Eksploracja tekstu i wyszukiwanie informacji w mediach społecznościowych

LABORATORIUM 8

- Metoda LSA
- Metoda LDA

Latent Semantic Analysis

Latent Semantic Analysis (**LSA**), to dość bezposrednia metoda, która nie wymaga specjalnego przygotowania, jesli chodzi o zapis matematyczny. Korzystamy tu z rozkładu macierzy na wartości osobliwe (*singular value decomposition*, **SVD**), który mówi, że macierz $\mathbf{A}$ możemy zapisać jako iloczyn macierzy

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

Z formalnego punktu widzenia $\mathbf{U}$ zawiera wektory własne $\mathbf{A}\mathbf{A}^T$, natomiast $\mathbf{V}$ - macierzy $\mathbf{A}^T\mathbf{A}$. Z tego wynika, że macierz $\mathbf{\Sigma}$ jest macierza diagonalna, zwana **macierzą osobliwą**, zawierająca wartości własne $\mathbf{A}\mathbf{A}^T$ (lub $\mathbf{A}^T\mathbf{A}$ - są takie same).

W naszym przypadku macierz $\mathbf{A}$ jest macierza termów-dokumentów (w rzędach termy, w kolumnach dokumenty). Problemem jest fakt, iż taka macierz może być olbrzymia i olbrzymi może również być efekt wykonania SVD. Z pomocą przychodzi nam twierdzenie, które mówi, że jeżeli wybierze się k największych wartości z macierzy $\mathbf{\Sigma}$, to "okrojenie" macierzy $\mathbf{U}$ oraz $\mathbf{V}$ do k kolumn da najmniejszy błąd przybliżenia oryginalnego SVD.

$$\mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^T$$

Aby rozpocząć musimy załadować zwykłe biblioteki oraz dedykowaną **lsa**

```
# PRZYKŁAD 8.1

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(tm)
```

```
## Loading required package: NLP
```

```
library(lsa)
```

```
## Loading required package: SnowballC
```

oraz stworzyć 5 bardzo prostych dokumentów

```
d1 <- "Romeo and Juliet."
d2 <- "Juliet: O happy dagger!"
d3 <- "Romeo died by dagger."
d4 <- "'Live free or die', that's the New-Hampshire's motto."
d5 <- "Did you know, New-Hampshire is in New-England."
```

Następnie wykonujemy nasz zwykły zestaw operacji: tworzymy z dokumentow źródło, a póxniej korpus i czyścimy teksty, finalnie ortrzymując macierz termów-dokumentów.

```
# PRZYKŁAD 8.2

removeSpecialChars <- function(x) gsub("[^0-9A-Za-z///' ]"," ",x)

data <- tibble(doc_id = 1:5, text = c(d1, d2, d3, d4, d5))

corpus <- VCorpus(DataframeSource(as.data.frame(data)))

corpus <- corpus %>% tm_map(removeWords, stopwords("en")) %>%
  tm_map(removePunctuation) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(removeSpecialChars)) %>%
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument)

doc <- TermDocumentMatrix(corpus)

as.matrix(doc)
```

```
##              Docs
## Terms         1 2 3 4 5
##   dagger      0 1 1 0 0
##   did         0 0 0 0 1
##   die         0 0 1 1 0
##   free        0 0 0 1 0
##   happi       0 1 0 0 0
##   juliet      1 1 0 0 0
##   know        0 0 0 0 1
##   live        0 0 0 1 0
##   motto       0 0 0 1 0
##   newengland  0 0 0 0 1
##   newhampshir 0 0 0 1 1
##   romeo       1 0 1 0 0
```

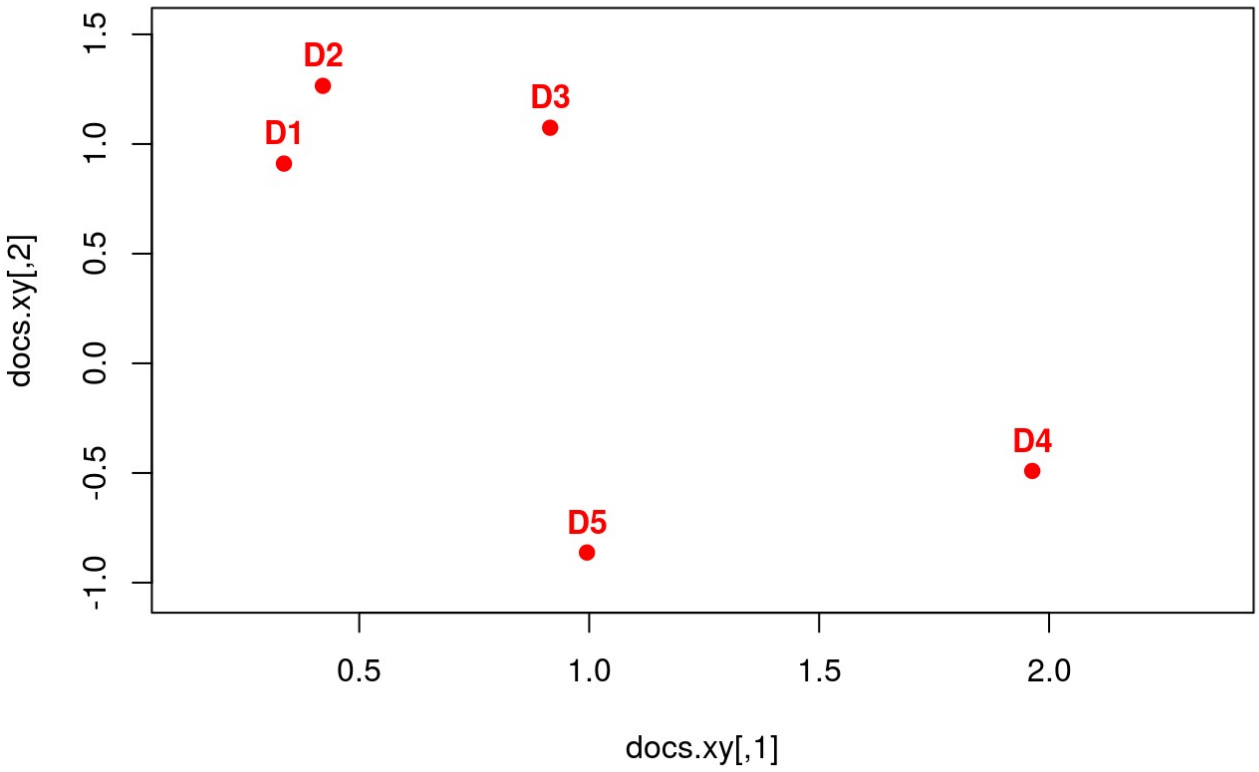
W tym momencie mamy już naszą docelowa macierz $\mathbf{A}$, którą przekażemy do funkcji **lsa()** z pakietu **lsa**. Wypiszmy, to co zwraca ta funkcja: są to pola **tk**, **sk** i **dk**, które odpowiadają macierzom $\mathbf{U}_k$, $\mathbf{\Sigma}_k$ i $\mathbf{V}_k^T$

```
# PRZYKŁAD 8.3

doc.lsa <- lsa(doc, 2)

doc.lsa
```

```
## $tk
##                   [,1]       [,2]
## dagger      0.22368403  0.5120325
## did         0.16661029 -0.1888053
## die         0.48192449  0.1277159
## free        0.32871351 -0.1073965
## happi       0.07047305  0.2769201
## juliet      0.12677483  0.4762459
## know        0.16661029 -0.1888053
## live        0.32871351 -0.1073965
## motto       0.32871351 -0.1073965
## newengland  0.16661029 -0.1888053
## newhampshir 0.49532380 -0.2962019
## romeo       0.20951275  0.4344382
##
## $dk
##        [,1]       [,2]
## 1 0.1375994  0.4260550
## 2 0.1722334  0.5919112
## 3 0.3744418  0.5025481
## 4 0.8033632 -0.2295580
## 5 0.4071892 -0.4035676
##
## $sk
## [1] 2.443962 2.137480
##
## attr(,"class")
## [1] "LSAspace"
```

