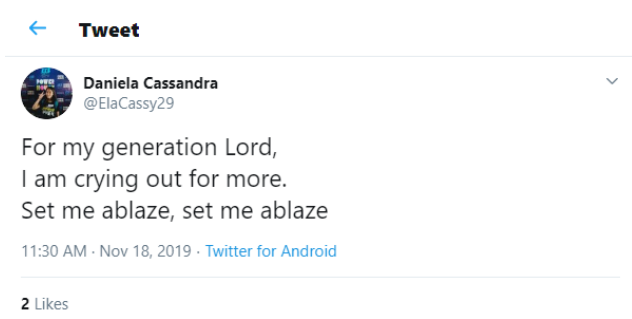
Disaster or not

Text Mining, 25-01-2020

Łukasz Księżak

Opis problemu

- Zbiór danych: **Real or Not? NLP with Disaster Tweets** (kaggle.com)
- Dwa zbiory: treningowy (7163 rekordów) i testowy (3263 rekordy)
- Dane: pliki csv [id, location, keyword, text, (target)]
- Zmienna celu: tweet opisuje faktyczne zdarzenie katastroficzne (tak bądź nie)
- Opis problemu:
  Zbudowanie modelu na podstawie danych treningowych, który będzie klasyfikował czy dany tweet opisuje faktyczną katastrofę czy nie



Przyjrzyjmy się danym

- Rzut okiem na pierwsze wiersze:

```
> head(train_data)
  id keyword location                                                                                                                    text target
1  1    <NA>     <NA>                                                                                 Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all      1
2  4    <NA>     <NA>                                                                                              Forest fire near La Ronge Sask. Canada      1
3  5    <NA>     <NA> All residents asked to 'shelter in place' are being notified by officers. No other evacuation or shelter in place orders are expected      1
4  6    <NA>     <NA>                                                                               13,000 people receive #wildfires evacuation orders in California      1
5  7    <NA>     <NA>                                                                 Just got sent this photo from Ruby #Alaska as smoke from #wildfires pours into a school      1
6  8    <NA>     <NA>                                                  #RockyFire Update => California Hwy. 20 closed in both directions due to Lake County fire - #CAfire #wildfires      1
```

- Przemyślenia:
  - Czy id będzie nam w jakikolwiek sposób potrzebne?
  - Mamy kolumny, w których brakuje danych – co z nimi?
- Akcje:
  - Usuwamy Id
  - Sprawdzamy co zrobić z [location] i [keyword]

```
> missing_train_data <- colSums(sapply(train_data, is.na))
> missing_train_data
      id  keyword location     text   target
       0       61     2533        0        0
```

  - Dużo braków w location
  - Co zrobić z keyword?

Przyjrzyjmy się danym

- Zobaczmy jak procentowo do całego zbioru mają się nasze braki

```
> # Jak to się ma procentowo do całości
> missing_train_data_prop <- missing_train_data / nrow(train_data)
>
> #keyword   location       text     target
> #0.00801261 0.33272035 0.00000000 0.00000000
>
> missing_test_data_prop <- missing_test_data / nrow(test_data)
> #    keyword    location        text
> # 0.007968127 0.338645418 0.000000000
>
```

- Ponad 30% braków w kolumnie **location** -> Usuwam kolumnę
- Ocena przydatności kolumny **keyword** względem zmiennej celu -> wydają się istotne, postanawiam je pozostawić do dalszej analizy

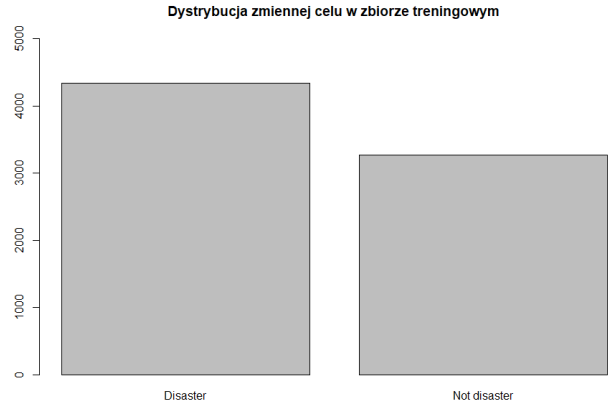
```
> library(dplyr)
> train_data$keyword <- sapply(train_data$keyword, as.character)
> train_data$keyword[is.na(train_data$keyword)] <- "no category"
>
> by_target <- train_data %>% group_by(keyword) %>% summarize(target_mean = sum(target)/n()) %>% arrange(desc(target_mean))
> View(by_target)
```

