## # ... with 21 more rows
```

Zliczanie słów (tidy text)

Do dalszej obróbki danych będziemy potrzebować jeszcze kilku istotnych funkcji. Pierwszą z nich jest **count()**, która zlicza wystąpienia poszczególnych słów - jeśli wyposażymy ją w opcję **sort=TRUE** otrzymamy tibble zawierająca słowa od najczęstszego do najrzadszego:

```
# PRZYKŁAD 2.3

df.words %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 30 x 2
##    word       n
##    <chr>  <int>
##  1 cię        2
##  2 bo         1
##  3 całéj      1
##  4 cenić      1
##  5 dowie      1
##  6 dziś       1
##  7 i          1
##  8 ile        1
##  9 jak        1
## 10 jesteś     1
## # ... with 20 more rows
```

Bardzo często w naszej strukturze musimy dodać jakieś zmienne (kolumny) lub po prostu je zmienić. Używamy wtedy funkcji **mutate()** z dplyr (ta sama klasa funkcji, co **arrange**, **filter** etc) np

```
# PRZYKŁAD 2.4

df <- tibble(x = 1:5, y = letters[1:5])
df
```

```
## # A tibble: 5 x 2
##       x y
##   <int> <chr>
## 1     1 a
## 2     2 b
## 3     3 c
## 4     4 d
## 5     5 e
```

```
df %>%
  mutate(y = letters[6:10], z = runif(5))
```

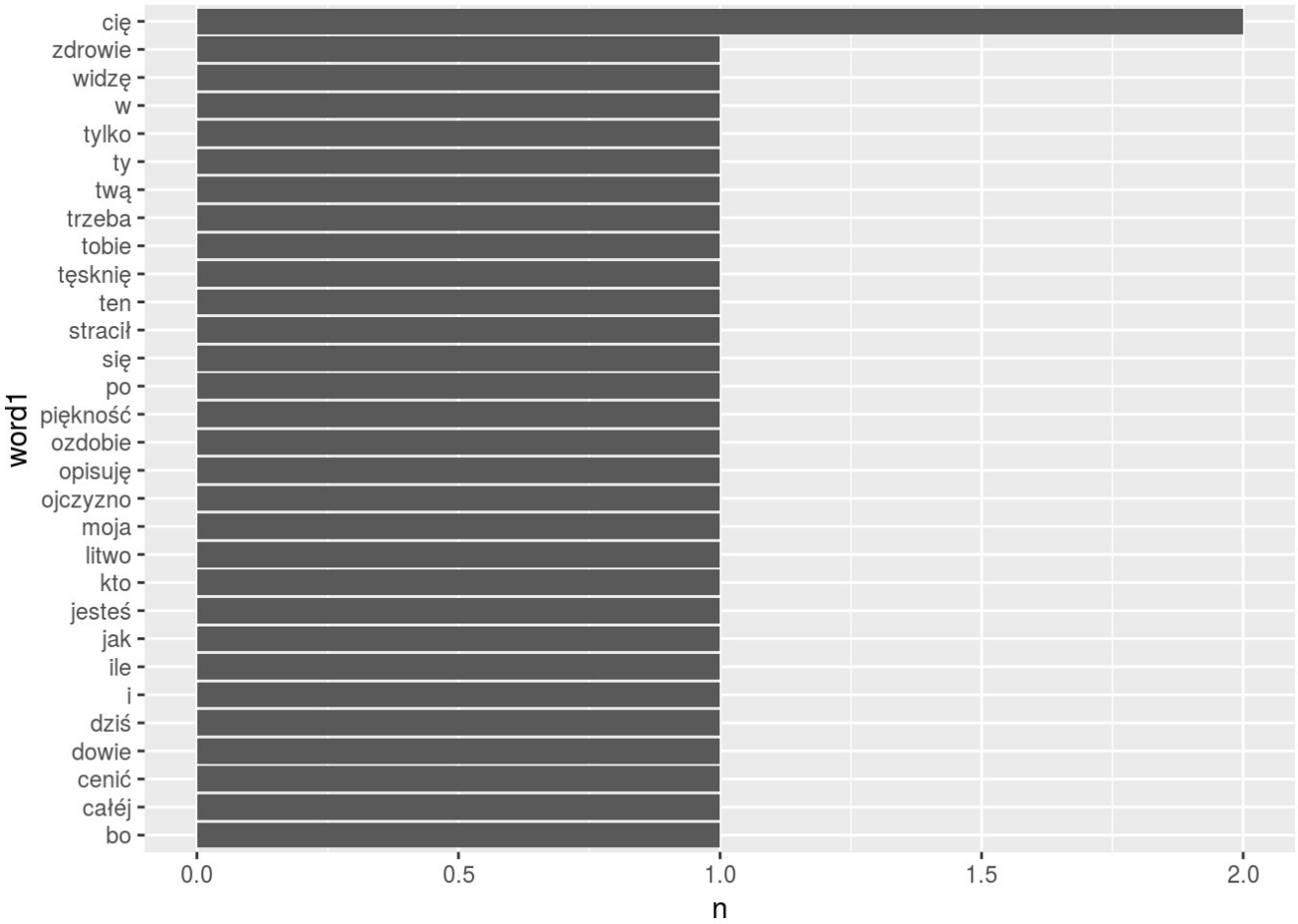
```
## # A tibble: 5 x 3
##       x y          z
##   <int> <chr>  <dbl>
## 1     1 f     0.253
## 2     2 g     0.494
## 3     3 h     0.487
## 4     4 i     0.406
## 5     5 j     0.0608
```

Stworzymy teraz wykres kolumnowy (a w zasadzie wierszowy) częstości słów z rozpatrywanego kawałka tekstu. Posłużymy się pakietem **ggplot**, żeby jednak można było wykonać wykres musimy przekształcić zmienne char na typ wyliczeniowy (factor) z określoną kolejnością za pomocą funkcji **reorder()**.