Główną zaletą takiej transformacji jest to, że otrzymujemy przedstawienie dokumentów oraz wyrazów (termów) w pewnej niskowymiarowej przestrzeni. Formalnie musimy pomnożyć każdy wiersz $\mathbf{U}_k$, i $\mathbf{V}_k$ przez $\mathbf{\Sigma}_k$, następnie można je swobodnie wykreslić (domyślnie mamy dwuwymiarową przestrzeń)

```
# PRZYKŁAD 8.4

docs.xy <- t(diag(doc.lsa$sk) %*% t(doc.lsa$dk))
terms.xy <- doc.lsa$tk %*% diag(doc.lsa$sk)

max.xy <- apply(rbind(apply(docs.xy, 2, max), apply(terms.xy, 2, max)), 2, max)
min.xy <- apply(rbind(apply(docs.xy, 2, min), apply(terms.xy, 2, min)), 2, min)

max.xy <- max.xy + 0.2 * abs(max.xy)
min.xy <- min.xy - 0.2 * abs(min.xy)

plot(docs.xy, pch = 19, col = "red", xlim = c(min.xy[1], max.xy[1]), ylim = c(min.xy[2
], max.xy[2]))
text(docs.xy, col = "red", label = c("D1", "D2", "D3", "D4", "D5"), font = 2, pos = 3)
```



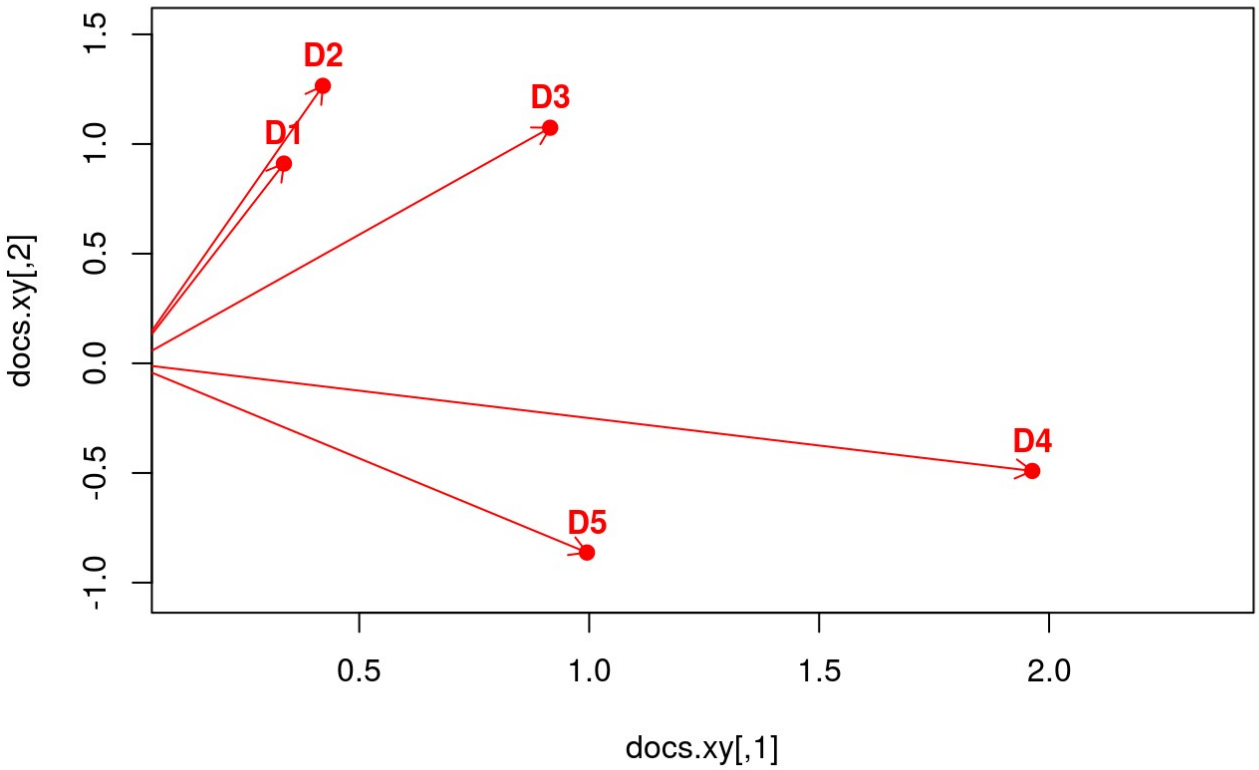
Jak widać, dokumenty d_1 i d_2 leżą bardzo blisko siebie. Ten wniosek można potwierdzić badając macierz cosinusa podobieństwa - korzystamy tu z funkcji **cosine()** z **lsa**

```
# PRZYKŁAD 8.5

cosine(t(docs.xy))
```

```
##            1           2           3          4           5
## 1  1.0000000  0.99947009  0.93872955 0.10862577 -0.35268605
## 2  0.9994701  1.00000000  0.92701344 0.07621034 -0.38295800
## 3  0.9387296  0.92701344  1.00000000 0.44458545 -0.00856909
## 4  0.1086258  0.07621034  0.44458545 1.00000000  0.89189386
## 5 -0.3526861 -0.38295800 -0.00856909 0.89189386  1.00000000
```

