PLMR - projekt

Klasyfikacja tweetów na pozytywne, negatywne

Jakub Sękulski

1. Wstęp

Celem projektu jest przeprowadzenie analizy tweetów i klasyfikacji ich na pozytywne, negatywne. Zbiór danych koniecznych do przeprowadzenia analizy został pobrany ze strony linked phrase i stanowi 1.6 miliona tweetów. Program został nauczony za pomocą regresji liniowej z walidacją krzyżową. Po nauczeniu program dodatkowo sprawdzi tweety pobrane z bazy tweetera i sklesyfikuje je.

2. Wczytanie niezbednych bibliotek i danych

```
library(twitteR)
library(ROAuth)
library(tidyverse)

## ── Attaching packages
─────────────────────────────────────────────────────── tidyverse
1.3.0 ──

## ✓ ggplot2 3.3.3     ✓ purrr   0.3.3
## ✓ tibble  2.1.3     ✓ dplyr   0.8.4
## ✓ tidyr   1.0.2     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.4.0

## ── Conflicts
────────────────────────────────────────────────────────────────────
tidyverse_conflicts() ──
## x dplyr::filter()   masks stats::filter()
## x dplyr::id()       masks twitteR::id()
## x dplyr::lag()      masks stats::lag()
## x dplyr::location() masks twitteR::location()

library(text2vec)
library(caret)

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 3.0-2

library(ggrepel)
library(purrrlyr)
library(base64enc)
library(ggplot2)

twitterTexts <- read_csv("/home/sekuba101/rExercises/Machine Learning
RRRR/data_set/tweetsData.csv",
                         col_names = c('sentiment', 'id', 'date',
'query', 'user', 'text'))

## Parsed with column specification:
## cols(
##   sentiment = col_double(),
##   id = col_double(),
##   date = col_character(),
##   query = col_character(),
##   user = col_character(),
##   text = col_character()
## )

print(twitterTexts)

## # A tibble: 1,600,000 x 6
##    sentiment       id date          query  user     text

##        <dbl>    <dbl> <chr>         <chr>  <chr>    <chr>

##  1         0   1.47e9 Mon Apr 06 2… NO_QU… _TheSpe… @switchfoot
http://twitpic.…
##  2         0   1.47e9 Mon Apr 06 2… NO_QU… scottha… is upset that
he can't upda…
##  3         0   1.47e9 Mon Apr 06 2… NO_QU… mattycus @Kenichan I
dived many time…
##  4         0   1.47e9 Mon Apr 06 2… NO_QU… ElleCTF  my whole body
feels itchy a…
##  5         0   1.47e9 Mon Apr 06 2… NO_QU… Karoli
@nationwideclass no, it's n…
##  6         0   1.47e9 Mon Apr 06 2… NO_QU… joy_wolf @Kwesidei not
the whole crew
##  7         0   1.47e9 Mon Apr 06 2… NO_QU… mybirch  Need a hug
```

```
##  8         0   1.47e9 Mon Apr 06 2… NO_QU… coZZ     @LOLTrish hey
long time no…
##  9         0   1.47e9 Mon Apr 06 2… NO_QU… 2Hood4H… @Tatiana_K nope
they didn't…
## 10         0   1.47e9 Mon Apr 06 2… NO_QU… mimismo  @twittera que
me muera ?
## # … with 1,599,990 more rows
```

3. Przygotowanie danych

1. Zamiana niektórych symboli z kodu latin1 na ASCII (better save than sorry)

```r
#konwersja symboli
conv_fun <- function(x) iconv(x, "latin1", "ASCII", "")
twitterTexts<-dmap_at(twitterTexts,'text', conv_fun)
```

2. Zmiana wartości 4 na 1 w rubryce 'sentiment', dla pewnosci inne wartosci to zero

```r
twitterTexts<-mutate(twitterTexts, sentiment = ifelse(sentiment == 0,
0, 1))
```