	keyword	target_mean
1	debris	1.00000000
2	derailment	1.00000000
3	wreckage	1.00000000
4	outbreak	0.97500000
5	oil%20spill	0.97368421
6	typhoon	0.97368421
7	suicide%20bombing	0.96969697
8	suicide%20bomber	0.96774194
9	bombing	0.93103448
10	rescuers	0.91428571
11	suicide%20bomb	0.91428571
12	nuclear%20disaster	0.91176471
13	evacuated	0.88888889

	keyword	target_mean
1	aftershock	0.00000000
2	body%20bags	0.02439024
3	ruin	0.02702703
4	blazing	0.02941176
5	body%20bag	0.03030303
6	electrocute	0.03125000
7	screaming	0.05555556
8	traumatised	0.05714286
9	blew%20up	0.06060606
10	panicking	0.06060606
11	blight	0.06250000
12	wrecked	0.07692308
13	explode	0.07894737

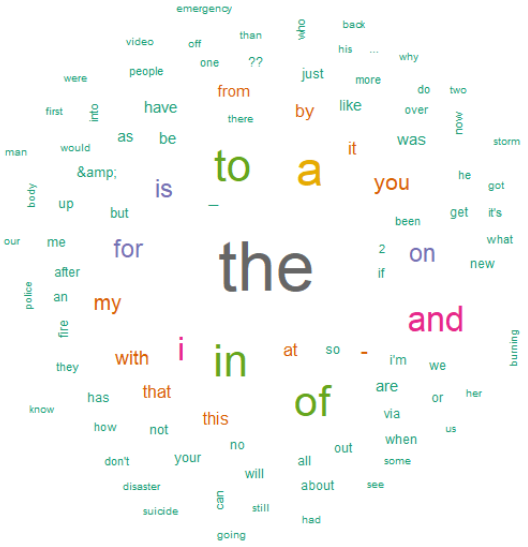
Wizualizacje

Dystrybucja zmiennej celu w zbiorze treningowym



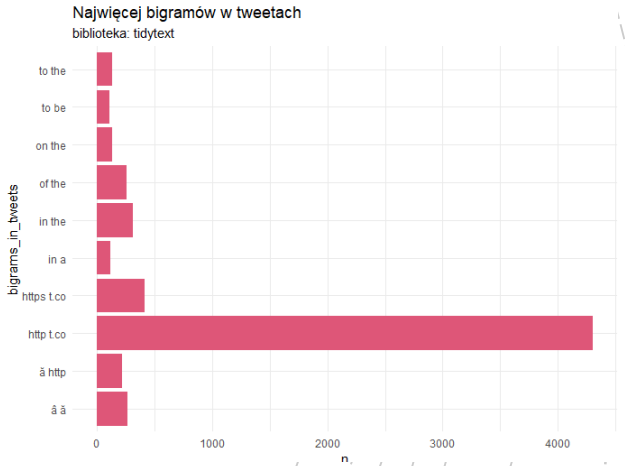
- Dystrybucja wydaje się być zrównoważona, nie widzę konieczności redystrybucji danych do nauki

Chmura słów – treść tweet-a



- Najczęstsze słowa to angielskie stop-words – koniecznie trzeba oczyścić dane

Bigramy – analiza występujących par słów



- Dzięki bigramom widzimy pary słów. Wskazują, że do oczyszczenia są również chociażby linki

Procesowanie tekstu

1

Połączenie zbiorów w jeden

- Wszystkie operacje procesowania tekstu takie same dla zbioru treningowego i testowego

```
all_tweets_df <- bind_rows(train_data, test_data)
all_data_corpus <- Corpus(VectorSource(all_tweets_df$text))
```

2

Lowercase dla treści tweetów

```
all_data_corpus <- tm_map(
  all_data_corpus, content_transformer(
    stri_trans_tolower))
```

3

Usunięcie url-ów z tweetów

```
# Ususnięcie linków (ulr-ów)
remove_urls <- function(x) {
  gsub("http[^[:space:]]*", "", x)
}

all_data_corpus <- tm_map(
  all_data_corpus, content_transformer(
    remove_urls))
```

4

Usunięcie nazw użytkowników

```
# Usunięcie nazw użyttkowników
remove_usernames <- function(x){
  gsub("@[^[:space:]]*", "", x)
}

all_data_corpus <- tm_map(
  all_data_corpus, content_transformer(
    remove_usernames))
```