```
# PRZYKŁAD 2.3

library(ggplot2)

df.words %>%
  count(word, sort = TRUE) %>%
  mutate(word1 = reorder(word, n)) %>%
  ggplot() +
  geom_col(aes(word1, n)) +
  coord_flip()
```



Słowa funkcyjne

Większość słów na otrzymanym wykresie mało wnosi do jakiejkolwiek analizy tekstu - są to tak zwane "stopwords" (słowa przestankowe, funkcyjne). Warto się ich pozbyć z tekstu. Aby jednak to zrobić, wprowadzimy jeszcze jedną istotną funkcję **anti_join()**. Należy ona do zestawu funkcji działających podobnie do zapytań SQL a zawierających między innymi **inner_join()**, **left_join()**, **right_join()** oraz **full_join()**. Skorzystamy z dwóch wbudowanych zbiorów danych

```
band_members
```

```
## # A tibble: 3 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
## 2 John  Beatles
## 3 Paul  Beatles
```

```
band_instruments
```

```
## # A tibble: 3 x 2
##   name  plays
##   <chr> <chr>
## 1 John  guitar
## 2 Paul  bass
## 3 Keith guitar
```

i sprawdzimy jakie są efekty

```
band_members %>% inner_join(band_instruments)
```

```
## Joining, by = "name"
```

```
## # A tibble: 2 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
```

```
band_members %>% left_join(band_instruments)
```

```
## Joining, by = "name"
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
```

```
band_members %>% right_join(band_instruments)
```

```
## Joining, by = "name"
```

```
## # A tibble: 3 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 John  Beatles guitar
## 2 Paul  Beatles bass
## 3 Keith <NA>    guitar
```

```
band_members %>% full_join(band_instruments)
```

```
## Joining, by = "name"
```

```
## # A tibble: 4 x 3
##   name  band    plays
##   <chr> <chr>   <chr>
## 1 Mick  Stones  <NA>
## 2 John  Beatles guitar
## 3 Paul  Beatles bass
## 4 Keith <NA>    guitar
```