3. Podział na zbiory uczący i testowy.
Ze względu na bardzo dużą liczbę danych zdecydowałem się na podział 7:3 (zbiór uczący:zbiór testowy). Warto zauważyć, że zmiana proporcji na 8:2 lub 9:1 nie powoduje znaczącej zmiany w rezultacie działania programu (dokładnosć różni się maksymalnie o 0.002)

```r
#generujemy losową liczbę - dzięki temu uzyskamy za każdym razem inny
podział danych treningowych
set.seed(5144)
#generujemy losowo indeksy ze zioru twitterTexts które użyjemy w
klasyfikacji jako zbiór treningowy - jako tabele(list     = FALSE)
trainIndex <- createDataPartition(twitterTexts$sentiment, p = 0.7,
                                  list = FALSE,
                                  times = 1,
                                  )
tweets_train <- twitterTexts[trainIndex, ]
tweets_test <- twitterTexts[-trainIndex, ]
```

4. Tokenizacja i słownik
W tym kroku dzielimy znaki na słowa przy okazji zmieniając wszystkie duże litery w małe. Następnie tworzymy słownik pokazujący ile razy wystąpiło dane słowo.

```r
```

```
#funkcja przygotowujaca zmieniajaca wszystkie duze litery w małe
prep_fun <- tolower
#tokenizacja sprawia, że tekst dzielony jest na słowa, czyli z
kilkudziesięciu znaków jesteśmy w stanie stworzyć słowa na podstawie
znajdujacych sie miedzy nimi spacji
tok_fun <- word_tokenizer

it_train <- itoken(tweets_train$text,
                   preprocessor = prep_fun,
                   tokenizer = tok_fun,
                   ids = tweets_train$id,
                   progressbar = TRUE)
it_test <- itoken(tweets_test$text,
                  preprocessor = prep_fun,
                  tokenizer = tok_fun,
                  ids = tweets_test$id,
                  progressbar = TRUE)
# creating vocabulary and document-term matrix
#tworzymy słownik
#słownik określa funkcją ilości wystąpień słów w tekście
vocab <- create_vocabulary(it_train) #this function collects unique
terms and corresponding statistics.
print(vocab) #doc count - różne sekwencje w których wystepuje słowo
term_count- całkowita ilość wystąpień słowa
```

```
## Number of docs: 1120000
## 0 stopwords:  ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##                        term term_count doc_count
##      1:                 0'9          1         1
##      2:              0,140           1         1
##      3: 0,2845,2344543,00            1         1
##      4:               0,9l           1         1
##      5:           0.0.0.0            1         1
##     ---
## 559305:                 my     221603    197563
## 559306:                  a     266654    232954
## 559307:                the     366615    299312
## 559308:                 to     395481    320418
## 559309:                  i     537836    402096
```

5. Obliczenie statystycznych wag dla każdego słowa.
W tym kroku najpierw tworzymy document-term matrix (DMT) - macierz w której rzędy odpowiadają tweetom, a kolumny poszczególnym słowom. każda komórka ij pokazuje ile razy słowo j wystąpiło w tweecie i(więcej o DMT [linked phrase](term Frequency Inverse Document Frequency model)). Na podstawie tej macierzy obliczamy statystycznie

```
wagi dla kazdego słowa metodą TFIDF (więcej o tej metodzie: [linked
phrase](https://pl.wikipedia.org/wiki/TFIDF)).

```r
vectorizer <- vocab_vectorizer(vocab) #This function creates an object
(closure) which defines on how to transform list of tokens into vector
space - i.e. how to map words to indices. It supposed to be used only
as argument to create_dtm, create_tcm, create_vocabulary.