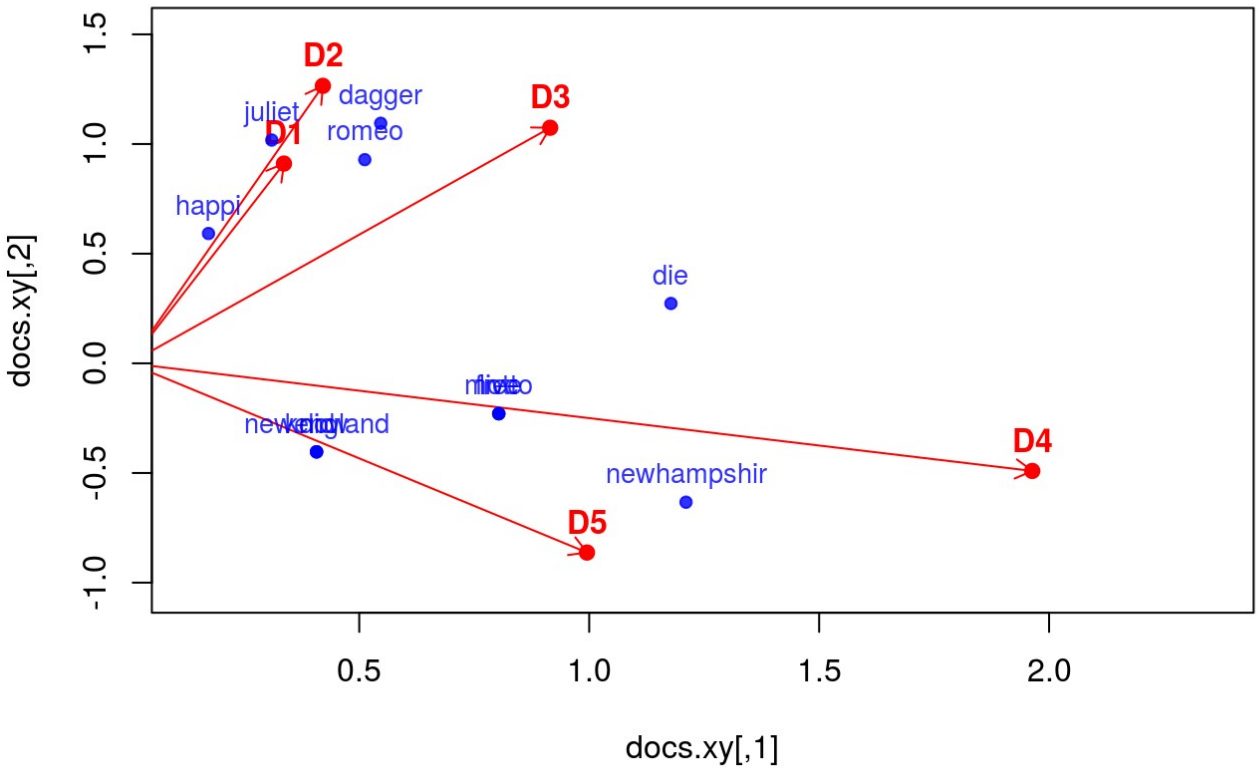
dodatkowo możemy na wykres nacieść strzałki odpowiadające dokumetom jako wektorom

```
# PRZYKŁAD 8.6

arrows(0, 0, docs.xy[,1], docs.xy[,2], length = 0.1, col = "red")
```



W taki sam sposób umieścimy na wykresie również poszczególne słowa:

```
# PRZYKŁAD 8.6

points(terms.xy, pch = 19, cex = 0.8, col = rgb(0, 0, 1, 0.8))
text(terms.xy, label=rownames(terms.xy), cex = 0.85, col = rgb(0, 0, 1, 0.8), pos = 3)
```

