Eksploracja tekstu i wyszukiwanie informacji w mediach społecznościowych

LABORATORIUM 5

- Bigramy
- Zliczanie i filtrowanie bi- i n-gramów
- Analiza bigramów
- Grafy

Bigramy

W trakcie kilku poprzednich zajęć korzystaliśmy często z funkcji **unnest_tokens()**, aby rozbić zwarty tekst na poszczególne tokeny -- w tym przypadku pojedyncze słowa. Jest to bezpośrednie zastosowanie znanego modelu **bag-of-words**, czyli przypadku, gdy traktujemy pojedyncze słowa jako wrzucone do torby i przemieszane ze sobą.

Tak jak poprzednio, wykorzystamy zbiór trzech książek Juliusza Verne'a:

```
# PRZYKŁAD 5.1
library(gutenbergr)
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
```

```
library(tidytext)
library(magrittr)
library(ggplot2)

# g <- gutenberg_works()
#
# v <- gutenberg_download(c(83, 103, 1268))
#
# books <- g[g$gutenberg_id %in% c(83, 103, 1268),c("gutenberg_id","title")]
#
# v %<>% left_join(books) %>%
#   mutate(gutenberg_id = NULL)

load("v.data")
```

Wykonujemy podobne operacje, jak na poprzednich zajęciach, tzn. zliczamy poszczególne słowa w każdej z trzech książek i dodatkowo dokładamy całkowita liczbę słów

```
# PRZYKŁAD 5.2

verne_bigrams <- v %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

verne_bigrams
```

```
## # A tibble: 350,152 x 2
##    title                          bigram
##    <chr>                          <chr>
##  1 Around the World in Eighty Days around the
##  2 Around the World in Eighty Days the world
##  3 Around the World in Eighty Days world in
##  4 Around the World in Eighty Days in eighty
##  5 Around the World in Eighty Days eighty days
##  6 Around the World in Eighty Days days contents
##  7 Around the World in Eighty Days contents chapter
##  8 Around the World in Eighty Days chapter i
##  9 Around the World in Eighty Days i in
## 10 Around the World in Eighty Days in which
## # ... with 350,142 more rows
```

Zliczanie i filtrowanie bi- i n-gramów

Podobnie jak w przypadku unigramów możemy korzystać z funkcji pakietu **dplyr** do zliczania poszczególnych tokenów.

```
# PRZYKŁAD 5.3
verne_bigrams %>%
  count(bigram, sort = TRUE)
```

```
## # A tibble: 140,364 x 2
##    bigram       n
##    <chr>    <int>
##  1 of the    4492
##  2 in the    1732
##  3 to the    1610
##  4 on the    1304
##  5 it was    1145
##  6 and the    987
##  7 at the     798
##  8 by the     756
##  9 from the   708
## 10 that the   628
## # ... with 140,354 more rows
```

Oczywiście, nie jest żadnym zaskoczeniem to, co przed chwila zaobserwowaliśmy -- najczęściej występują typowe zbitki słów składające się na funkcyjne wyrażenia w języku angielskim. Aby się pozbyć tych wyrażeń, moglibyśmy wyszukać poszczególne słowa za pomocą wyrażeń regularnych, korzystając ze zbioru **stop_words**. Łatwiej jednak dokonać rozbicia posczególnych wyrażeń na dwa słowa za pomocą funkcji **separate()** z pakietu **tidyr**. Funkcja ta rozkłada daną kolumnę korzystając z podanego wzorca (w naszym przypadku to po prostu spacja). Następnie w każdej kolumnie usuwamy słowa kluczowe i je zliczamy.

```
# PRZYKŁAD 5.4
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:magrittr':
##
##     extract
```

```
bigrams_sep <- verne_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_flt <- bigrams_sep %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

bigram_counts <- bigrams_flt %>%
  count(word1, word2, sort = TRUE)

bigram_counts
```

```
## # A tibble: 27,348 x 3
##    word1    word2         n
##    <chr>    <chr>     <int>
##  1 cyrus    harding     534
##  2 granite  house       329
##  3 gideon   spilett     292
##  4 phileas  fogg        241
##  5 michel   ardan       200
##  6 lincoln  island      174
##  7 gun      club        109
##  8 replied  barbicane    86
##  9 replied  pencroft     78
## 10 prospect heights      76
## # ... with 27,338 more rows
```