5

Usunięcie słów jedno i dwu literowych

```
# Usunięcie słów jedno i dwu wyrazowych
remove_one_char <- function(x){
  gsub(" . ", " ", x)
}
remove_two_chars <- function(x){
  gsub(" .. ", " ", x)
}

all_data_corpus <- tm_map(all_data_corpus, content_transformer(remove_one_char))
all_data_corpus <- tm_map(all_data_corpus, content_transformer(remove_two_chars))

# stemming słów z textów tweetów
all_data_corpus <- tm_map(
  all_data_corpus, stemDocument, "english")
```

Procesowanie tekstu

6

Usunięcie znaków innych niż litery

```
remove_anything_but_text <- function(x){
  gsub("[^[:alpha:][:space:]]*", "", x)
}
all_data_corpus <- tm_map(
  all_data_corpus, content_transformer(
    remove_anything_but_text))
```

7

Usunięcie angielskich stop-words

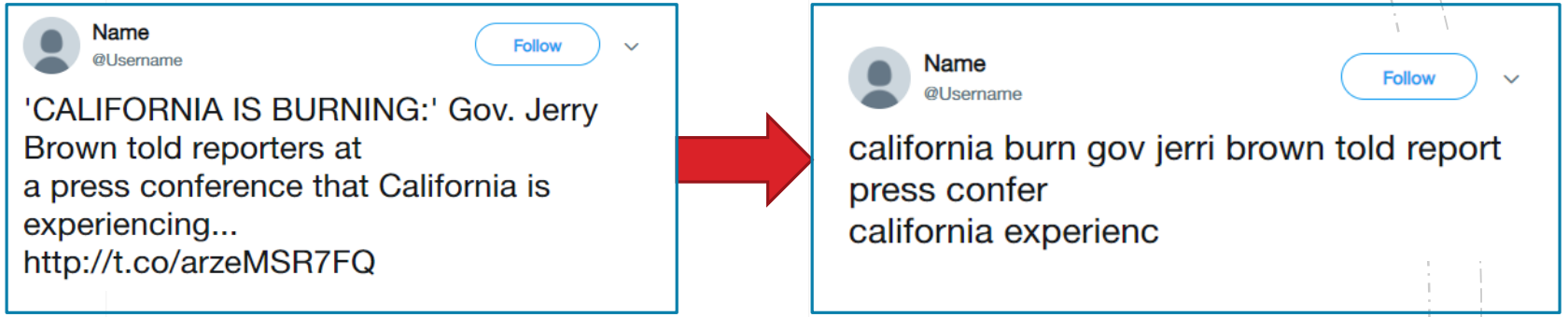
```
#install.packages('qdap')
library(qdap)
# Usunięcie angielskich stop-words
all_data_corpus <- tm_map(
  all_data_corpus, removeWords, Top200Words)
```