# DMT
dtm_train <- create_dtm(it_train, vectorizer)
dtm_test <- create_dtm(it_test, vectorizer)

# TFIDF
tfidf <- TfIdf$new()

dtm_train_tfidf <- fit_transform(dtm_train, tfidf)
dtm_test_tfidf <- fit_transform(dtm_test, tfidf)
```
```

4. Trening modelu

Trening modelu został wykonany za pomocą regresji liniowej z walidacją krzyżową za pomocą pakietu glmnet, który pozwala na wykonanie bardzo efektywnej procedury dopasowania metodą Lasso lub elastic-net regularization dla regresji liniowej(i nie tylko). Więcej o pakiecie linked phrase

```
t1 <- Sys.time()
#Does k-fold cross-validation for glmnet, produces a plot, and returns
a value for lambda (and gamma if relax=TRUE)
#https://cran.r-project.org/web/packages/glmnet/index.html
glmnet_classifier <- cv.glmnet(x = dtm_train_tfidf, y =
tweets_train[['sentiment']],
                               family = 'binomial',
                               # L1 penalty
                               alpha = 1,
                               # interested in the area under ROC
curve
                               type.measure = "auc",
                               # 5-fold cross-validation
                               nfolds = 5,
                               # high value is less accurate, but has
faster training
                               thresh = 1e-2,
                               # again lower number of iterations for
faster training
                               maxit = 1e2)

## Warning: from glmnet Fortran code (error code -51); Convergence for
51th lambda
```

```
## value not reached after maxit=100 iterations; solutions for larger
lambdas
## returned

## Warning: from glmnet Fortran code (error code -50); Convergence for
50th lambda
## value not reached after maxit=100 iterations; solutions for larger
lambdas
## returned

## Warning: from glmnet Fortran code (error code -50); Convergence for
50th lambda
## value not reached after maxit=100 iterations; solutions for larger
lambdas
## returned

## Warning: from glmnet Fortran code (error code -50); Convergence for
50th lambda
## value not reached after maxit=100 iterations; solutions for larger
lambdas
## returned

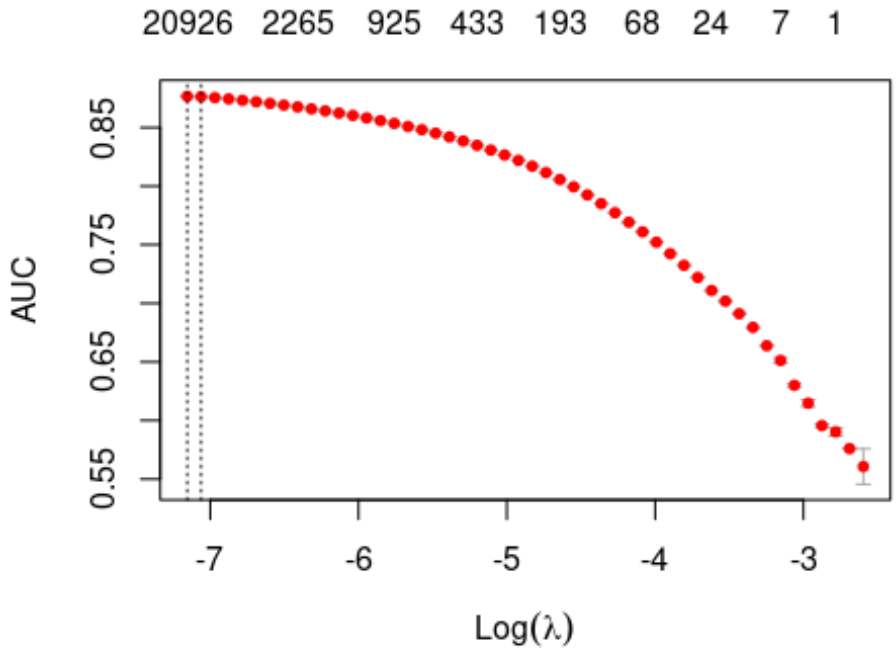
## Warning: from glmnet Fortran code (error code -52); Convergence for
52th lambda
## value not reached after maxit=100 iterations; solutions for larger
lambdas
## returned

## Warning: from glmnet Fortran code (error code -52); Convergence for
52th lambda
## value not reached after maxit=100 iterations; solutions for larger
lambdas
## returned

print(difftime(Sys.time(), t1, units = 'mins'))

## Time difference of 5.97196 mins

plot(glmnet_classifier)
```




```
print(paste("max AUC =", round(max(glmnet_classifier$cvm), 4)))
```

```
## [1] "max AUC = 0.8766"
```

```
preds <- predict(glmnet_classifier, dtm_test_tfidf, type = 'response')
[ ,1]
glmnet:::auc(as.numeric(tweets_test$sentiment), preds)
```

```
## [1] 0.876101
```

```
# zapisanie modelu do przyszłych analiz
saveRDS(glmnet_classifier, 'glmnet_classifier.RDS')
```