Jak widac typowe znaczące połaczenia to nazwy własne (imię + nazwisko) lub lokazlizacje. Gdybyśmy z jakichś powodów chcieli dokonać powtórnego "zlepienia" poszczególnych słów możemy wykorzystać do tego funkcję **unite()** z **tidyr**.

```
# PRZYKŁAD 5.5

bigrams_uni <- bigrams_flt %>%
  unite(bigram, word1, word2, sep = " ")

bigrams_uni
```

```
## # A tibble: 37,030 x 2
##    title                          bigram
##    <chr>                          <chr>
##  1 Around the World in Eighty Days eighty days
##  2 Around the World in Eighty Days days contents
##  3 Around the World in Eighty Days contents chapter
##  4 Around the World in Eighty Days phileas fogg
##  5 Around the World in Eighty Days passepartout accept
##  6 Around the World in Eighty Days ideal iii
##  7 Around the World in Eighty Days conversation takes
##  8 Around the World in Eighty Days cost phileas
##  9 Around the World in Eighty Days phileas fogg
## 10 Around the World in Eighty Days fogg dear
## # ... with 37,020 more rows
```

Rzecz jasna, stosując podobną procedurę rozkładania na poszczególne słowa i usuwania słów kluczowych, możemy również otrzymać n-gramy z $n > 2$, np. trigramy:

```
# PRZYKŁAD 5.6

v %>%
  unnest_tokens(trigram, text, token = "ngrams", n = 3) %>%
  separate(trigram, c("word1", "word2", "word3"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word,
         !word3 %in% stop_words$word) %>%
  count(word1, word2, word3, sort = TRUE)
```

```
## # A tibble: 9,197 x 4
##    word1     word2    word3        n
##    <chr>     <chr>    <chr>    <int>
##  1 replied   cyrus    harding     42
##  2 exclaimed michel   ardan       26
##  3 replied   gideon   spilett     22
##  4 answered  cyrus    harding     19
##  5 replied   michel   ardan       17
##  6 twenty    thousand pounds      17
##  7 harding   gideon   spilett     13
##  8 gideon    spilett  herbert     12
##  9 observed  gideon   spilett     12
## 10 sir       francis  cromarty    12
## # ... with 9,187 more rows
```

Analiza n-gramów

Korzystając z uzyskanego przez nas formatu bigramów łatwo jest wykonać proste zadania eksploracyjne, np. we wszystkich książkach rzeczownik "wyspa" i sprawdzić jakie określenia są do niego przypisane:

```
# PRZYKŁAD 5.7

bigrams_flt %>%
  filter(word2 == "island") %>%
  count(title, word1, sort = TRUE)
```

```
## # A tibble: 23 x 3
##    title                                          word1           n
##    <chr>                                          <chr>       <int>
##  1 The Mysterious Island                          lincoln       174
##  2 The Mysterious Island                          tabor          75
##  3 The Mysterious Island                          desert          5
##  4 The Mysterious Island                          norfolk         4
##  5 Around the World in Eighty Days                rock            2
##  6 The Mysterious Island                          unknown         2
##  7 Around the World in Eighty Days                fire            1
##  8 Around the World in Eighty Days                noble           1
##  9 Around the World in Eighty Days                uninhabited     1
## 10 From the Earth to the Moon; and, Round the Moon melville        1
## # ... with 13 more rows
```

