Eksploracja tekstu i wyszukiwanie informacji w mediach społecznościowych

LABORATORIUM 7

- Analiza sentymentu pod nadzorem
- Zbiór danych i wstępna obróbka
- Wybór cech
- Użycie klasyfikatora

Analiza sentymentu pod nadzorem

Jak już wspomniano na poprzednich zajęciach, alternatywą do metod słownikowych analizy sentymentu jest **analiza pod nadzorem**. Jest to dość ogólna nazwa i nie odnosi się do żadnej konkretnej metody, ani też do sposobu reprezentacji tekstu. Jednakże, najczęściej myśli się tu o przedstawieniu tekstu pojedynczego dokumentu w formie **modelu przestrzeni wektorowej**, co w przypadku rozpatrywania wielu dokumentów prowadzi do **macierzy termów-dokumentów**. Jeśli zaś chodzi o kwestię metody, to jest tu pełna swoboda, możemy wybierać spomiędzy analizy dyskryminacyjnej, drzew, maszyn wektorów podpierających etc.

Zbiór danych

Podstawowym problemem metod pod nadzorem jest, że aby je wykorzystać, musimy być w posiadaniu **uprzednia sklasyfikowanego** zbioru danych. Podczas gdy dostępnych słowników jest co najmniej kilka(naście), w przypadku dokumentów może już nie być aż tak łatwo. My trochę nagniemy rzeczywistość i skorzystamy z danych sklsyfikowanych, ale nie przez ludzi - tzw. kompetentnych sędziów, ale własnie przez program. Użyjemy danych z Forum BBC, związanych z dyskusjami na tematy ogólne, odnoszące się do aktualnych (wtedy, rzecz jasna) wiadomości.

```
# PRZYKŁAD 7.1

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(magrittr)

data.bbc <- as_tibble(read.table("http://www.fizyka.pw.edu.pl/~julas/TEXT/lab/data_bbc
.csv", stringsAsFactors = F))

data.bbc
```

```
## # A tibble: 8,867 x 2
##    text                                                                emo
##  * <chr>                                                             <int>
##  1 The world keeps quite as 21 synogogues are destroyed by this so .    -1
##  2 "     Where?  Could you post a link or source?"                      -1
##  3 he means the ones in the occupied territory of Gaza                  -1
##  4 How about the small mention on the BBC website that Palastians c.    -1
##  5 "     They're just buildings. Get a life, son.  Burned or bulldo.    -1
##  6 "     Why were the only buildings left standing in the former il.    -1
##  7 Bricks, Concrete, Steel, Glass... Put them together and build a .    -1
##  8 "Why start YET ANOTHER thread on this subject?  There are alread.    -1
##  9 there are no longer any users of those Synagogues, having left t.    -1
## 10 "who is \"we\" ?"                                                    -1
## # ... with 8,857 more rows
```

```
data.bbc %>%
  group_by(emo) %>%
  summarise(n = n())
```

```
## # A tibble: 3 x 2
##     emo     n
##   <int> <int>
## 1    -1  6269
## 2     0  1382
## 3     1  1216
```

Jak widać powyżej, zbiór danych zawiera prawie 9000 rekordów, przy czym występują w nim wartości pozytywne (1), negatywne (-1) oraz obiektywne (0). Na nasze potrzeby okroimy ten zbiór do około 1000 rekordów, pozostawiając jedynie posty pozytywne i negatywne i mniej więcej utzymując równe proporcje pomiędzy nimi. Poza tym dokonamy jeszcze dwóch operacji: zmienimy kodowanie na UTF-8 oraz zmienna *emo* zmienimy na typ wyliczeniowy. Dodatkowo dokonamy jeszcze przetasowania danych.

```
# PRZYKŁAD 7.2

data.bbc %<>%
  arrange(desc(emo)) %>%
  slice(-c(501:(n()-498)))

data.bbc$text <- sapply(data.bbc$text, enc2utf8)
data.bbc$emo <- as.factor(data.bbc$emo)

data.bbc <- data.bbc[sample(1:nrow(data.bbc)),]
data.bbc$doc_id <- 1:nrow(data.bbc)
```