5. Analiza danych z tweetera

Tweeter udostępnia tweety i ich dane tweetów dla firm, oraz na cele nauki. Dzieki temu wykorzystując pakiet o zgrabnej nazwie twitteR możemy pobierać losowe tweety z bazy danych. Licencja studencka pozwala na pobranie 95 tweetów, które zostaną poddane weryfikacji w tej części projektu. Przygotowanie danych do analizy odbywa się dokładnie tak jak we wczesniejszej czesci projektu.

```
### pobieranie i analiza tweetów ###
download.file(url = "http://curl.haxx.se/ca/cacert.pem",
              destfile = "cacert.pem")
consumer_key <- 'pz3AeqUFxLJUAu291gsJ6bwjW'
consumer_secret <-
'6ztcmOU8KFHbQTPvbbXt6cC2BySwch9mTxYCRUNwmJY9mRdUog'
```

```
access_token <- '1357062011033681935-Io7jRytMN0SJO0vnpzpjMG62wCCgBH'
access_secret <- 'CYarAan8dkLf6EbeVCNCkW4e4EidWgcIjyoesjNs6lGMR'

twitteR:::setup_twitter_oauth(consumer_key, # api key
                    consumer_secret, # api secret
                    access_token, # access token
                    access_secret # access token secret
)

## [1] "Using direct authentication"

#This function will take a list of objects from a single twitteR class
and return a data.frame version of the members
df_tweets <- twitteR:::twListToDF(twitteR:::searchTwitter('setapp OR
#setapp', n = 95, lang = 'en'))


  dmap_at(df_tweets,'text', conv_fun)

#tutaj pozwoliłem sobie wyciać 18 stron macierzy df_tweets



it_tweets <- text2vec:::itoken(df_tweets$text,
                    preprocessor = prep_fun,
                    tokenizer = tok_fun,
                    ids = df_tweets$id,
                    progressbar = TRUE)

dtm_tweets <- text2vec:::create_dtm(it_tweets, vectorizer)

dtm_tweets_tfidf <- mlapi:::fit_transform(dtm_tweets, tfidf)



preds_tweets <- predict(glmnet_classifier, dtm_tweets_tfidf, type =
'response')[ ,1]
# dodanie przewidywań z regresji do pobranych z tweetera danych
df_tweets$sentiment <- preds_tweets
```