Bigram możemy traktować podobnie jak unigramy, tzn np. policzyć dla niego transformację TF-IDF

```
# PRZYKŁAD 5.8

bigram_tf_idf <- bigrams_uni %>%
  count(title, bigram) %>%
  bind_tf_idf(bigram, title, n) %>%
  arrange(desc(tf_idf))

bigram_tf_idf
```

```
## # A tibble: 28,064 x 6
##    title                          bigram           n      tf   idf  tf_idf
##    <chr>                          <chr>        <int>   <dbl> <dbl>   <dbl>
##  1 Around the World in Eighty Da. phileas fogg   241 0.0319   1.10 0.0350
##  2 The Mysterious Island          cyrus hardi.   534 0.0283   1.10 0.0311
##  3 From the Earth to the Moon; a. michel ardan   200 0.0189   1.10 0.0207
##  4 The Mysterious Island          granite hou.   329 0.0174   1.10 0.0192
##  5 The Mysterious Island          gideon spil.   292 0.0155   1.10 0.0170
##  6 From the Earth to the Moon; a. gun club       109 0.0103   1.10 0.0113
##  7 The Mysterious Island          lincoln isl.   174 0.00923  1.10 0.0101
##  8 Around the World in Eighty Da. hong kong       63 0.00833  1.10 0.00915
##  9 From the Earth to the Moon; a. replied bar.    86 0.00811  1.10 0.00890
## 10 Around the World in Eighty Da. sir francis     53 0.00701  1.10 0.00770
## # ... with 28,054 more rows
```

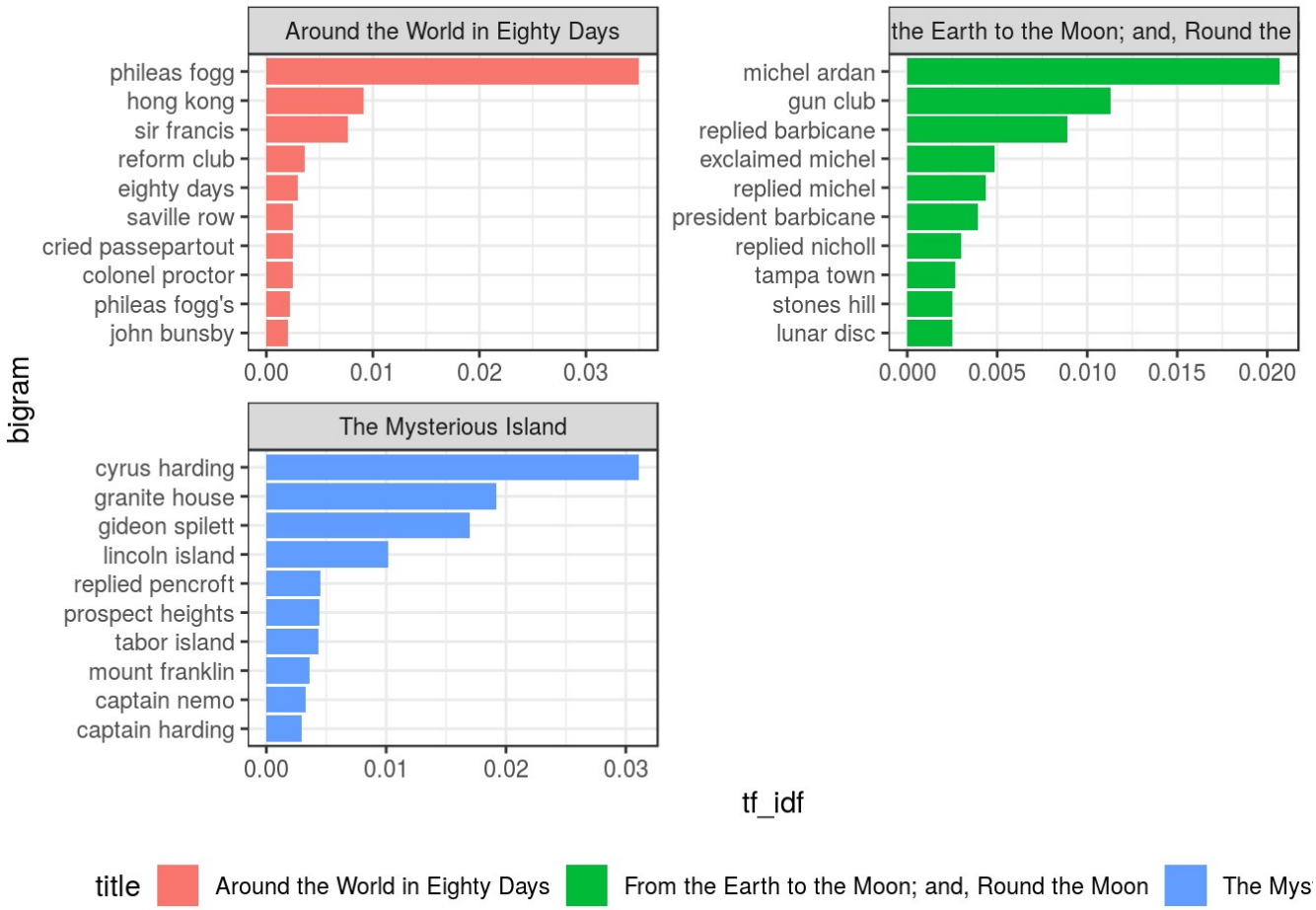
Podobnie, korzystając z pakietu **ggplot2** mamy szanse przedstawić porównanie pomiędzy poszczególnymi rozpatrywanymi przez nas książkami.

```
# PRZYKŁAD 5.9
library(ggplot2)

theme_set(theme_bw())

bigram_tf_idf %>%
  group_by(title) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(bigram = reorder(bigram, tf_idf)) %>%
  ggplot() +
  geom_col(aes(bigram, tf_idf, fill = title)) +
  coord_flip() +
  facet_wrap(~title, nrow = 2, scales = "free") +
  theme(legend.position = "bottom")
```

```
## Selecting by tf_idf
```