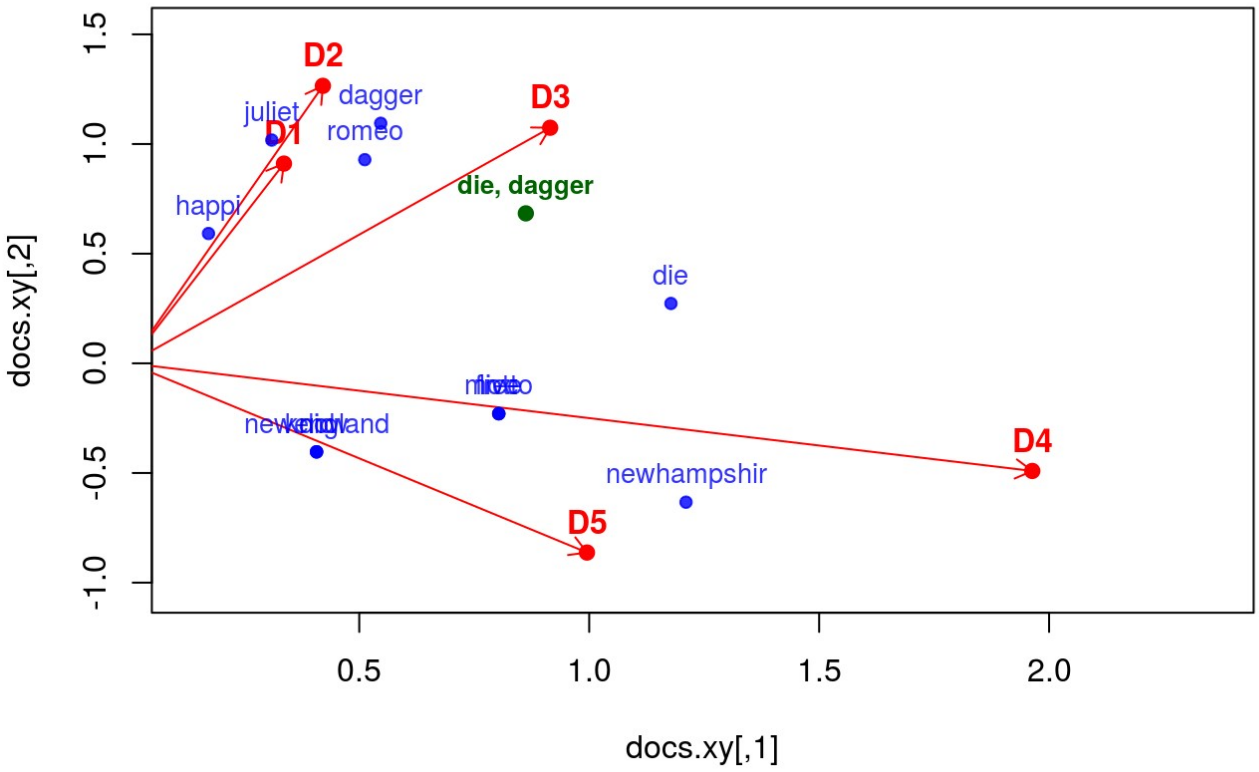


LSA jest wykorzystywane do wyszukiwania dokumentow spełniających określone zapytanie (*search query*). Oczywiście, w zapytaniu muszą znajdować się słowa, na bazie których stworzono reprezentację LSA - w przeciwnym wypadku takie wyrazy zostaną po prostu pominięte. Formalnie, aby oddać dokument $\mathbf{q}$ w nowej przestrzeni, powinniśmy policzyć $\mathbf{q}_k = \mathbf{\Sigma}_k^{-1}\mathbf{U}_k^T\mathbf{q}$. Nam wystarczy wyszukać słowa w macierzy $terms.xy$ i policzyć średnią po współrzędnych.

```
# PRZYKŁAD 8.7

query1 <- c("die", "dagger")
query1.xy <- apply(terms.xy[rownames(terms.xy) %in% query1,], 2, mean)
```

Taki punkt umieszczamy na naszym wykresie:

```
# PRZYKŁAD 8.8

points(t(apply(terms.xy[rownames(terms.xy) %in% query1,], 2, mean)), col = "darkgreen"
, pch = 19)
text(t(apply(terms.xy[rownames(terms.xy) %in% query1,], 2, mean)), col = "darkgreen",
label = "die, dagger", cex = 0.8, font = 2, pos = 3)
```



Nasz wyszukiwany łańcuch jest najbliżej dokumentu d_3, czego raczej należało sie spodziewać. Jeszcz dla czystego powtierdzenia rzut oka na macierz cosinusów z dodanym dokumentem.

```
# PRZYKŁAD 8.9

rbind(docs.xy, query1.xy) %>%
  t() %>%
  cosine() %>%
  round(3)
```