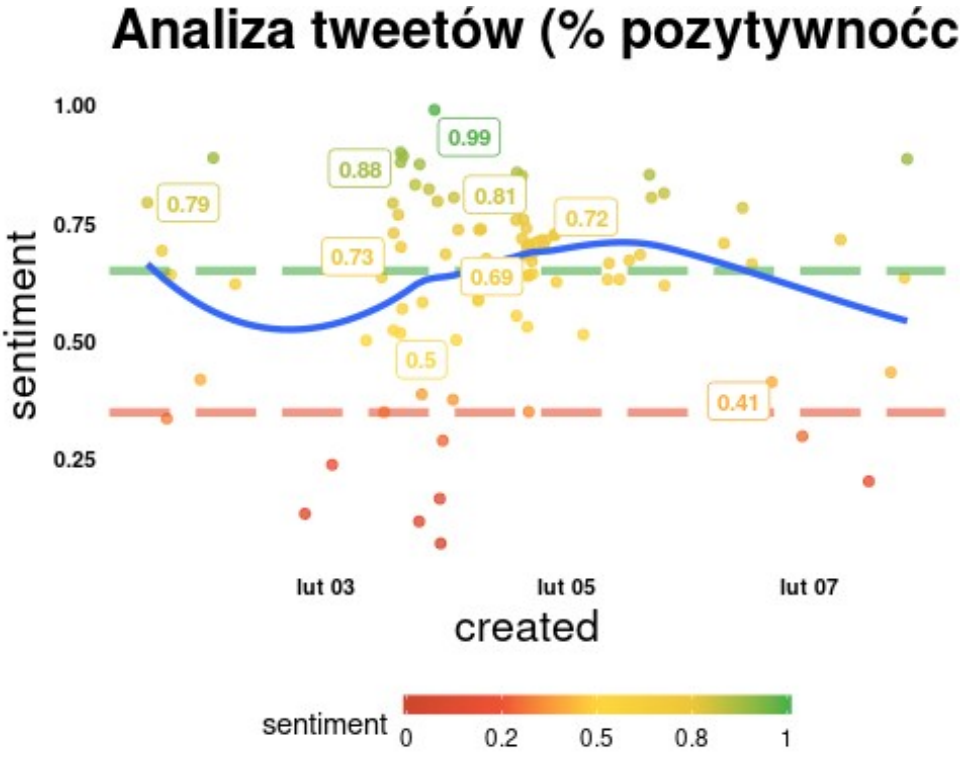
6. Wizualizacja zanalizowanych danych

Z racji na trudność w wyświetleniu danych na początku projektu stwierdziłem, że ciekawie będzie pokazać efekty działania powyższej analizy pobranych tweetów. Ciekawą obserwacją z mojej strony moze być to, że w trakcie wszystkich dokaonanych przeze mnie prób i pobrań tweety mają tendencje do bycia częściej pozytywnymi niz negatywnymi.

```
# color palette
cols <- c("#ce472e", "#f05336", "#ffd73e", "#eec73a", "#4ab04a")
set.seed(932)
samp_ind <- sample(c(1:nrow(df_tweets)), nrow(df_tweets) * 0.1) # 10%
```

```
for labeling
# plotting
ggplot2:::ggplot(df_tweets, aes(x = created, y = sentiment, color =
sentiment)) +
  theme_minimal() +
    #dodanie właściwych kolorów do punktów i paska
  scale_color_gradientn(colors = cols, limits = c(0, 1),
                        breaks = seq(0, 1, by = 1/4),
                         labels = c("0", round(1/4*1, 1), round(1/4*2,
1), round(1/4*3, 1), round(1/4*4, 1)),
                         guide = guide_colourbar(ticks = T, nbin = 50,
barheight = .5, label = T, barwidth = 10)) +
    #pokazanie punktów
  geom_point(aes(color = sentiment), alpha = 0.8) +
 #dodanie linii od których można przybliżyć, ze tweet jest
pozytywny/negatywny
  geom_hline(yintercept = 0.65, color = "#4ab04a", size = 1.5, alpha =
0.6, linetype = "longdash") +
  geom_hline(yintercept = 0.35, color = "#f05336", size = 1.5, alpha =
0.6, linetype = "longdash") +
    #dodanie linii uśredniajacej
  geom_smooth(size = 1.2, alpha = 0) +
    #dodanie przypisów do niektórych
  ggrepel:::geom_label_repel(data = df_tweets[samp_ind, ],
                    aes(label = round(sentiment, 2)),
                    fontface = 'bold',
                    size = 3,
                    max.iter = 100) +
  theme(legend.position = 'bottom',
        legend.direction = "horizontal",
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        plot.title = element_text(size = 20, face = "bold", vjust = 2,
color = 'black', lineheight = 0.8),
        axis.title.x = element_text(size = 16),
        axis.title.y = element_text(size = 16),
        axis.text.y = element_text(size = 8, face = "bold", color =
'black'),
        axis.text.x = element_text(size = 8, face = "bold", color =
'black')) +
  ggtitle("Analiza tweetów (% pozytywnośći)")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```


Analiza tweetów (% pozytywnoćc
1.00
0.75
0.50
0.25
sentiment
0.99
0.88
0.81
0.79
0.72
0.73
0.69
0.5
0.41
lut 03
lut 05
lut 07
created
sentiment
0
0.2
0.5
0.8
1