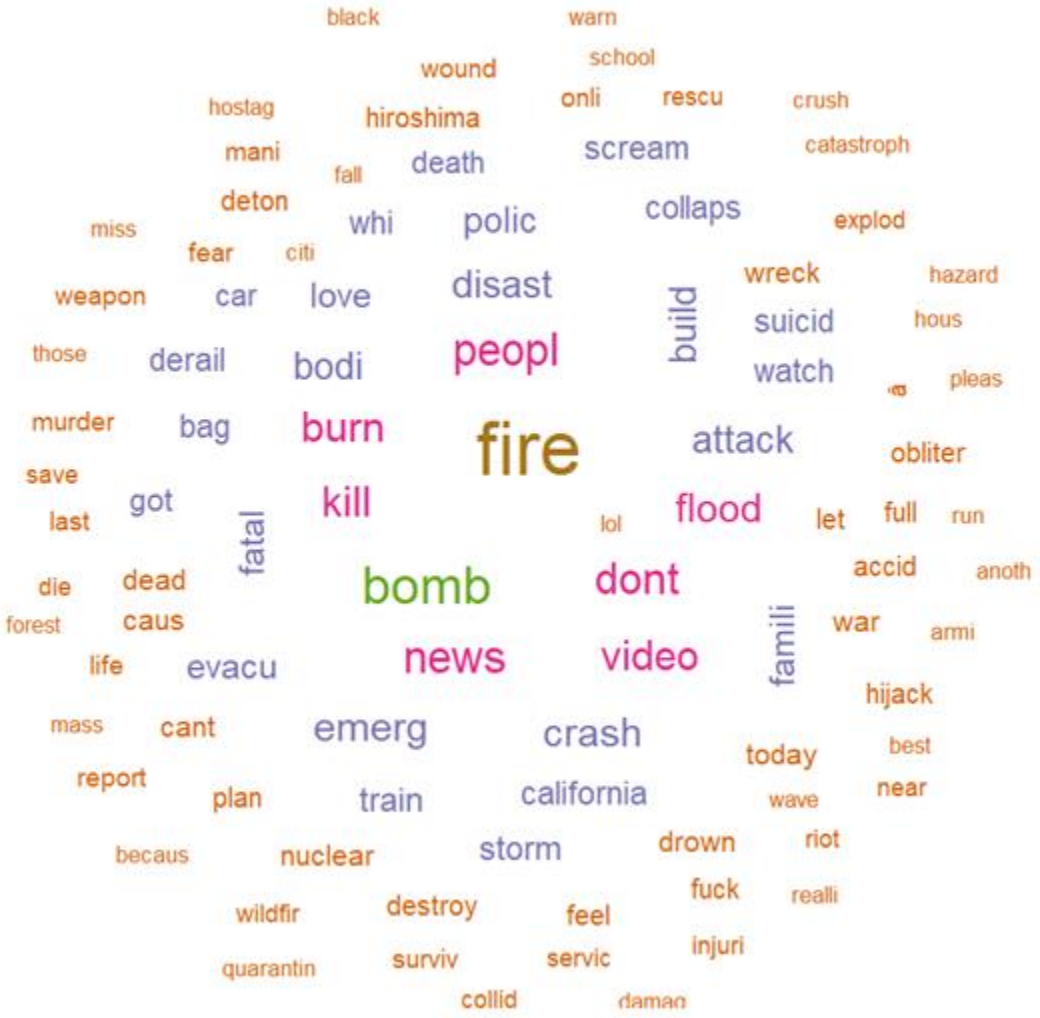
8

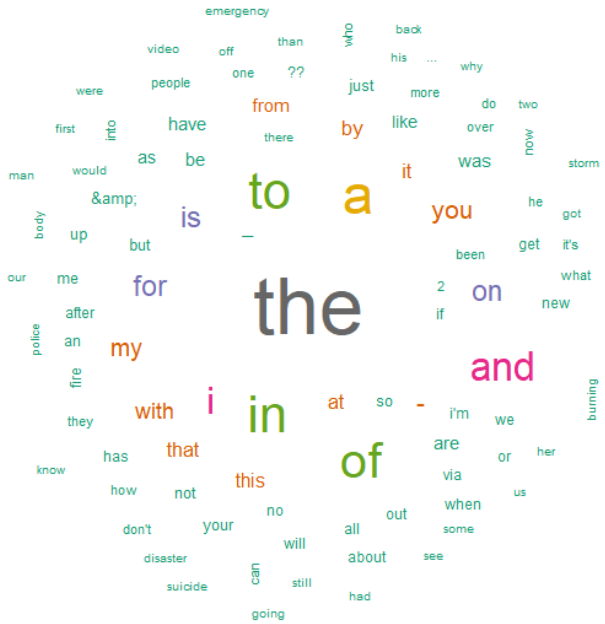
Stemming

```
# stemming słów z textów tweetów
all_data_corpus <- tm_map(
  all_data_corpus, stemDocument, "english")
```

Przekształcony tekst tweet-a







Chmura słów po procesowaniu

Budowanie modelu

Budowa DocumentTermMatrix

```
DTM <- DocumentTermMatrix(train_data_corpus)
inspect(DTM[1:10, 50:60])
```

```
Non-/sparse entries: 12/98
Sparsity           : 89%
Maximal term length: 7
Weighting          : term frequency (tf)
```

Docs	across	afraid	build	emerg	happen	hill	spring	street	top	wood
1	0	0	0	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1	1	0	0
8	0	0	0	0	0	1	0	0	1	1
9	1	0	1	1	1	0	0	1	0	0

Usunięcie rzadkich słów

```
sparse_DTM <- removeSparseTerms(DTM, 0.995)
tweetsSparse <- as.data.frame(as.matrix(sparse_DTM))
colnames(tweetsSparse) <- make.names(colnames(tweetsSparse))
tweetsSparse$target <- train_data$target
```

```
> inspect(sparse_DTM[1:5, 50:60])
<<DocumentTermMatrix (documents: 5, terms: 11)>>
Non-/sparse entries: 0/55
Sparsity           : 100%
Maximal term length: 5
Weighting          : term frequency (tf)
Sample             :
    Terms
Docs burn leav life everi next befor shot black train while truck
   1    0    0    0     0    0     0    0     0     0     0     0
   2    0    0    0     0    0     0    0     0     0     0     0
   3    0    0    0     0    0     0    0     0     0     0     0
   4    0    0    0     0    0     0    0     0     0     0     0
   5    0    0    0     0    0     0    0     0     0     0     0
```