Wybór cech

Wzmiankowanymi **cechami** (*features*) $f_1, \ldots, f_m$ są w naszym przypadku poszczególne termy, czyli wyrazy, występujące w dokumentach. Zakładamy tu, że po prostu każdy dokument $\mathbf{d}$ można zapisać w formie $\mathbf{d} = [n_1(d), n_2(d), \ldots, n_m(d)]$, gdzie $n_i(d)$ jest liczbą wystąpień f_i w dokumencie. Z technicznego punktu widzenia sprowadza się to do stworzenia macierzy termów-dokumentów, do którego to zadania wykorzystamy pakiet **tm**.

```
# PRZYKŁAD 7.3
library(tm)
```

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

```
source <- DataframeSource(as.data.frame(data.bbc))

corpus <- VCorpus(source)

corpus %<>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removePunctuation) %>%
  tm_map(stripWhitespace) %>%
  tm_map(removeNumbers)

tdm <- DocumentTermMatrix(corpus)

tdm
```

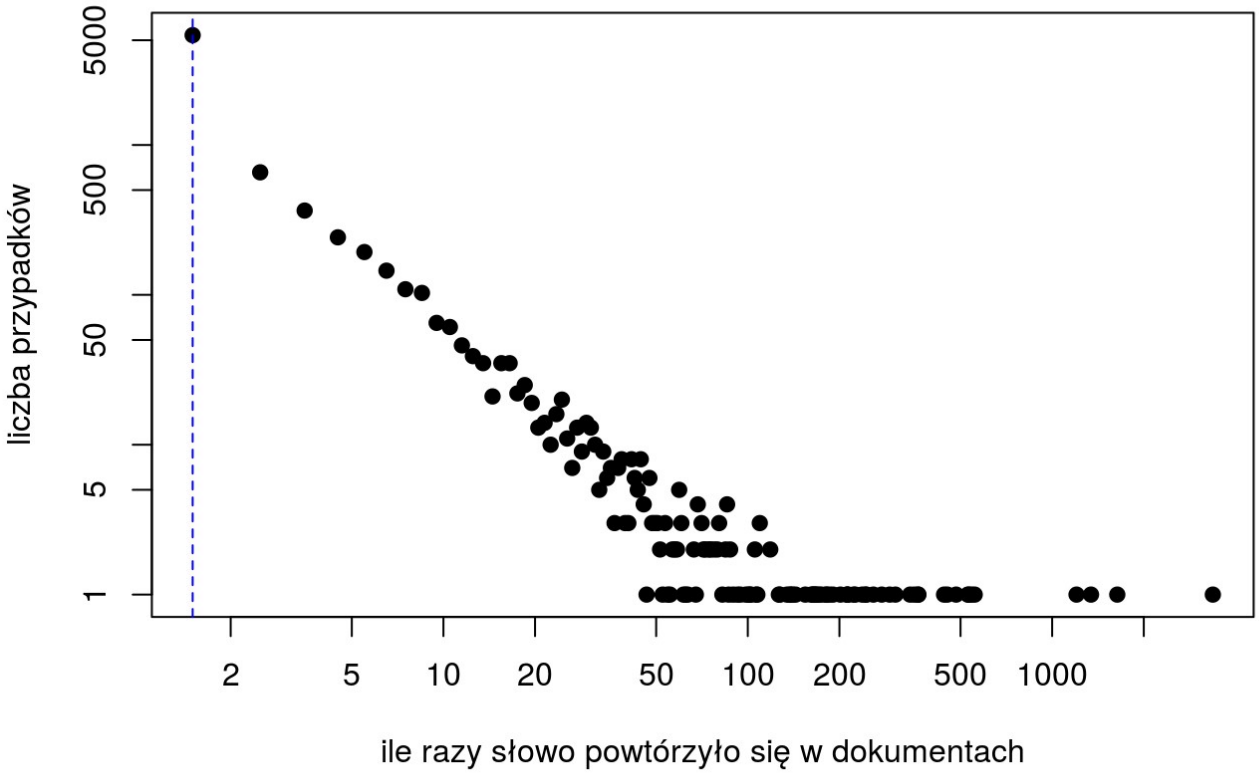
```
## <<DocumentTermMatrix (documents: 998, terms: 7993)>>
## Non-/sparse entries: 39046/7937968
## Sparsity           : 100%
## Maximal term length: 88
## Weighting          : term frequency (tf)
```