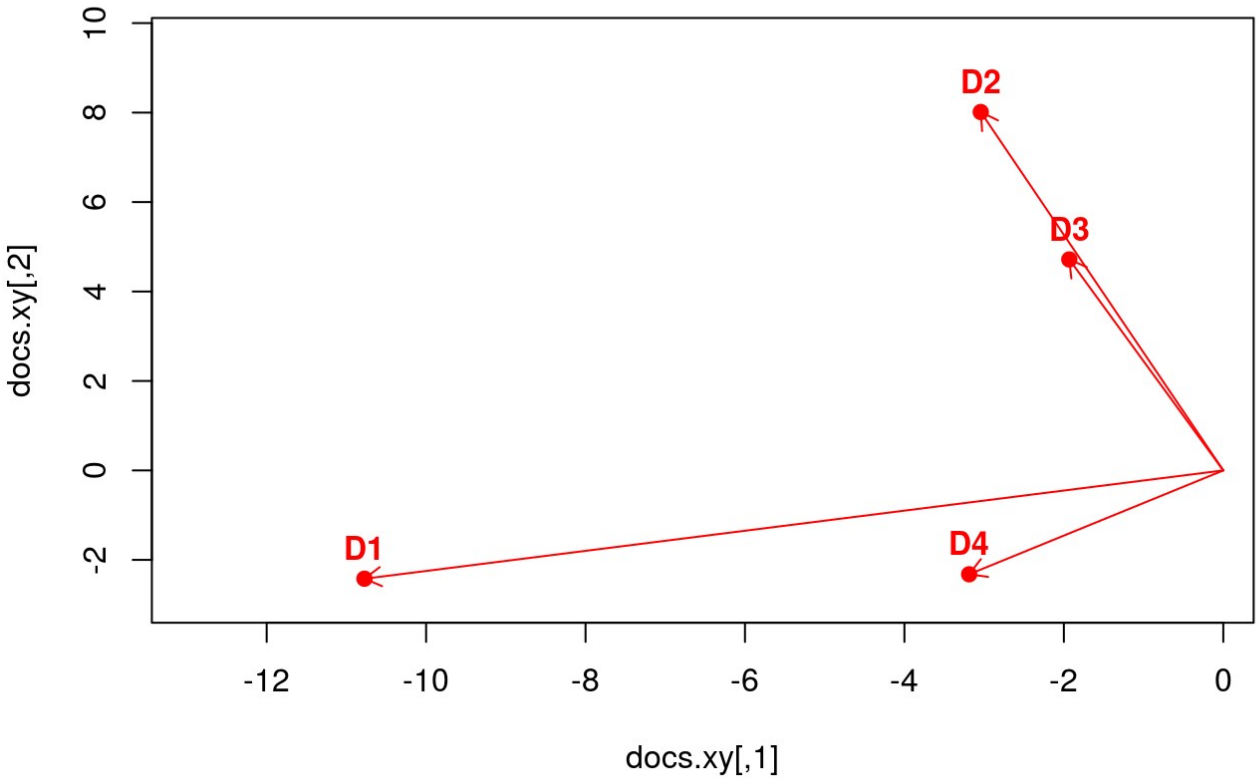
```
##                1      2      3     4      5 query1.xy
## 1          1.000  0.999  0.939 0.109 -0.353     0.854
## 2          0.999  1.000  0.927 0.076 -0.383     0.837
## 3          0.939  0.927  1.000 0.445 -0.009     0.981
## 4          0.109  0.076  0.445 1.000  0.892     0.610
## 5         -0.353 -0.383 -0.009 0.892  1.000     0.185
## query1.xy  0.854  0.837  0.981 0.610  0.185     1.000
```

Oczywiście, powstaje pytanie, czy tego typu przekształcenia działają też w przypadku większych dokumentów. Tym razem "na tapetę" pójda cztery teksty, które wykorzystano na wykładzie

```
# PRZYKŁAD 8.9

d1 <- "If you want to cause a commotion in any psychology department or any other plac
e where animal and human behaviour is studied, all that you have to do is to claim tha
t your dog loves you. Skeptics, critics, and even some ardent supporters will pour out
into the halls to argue the pros and cons of that statement. Among the skeptics you wi
ll find the veterinarian Fred Metzge, of Pennsylvania State University, who claims tha
t dogs probably don't feel love in the typical way humans do. Dogs make  investments i
n human beings because it works for them. They have something to gain from putting so-
called emotions out there.  Metzger believes that dogs 'love' us only as long as we co
ntinue to reward their behaviours with treats and attention. For most dog owners, howe
ver, there is little doubt that dogs can truly love people."

d2 <- "Emotions guide our lives in a million ways. Whether we're inclined to hide and
avoid or ponder and express them, most of us don't realize the extent to which they ar
e driving our thoughts and behavior. Exploring our emotions is a worthy endeavor for a
nyone hoping to know and develop themselves, build healthy relationships, and pursue w
hat they want in life. Recent research has even suggested that emotional intelligence
is more important than IQ, showing that it 'predicts over 54% of the variation in succ
ess' in relationships, health, and quality of life. Our emotions can offer us clues in
to who we are as well as how we've been affected by our history. Many of our actions a
re initiated by emotion, which leads to the natural question of what emotions are bein
g surfaced and why."

d3 <- "Curiosity is part of human nature. One of the first questions children learn to
ask is 'why?' As adults, we continue to wonder. Using empirical methods, psychologists
apply that universal curiosity to collect and interpret research data to better unders
tand and solve some of society's most challenging problems. It's difficult, if not imp
ossible, to think of a facet of life where psychology is not involved. Psychologists e
mploy the scientific method - stating the question, offering a theory and then constru
cting rigorous laboratory or field experiments to test the hypothesis. Psychologists a
pply the understanding gleaned through research to create evidence-based strategies th
at solve problems and improve lives."

d4 <- "Olga, a 22-year-old woman in Saratov, Russia took her dog and her baby son Vadi
m to a park and met up with friends. After a few drinks, Olga went home and left her b
aby behind! Luckily, her dog Lada was with the baby. Olga woke the next morning and re
alized the child was missing. She thought Vadim had been abducted, but her father went
to the park and found the baby in his pram, with Lada still beside him. The rottweiler
had stood guard over him all night long. Vadim was wet and hungry, but unharmed,
and was placed in the care of his grandmother."
```