Grafy

Czasem bardzo wygodną metodą wizualizacji połączen słów są sieci. W ogólnym ujęciu sieć złożona to układ w którym wyróżniamy węzły, będące reprezentacją jakichś bytów (człowiek, strona WWW, związek chemiczny) oraz połączeń, które oddają relacje pomiędzy nimi (znajomość, hiperlink, reakcja).

```
# PRZYKŁAD 5.10

library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:tidyr':
##
##     %>%, crossing
```

```
## The following object is masked from 'package:magrittr':
##
##     %>%
```

```
## The following objects are masked from 'package:dplyr':
##
##     %>%, as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
bigram_graph <- bigram_counts %>%
  filter(n > 20) %>%
  graph_from_data_frame()

  bigram_graph
```

```
## IGRAPH DN-- 74 55 --
## + attr: name (v/c), n (e/n)
## + edges (vertex names):
##  [1] cyrus    ->harding   granite  ->house     gideon   ->spilett
##  [4] phileas  ->fogg      michel   ->ardan     lincoln  ->island
##  [7] gun      ->club      replied  ->barbicane replied  ->pencroft
## [10] prospect ->heights   tabor    ->island    hong     ->kong
## [13] mount    ->franklin  captain  ->nemo      hundred  ->feet
## [16] sir      ->francis   captain  ->harding   thousand ->pounds
## [19] half     ->past      cried    ->pencroft  exclaimed->michel
## [22] replied  ->herbert   replied  ->cyrus     san      ->francisco
## + ... omitted several edges
```

Pakiet **igraph** przeciąża funkcję **plot()** tak, że możlwie jest wykreślenie relacji w postaci grafu:

```
# PRZYKŁAD 5.10
  plot.graph <- function(g) {

    plot(g, vertex.size = 0, edge.arrow.size = 0.75)
  }

  plot.graph(bigram_graph)
```