Podział na zbiór treningowy i testowy

```
# nauka - podział na 70% danych treningowych i 30 testowych
smp_size <- floor(0.7 * nrow(tweetsSparse))
set.seed(123)
train_ind <- sample(seq_len(nrow(tweetsSparse)), size = smp_size)
train <- tweetsSparse[train_ind, ]
test <- tweetsSparse[-train_ind, ]
```

Drzewo decyzyjne

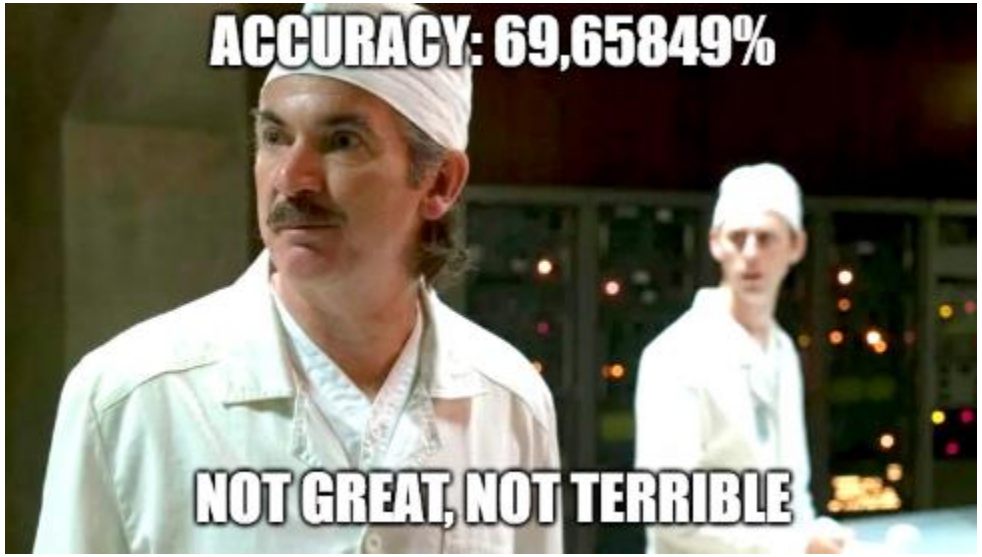
Trenowanie modelu + ewaluacja

```
# ternowanie modelu - drzewo decyzyjne
library('rpart')

tweetCART <- rpart(target ~ . , data = train, method = "class")

predictCART <- predict(tweetCART, newdata = test, type = "class")
cmat_CART <- table(test$target, predictCART)
cmat_CART
accu_CART <- (cmat_CART[1,1] + cmat_CART[2,2])/sum(cmat_CART)
accu_CART
```

```
   predictCART
      0    1
  0 1233  123
  1  570  358
> accu_CART <- (cmat_CART[1,1] + cmat_CART[2,2])/sum(cmat_CART)
> accu_CART
[1] 0.6965849
> tweet_dtm
```



SVM

Trenowanie modelu + ewaluacja

```
# trenowanie modelu - SVM

model_svm <- e1071::svm(target ~ .,
                        data = train, type = "nu-classification", kernal = "gaussian")


predictions_svm <- predict(model_svm, test)
truth_table <- table(test$target, predictions_svm)
truth_table

accu_svm <- (truth_table[1,1] + truth_table[2,2])/sum(truth_table)
accu_svm
```

```
> truth_table <- table(test$target, predictions_svm)
> truth_table
   predictions_svm
      0    1
  0 1164  192
  1  354  574
> accu_svm <- (truth_table[1,1] + truth_table[2,2])/sum(truth_table)
> accu_svm
[1] 0.7609457
>
```

Podsumowanie

Co zostało zrobione

- Analiza zbioru danych
- Procesowanie tekstu celem przygotowania do analizy
- Wizualizacja i analiza uzyskanych rezultatów po procesowaniu
- Trenowanie modelu i ewaluacja

Dalsze plany

- Próba poprawienia wyniku przewidywania przez model – użycie bardziej zaawansowanych algorytmów, np. xgboost
- Praca nad doborem hiperparametrów do trenowania – analiza porównawcza wyników
- Dalsza analiza z wykorzystaniem zbioru testowego celem ukończenia konkursu na stronie Kaggle