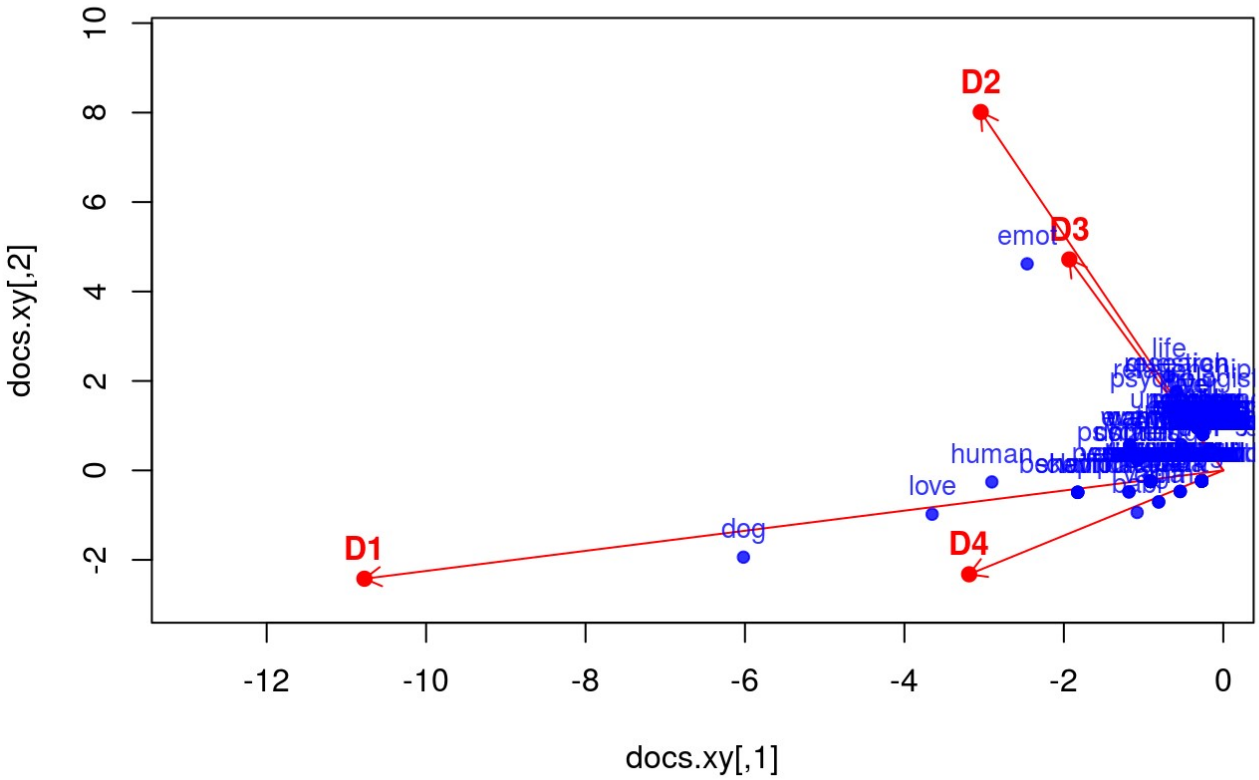
Dalsza procedura jest identyczna jak w poprzednim przykładzie.

```
# PRZYKŁAD 8.10

data <- tibble(doc_id = 1:4, text = c(d1, d2, d3, d4))

corpus <- VCorpus(DataframeSource(as.data.frame(data)))

corpus <- corpus %>% tm_map(removeWords, stopwords("english")) %>%
  tm_map(removePunctuation) %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(content_transformer(removeSpecialChars)) %>%
  tm_map(stripWhitespace) %>%
  tm_map(stemDocument)

doc <- TermDocumentMatrix(corpus)

doc.lsa <- lsa(doc, 2)

docs.xy <- t(diag(doc.lsa$sk) %*% t(doc.lsa$dk))
terms.xy <- doc.lsa$tk %*% diag(doc.lsa$sk)

max.xy <- apply(rbind(apply(docs.xy, 2, max), apply(terms.xy, 2, max)), 2, max)
min.xy <- apply(rbind(apply(docs.xy, 2, min), apply(terms.xy, 2, min)), 2, min)

max.xy <- max.xy + 0.2 * abs(max.xy)
min.xy <- min.xy - 0.2 * abs(min.xy)

plot(docs.xy, pch = 19, col = "red", xlim = c(min.xy[1], max.xy[1]), ylim = c(min.xy[2
], max.xy[2]))
text(docs.xy, col = "red", label = c("D1", "D2", "D3", "D4"), font = 2, pos = 3)
arrows(0, 0, docs.xy[,1], docs.xy[,2], length = 0.1, col = "red")
```



W tym wypadku, rzecz jasna, jest dużo więcej słów:

```
# PRZYKŁAD 8.11
points(terms.xy, pch = 19, cex = 0.8, col = rgb(0, 0, 1, 0.8))
text(terms.xy, label=rownames(terms.xy), cex = 0.85, col = rgb(0, 0, 1, 0.8), pos = 3)
```



Latent Dirichlet Allocation

W odróżnieniu od LSA, metoda *Latent Dirichlet Allocation* (**LDA**) ma dość skomplikowane podłoże matematyczne. Ogólnie mówiac, tak jak i w innych podobnych metodach zakładamy, że:

- każdy dokument jest mieszaniną tematów, i
- każdy temat jest mieszanką słów.

LDA zakłada pewne postaci rozkładów statystycznych przynalezności dokumentów do tematów oraz słów do tematów. Już ten opis automatycznie sugaruje, że LDA jest metodą, w której w zależności od ziarna generatora liczb losowych otrzymamy różne wyniki.

Skorzystamy z pakietu **topicmodels** i jego głównej funkcji **LDA()**, ustawiając 2 tematy i wypsując, które słowa (po 10 najistotniejszych) zostaną przypisane do którego tematu oraz który temat do którego dokumentu.

```
# PRZYKŁAD 8.12

library(topicmodels)

doc <- DocumentTermMatrix(corpus)

doc.lda <- LDA(doc, k = 2)
terms(doc.lda, 10)
```

```
##       Topic 1        Topic 2
##  [1,] "dog"          "emot"
##  [2,] "human"        "babi"
##  [3,] "love"         "olga"
##  [4,] "psychologist" "vadim"
##  [5,] "appli"        "lada"
##  [6,] "behaviour"    "life"
##  [7,] "claim"        "park"
##  [8,] "continu"      "realiz"
##  [9,] "curios"       "relationship"
## [10,] "method"       "thought"
```

```
topics(doc.lda)
```

```
## 1 2 3 4
## 1 2 1 2
```

Podobnie jak w poprzednim przypadku, dokumenty d_1 i d_4 zostały uznane za bliskie sobie. Tak samo metoda potraktowała d_3 i d_4. Ciekawe może być porównanie z przypadkiem gdy $k = 3$

```
# PRZYKŁAD 8.13

doc.lda <- LDA(doc, k = 3)
terms(doc.lda, 10)
```

```
##       Topic 1        Topic 2     Topic 3
##  [1,] "psychologist" "dog"       "emot"
##  [2,] "appli"        "babi"      "life"
##  [3,] "curios"       "love"      "relationship"
##  [4,] "method"       "human"     "action"
##  [5,] "problem"      "olga"      "affect"
##  [6,] "question"     "vadim"     "anyon"
##  [7,] "research"     "behaviour" "avoid"
##  [8,] "solv"         "claim"     "behavior"
##  [9,] "understand"   "lada"      "build"
## [10,] "adult"        "long"      "can"
```

```
topics(doc.lda)
```

```
## 1 2 3 4
## 2 3 1 2
```