```
band_members %>% anti_join(band_instruments)
```

```
## Joining, by = "name"
```

```
## # A tibble: 1 x 2
##   name  band
##   <chr> <chr>
## 1 Mick  Stones
```

Teraz skorzystamy już z pliku **stopw.dat** (pobrany z stąd (https://github.com/bieli/stopwords/blob/master/polish.stopwords.txt))

```
stops <- read.table("http://www.if.pw.edu.pl/~julas/TEXT/lab/stopw.dat", stringsAsFactors = F)
stops <- tibble(word = stops$V1)
stops
```

```
## # A tibble: 350 x 1
##    word
##    <chr>
##  1 a
##  2 aby
##  3 ach
##  4 acz
##  5 aczkolwiek
##  6 aj
##  7 albo
##  8 ale
##  9 alez
## 10 ależ
## # ... with 340 more rows
```

i za jego pomocą pozbędziemy się słów funkcyjnych

```
df.words %>%
  count(word, sort = TRUE) %>%
  anti_join(stops)
```

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

```
## # A tibble: 14 x 2
##    word         n
##    <chr>    <int>
##  1 całéj        1
##  2 cenić        1
##  3 dowie        1
##  4 jesteś       1
##  5 litwo        1
##  6 ojczyzno     1
##  7 opisuję      1
##  8 ozdobie      1
##  9 piękność     1
## 10 stracił      1
## 11 tęsknię      1
## 12 twą          1
## 13 widzę        1
## 14 zdrowie      1
```

Pakiet Gutenbergr

Przytoczona anliza jest oczywiście tylko przykładem wykonanym na bardzo krótkim tekście. Aby dostać się do większych zbiorów, można skorzystać z biblioteki **gutenbergr** i zawartych w niej funkcji **gutenberg_works()** oraz **gutenberg_download()**

```
library(gutenbergr)

gutenberg_works()
```

```
## # A tibble: 40,737 x 8
##    gutenberg_id title author gutenberg_autho.. language gutenberg_books..
##           <int> <chr> <chr>             <int> <chr>    <chr>
##  1            0 <NA>  <NA>                NA en       <NA>
##  2            1 The .. Jeffe..          1638 en         United States L..
##  3            2 "The.. Unite..             1 en         American Revolu..
##  4            3 John.. Kenne..          1666 en         <NA>
##  5            4 "Lin.. Linco..             3 en         US Civil War
##  6            5 The .. Unite..             1 en         American Revolu..
##  7            6 Give.. Henry..             4 en         American Revolu..
##  8            7 The .. <NA>               NA en       <NA>
##  9            8 Abra.. Linco..             3 en         US Civil War
## 10            9 Abra.. Linco..             3 en         US Civil War
## # ... with 40,727 more rows, and 2 more variables: rights <chr>,
## #   has_text <lgl>
```

```
gutenberg_works(languages = "pl") %>%
  filter(author == "Mickiewicz, Adam")
```

```
## # A tibble: 6 x 8
##   gutenberg_id title author gutenberg_autho.. language gutenberg_books..
##          <int> <chr> <chr>             <int> <chr>    <chr>
## 1        27081 Sone.. Micki..            32429 pl       <NA>
## 2        27723 Moja.. Micki..            32429 pl       <NA>
## 3        27729 Bajki Micki..             32429 pl       <NA>
## 4        28049 Bala.. Micki..            32429 pl       <NA>
## 5        28153 Graż.. Micki..            32429 pl       <NA>
## 6        31536 "Pan.. Micki..            32429 pl       <NA>
## # ... with 2 more variables: rights <chr>, has_text <lgl>
```

```
gutenberg_download(31536)
```

```
## Determining mirror for Project Gutenberg from http://www.gutenberg.org/robot/harves
t
```

```
## Using mirror http://aleph.gutenberg.org
```

```
## # A tibble: 5,370 x 2
##    gutenberg_id text
##           <int> <chr>
##  1        31536 PAN TADEUSZ.
##  2        31536 ""
##  3        31536 ""
##  4        31536 TOM PIERWSZY.
##  5        31536 ""
##  6        31536 ""
##  7        31536 ""
##  8        31536 ""
##  9        31536 PAN TADEUSZ.
## 10        31536 ""
## # ... with 5,360 more rows
```