Otrzymaliśmy macierz 1000x8000 elementów. To dość sporo, bo oznacza, że nasz klasyfikator będzie pracował na 8000-wymiarowej przestrzeni, a w końcu nie będziemy używac serwera obliczeniowego tylko komputerów w pracowni :-) Poza tym, spora część słów występuje tylko w pojedynczych dokumentach:

```
# PRZYKŁAD 7.4

tdm.count <- apply(tdm, 2, sum)

h <- hist(tdm.count, breaks = max(tdm.count), plot = F)

plot(h$mids, h$counts, log="xy", pch = 19, xlab = "ile razy słowo powtórzyło się w dok
umentach", ylab="liczba przypadków")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 3233 y values <= 0 omitted
## from logarithmic plot
```

```
abline(v = 1.5, col = "blue", lty = 2)
```



Z tego też powodu "okroimy" naszą macierz, zatrzymując jedynie słowa, które pojawiły się **co najmniej 5 razy**

```
# PRZYKŁAD 7.5

tdm <- tdm[, apply(tdm, 2, sum) > 4]
tdm
```

```
## <<DocumentTermMatrix (documents: 998, terms: 1567)>>
## Non-/sparse entries: 29550/1534316
## Sparsity           : 98%
## Maximal term length: 16
## Weighting          : term frequency (tf)
```

Pozbyliśmy się prawie 85% cech. Na tym etapie trzeba przeprowadzić jeszcze jedno sprawdzenie: przy poprzedniej operacji mogliśmy przypadkowo doprowadzić do takiej sytuacji, że z niektórych dokumentów usunęliśmy wszystkie wyrazy lub zostawiliśmy tylko jeden. Warto pozbyć się takich przypadków, pamiętając jednocześnie o tym, że musimy zaktualizować nasz wektor wartości emocjonalnych, który przy okazji zapiszemy do oddzielnej zmiennej.

```
# PRZYKŁAD 7.6

tdm <- as.matrix(tdm)

ind <- apply(tdm, 1, sum) > 1

tdm <- tdm[ind, ]
class <- data.bbc$emo[ind]

dim(tdm); length(class)
```

```
## [1]  960 1567
```

```
## [1] 960
```

Użycie klasyfikatora

Zatem mamy macierz 960x1567 i możemy przystąpić do wykorzystania metod uczenia pod nadzorem. Za każdym razem schemat jest podobny:

- uczymy wybrany klasyfikator na zestawie danych,
- następnie za jego pomocą przewidujemy klasę danych,
- oceniamy skuteczność dzieląc diagonalę macierzy pomyłek przez wszystkie wystąpienia.