Generalnie, zarówno dla słów jak i dokumentów mamy po prostu wartości prawdopodobieństw a posteriori przynależności do konkretnego zbioru

```
# PRZYKŁAD 8.14

t(posterior(doc.lda)$terms)[1:10,] %>% round(4)
```

```
##                1      2      3
## 22yearold 0.0000 0.0075 0.0000
## abduct    0.0000 0.0075 0.0000
## action    0.0000 0.0000 0.0161
## adult     0.0156 0.0000 0.0000
## affect    0.0000 0.0000 0.0161
## after     0.0000 0.0075 0.0000
## among     0.0000 0.0075 0.0000
## anim      0.0000 0.0075 0.0000
## anyon     0.0000 0.0000 0.0161
## appli     0.0313 0.0000 0.0000
```

```
posterior(doc.lda)$topics %>% round(4)
```

```
##        1      2      3
## 1 0.0003 0.9994 0.0003
## 2 0.0003 0.0003 0.9993
## 3 0.9993 0.0003 0.0003
## 4 0.0004 0.9993 0.0004
```

Oczywiście, to są jedynie suche liczby. Wykorzystując potencjał poznanych pakietów można wykonać bardziej sugestywne grafiki - poniżej użyjemy dużego zbioru AP (Associated Press)

```
# PRZYKŁAD 8.15
library(tidytext)
library(ggplot2)
```

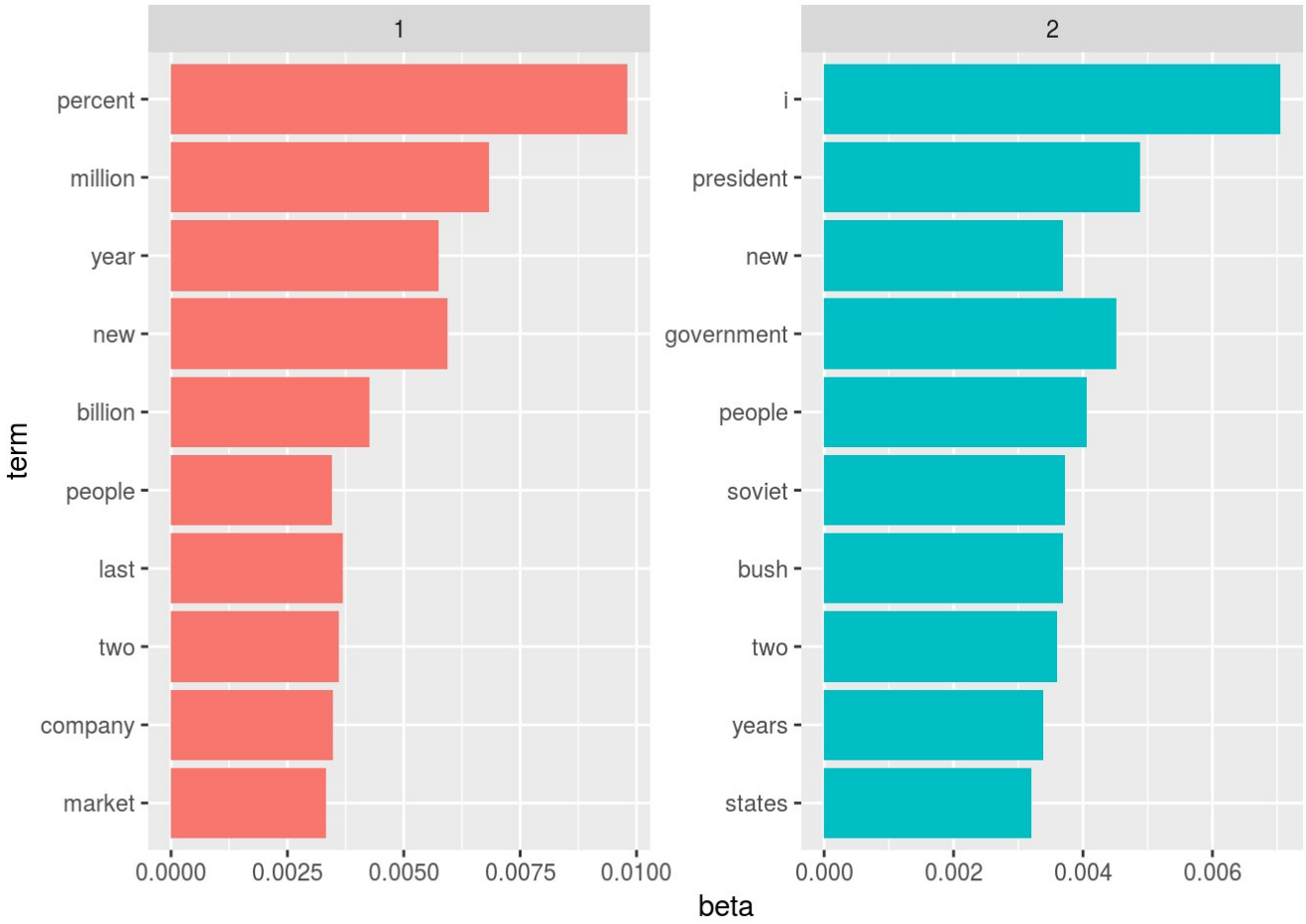
```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
data("AssociatedPress")
AssociatedPress
```

```
## <<DocumentTermMatrix (documents: 2246, terms: 10473)>>
## Non-/sparse entries: 302031/23220327
## Sparsity           : 99%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
ap_lda <- LDA(AssociatedPress, k = 2, control = list(seed = 1234))
ap_topics <- tidy(ap_lda, matrix = "beta")

ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



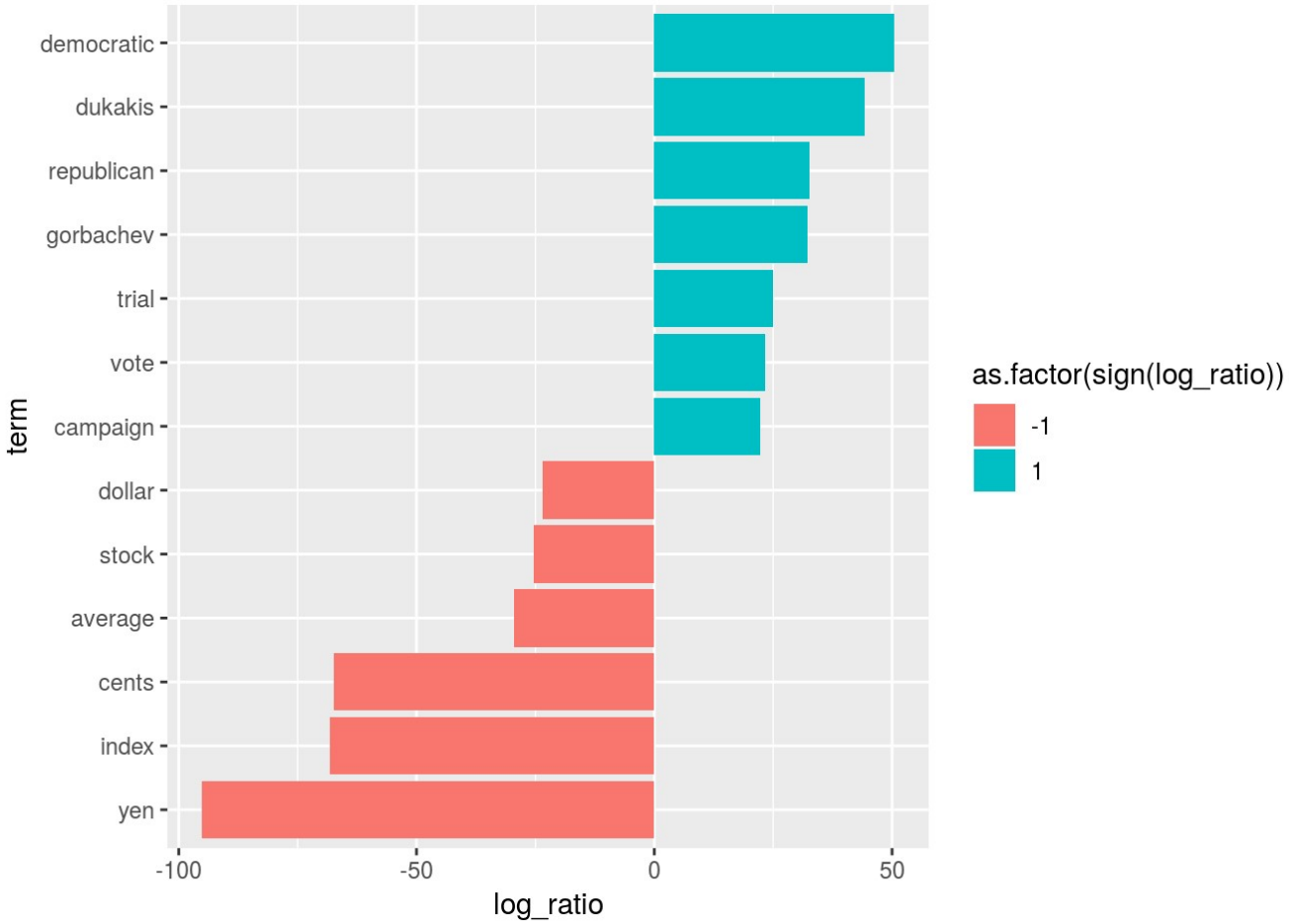
Występujący parametr β jest tym samym, co wspomniane prawdopodobieństwo a posteriori. Na jego bazie, porównujac $\log(\beta_1/\beta_2)$ możemy pokazać które słowa mają najwiekszą róznicę w przynalezności do poszczególnych tematów.

```
# PRZYKŁAD 8.16

library(tidyr)

beta_spread <- ap_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  spread(topic, beta) %>%
  filter(topic1 > .001 | topic2 > .001) %>%
  mutate(log_ratio = log2(topic2 / topic1))

beta_spread
```

```
## # A tibble: 198 x 4
##    term              topic1      topic2 log_ratio
##    <chr>              <dbl>       <dbl>     <dbl>
##  1 administration 0.000431  0.00138         1.68
##  2 ago            0.00107   0.000842       -0.339
##  3 agreement      0.000671  0.00104         0.630
##  4 aid            0.0000476 0.00105         4.46
##  5 air            0.00214   0.000297       -2.85
##  6 american       0.00203   0.00168        -0.270
##  7 analysts       0.00109   0.000000578   -10.9
##  8 area           0.00137   0.000231       -2.57
##  9 army           0.000262  0.00105         2.00
## 10 asked          0.000189  0.00156         3.05
## # ... with 188 more rows
```

na koniec wizualizując te wyrazy, które mają wyjątkowo dużą różnicę

```
# PRZYKŁAD 8.17

beta_spread %>%
  arrange(log_ratio) %>%
  mutate(term = reorder(term, log_ratio)) %>%
  filter(abs(log_ratio) >20) %>%
  ggplot() +
  geom_col(aes(term, log_ratio, fill = as.factor(sign(log_ratio)))) +
  coord_flip()
```


democratic
dukakis
republican
gorbachev
trial
vote
campaign
dollar
stock
average
cents
index
yen
term
-100
-50
0
50
log_ratio
as.factor(sign(log_ratio))
-1
1