W pierwszym kroku dokonamy tzw. ponownego podstawienia, czyli użycia tych samych danych zarówno do nauki, jak i do oceny. Jest to ekstremalny przypadek, dając nader optymistyczną estymację błędu

Zaczynamy od **klasyfikatora LDA** z pakietu **MASS**.

```
# PRZYKŁAD 7.7
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
CM <- function(org.class, pred.class) {

  CM <- table(org.class, pred.class)
  return(sum(diag(CM)) / sum(CM))
}

bbc.lda <- lda(tdm, class)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
bbc.lda.pred <- predict(bbc.lda, tdm)

table(class, bbc.lda.pred$class)
```

```
##
## class  -1   1
##    -1 469   2
##    1    1 488
```

```
CM(class, bbc.lda.pred$class)
```

```
## [1] 0.996875
```

a następnie użyjemy dwóch wersji klasyfikatora SVM (pakiet **e1071**), czyli maszyn wektorów podpierających: jednego z jądrem liniowym, drugiego z radialnym.

```
# PRZYKŁAD 7.8
library(e1071)

bbc.svml <- svm(tdm, class, type = "C-classification", kernel = "linear")
bbc.svml.pred <- predict(bbc.svml, tdm)

table(class, bbc.svml.pred)
```

```
##      bbc.svml.pred
## class  -1   1
##    -1 471   0
##    1    0 489
```

```
CM(class, bbc.svml.pred)
```

```
## [1] 1
```

```
bbc.svmr <- svm(tdm, class, type = "C-classification")
bbc.svmr.pred <- predict(bbc.svmr, tdm)

table(class, bbc.svmr.pred)
```

```
##       bbc.svmr.pred
## class  -1   1
##    -1 471   0
##    1  135 354
```

```
CM(class, bbc.svmr.pred)
```

```
## [1] 0.859375
```

Skuteczność użytych klasyfikatorów jest bardzo duża, ale to niewątpliwie efekt powtórnego podstawienia. Żeby tego uniknąć, można wykonać np. korswalidację, czyli podzielić zbiór danych na kawałki, następnie sekwencyjnie uczyć na jednej części (90%) i testować na innej (np. 10%). Metodę oprzemy o SVM z liniowym jądrem.

```
# PRZYKŁAD 7.9

CV <- function(data, class, K) {

  N <- nrow(data)

  # Dane przetasowane
  ind <- sample(1:N)
  data.rnd <- data[ind,]
  class <- class[ind]

  # Tworzenie K pseudoprób
  sets <- sapply(1:K, function(i) ((i-1) * (N/K) + 1):(i * (N/K)))

  # Przypadek K = 1
  if(is.vector(sets)) sets <- t(as.matrix(sets))

  # Dla każdej pseudopróby wyznaczamy liczbę pomyłek
  res <- t(sapply(1:K, function(k) CV.main(data.rnd[-c(sets[,k]),], class[-c(sets[,k])
], data.rnd[sets[,k],], class[sets[,k]])))

  res
}

# Główna funkcja odpowiedzialna za CV
# przyjmuje PU (jedna z pseudoprób) oraz PT
CV.main <- function(learn, class_learn, test, class_test) {

  learn.classifier <- svm(learn, class_learn, type = "C-classification", kernel = "lin
ear")
  test.pred <- predict(learn.classifier, newdata = test)

  # Macierz pomyłek
  CM <- table(class_test, test.pred)

  # Liczba błędów
  # print(sum(CM) - sum(diag(CM)))
  sum(CM) - sum(diag(CM))
}

1 - sum(CV(tdm, class, 10)) / nrow(tdm)
```

```
## Warning in svm.default(learn, class_learn, type = "C-classification",
## kernel = "linear"): Variable(s) 'unto' and 'wilderness' constant. Cannot
## scale data.
```

```
## Warning in svm.default(learn, class_learn, type = "C-classification",
## kernel = "linear"): Variable(s) 'islamofascism' constant. Cannot scale
## data.
```

```
## Warning in svm.default(learn, class_learn, type = "C-classification",
## kernel = "linear"): Variable(s) 'engineers' constant. Cannot scale data.
```

```
## [1] 0.7458333
```

Jak widać, otrzymujemy ponad 70% skuteczność, co daje dużo więcej niż ok. 50% skutecznośc losowego trafu, czy nawet ludzka ewaluacja (zwykle zgodność rzędu 60-65%).