Report 6 - Naive Bayes and Text Mining

Heitor Gabriel S. Monteiro *CAEN/UFC*

Contents

Exercício Proposto.

Exercitar as funções de mineração de texto e comparar um Naive Bayes, com distribuição Bernoulli, de Laplace 0 e 1.
Usaremos os pacotes e fixaremos o diretório de trabalho:

```
setwd("/home/heitor/Documentos/Economia/Estatística e Econometria/Statistical Learning/Análise M
library(tidyverse)   # standard
library(tidymodels)  # standard
library(tm)          # text mining
library(SnowballC)   # to stemming words
library(wordcloud2)  # make words visualizations, the input have to be a data.frame
library(naivebayes)  # naive bayes procedures
library(gmodels)     # to tabulate results
library(knitr)
```

Importação e Tratamento

Importamos os dados:

```
dd <- read_csv("sms_spam.csv") %>% as_tibble()
dd$type <- as.factor(dd$type)
```

```
dd %>% glimpse() %>% summary()
```

```
## Rows: 5,559
## Columns: 2
## $ type <fct> ham, ham, ham, spam, spam, ham, ham, ham, spam, ham, ham, ham, ha~
## $ text <chr> "Hope you are having a good week. Just checking in", "K..give bac~
```

```
##    type          text
##  ham :4812   Length:5559
##  spam: 747   Class :character
##              Mode  :character
```

```
dd_corpus1 <- VCorpus(VectorSource(dd$text))
```

Faremos uma função de limpeza dos dados de texto e a aplicaremos sobre nosso *Corpus*, gerando um novo corpus, `dd_corpus2`. Na limpeza vamos 1) remover pontuação; 2)remover espaços em branco; 3)transformar qualquer caractere que ainda não esteja no padrão UTF-8; 4) remover números; 5) remover palavras que são meros conectivos; 6) padronizar tudo para minúsculos; 7) remover verbetes e abreviações em latim; 8) reduzir os vocábulos ao seu radical.

```
clean_corpus <- function(corpus_to_use){
  corpus_to_use %>%
    tm_map(removePunctuation) %>%
    tm_map(stripWhitespace) %>%
    tm_map(content_transformer(function(x) iconv(x, to='UTF-8', sub='byte'))) %>%
    tm_map(removeNumbers) %>%
    tm_map(removeWords, stopwords("en")) %>%
    tm_map(content_transformer(tolower)) %>%
    tm_map(removeWords, c("etc","ie", "eg", stopwords("english"))) %>%
    tm_map(stemDocument)
}
dd_corpus2 <- clean_corpus(dd_corpus1)
remove(dd_corpus1)
```

Tabela de Frequência e Visualização

Faremos, com os dados limpos, uma matriz com cada e-mail como observação, nas linhas, e as palavras como variáveis, nas colunas. Tal objeto é chamado de *DTM*. Após, contaremos a incidência de cada palavra em cada documento e somaremos as aparições totais.

```
find_freq_terms_fun <- function(corpus_in){
   dd_dtm        <- DocumentTermMatrix(corpus_in)
   freq_terms    <- findFreqTerms(dd_dtm)[1:max(dd_dtm$ncol)]
   terms_grouped <- dd_dtm[,freq_terms] %>%
     as.matrix() %>%
     colSums() %>%
     data.frame(Term=freq_terms, Frequency = .) %>%
     arrange(desc(Frequency)) %>%
     mutate(prop_term_to_total_terms=Frequency/nrow(.))
   return(data.frame(terms_grouped))
}
freq_terms_crp <- data.frame(find_freq_terms_fun(dd_corpus2))
head(freq_terms_crp, n=10)
```

```
##      Term Frequency prop_term_to_total_terms
## call call       656                0.09521045
## now   now       478                0.06937591
## get   get       449                0.06516691
## can   can       405                0.05878084
## will will       386                0.05602322
## just just       367                0.05326560
## come come       299                0.04339623
## dont dont       293                0.04252540
## free free       278                0.04034833
## know know       270                0.03918723
```

Agora, faremos a visualização da nuvem de palavras usadas. Antes, retiraremos as palavras com menor freqência, para não poluir a visualização.

```
wc1 <- wordcloud2(freq_terms_crp[,1:2] %>%
                    filter(freq_terms_crp$Frequency>35),
                  shape="circle",
                  color="random-light",
                  backgroundColor = "black")
```

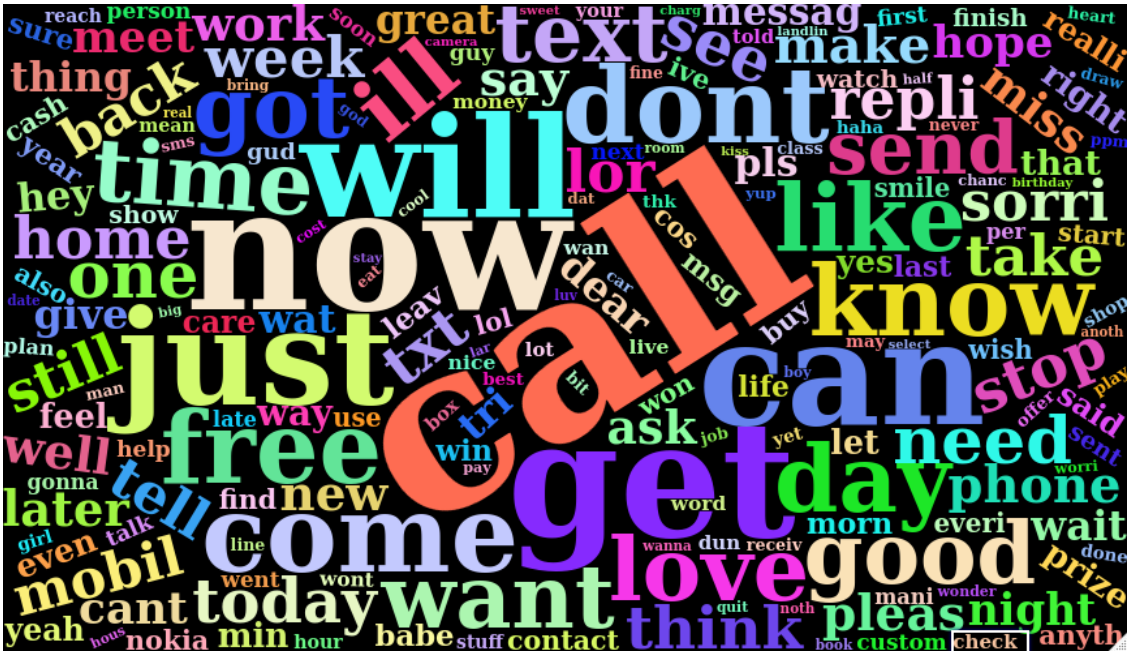

Figure 1: Word Cloud. Own elaboration.

Divisão entre Teste e Treino

```
dtm <- DocumentTermMatrix(dd_corpus2)
dtm_train <- dtm[1:4169, ]
dtm_test  <- dtm[4170:5559, ]
```

Repararemos se ambos têm a mesta proporção para os fatores:

```
train_type <- dd[1:4169, ]$type
test_type  <- dd[4170:5559, ]$type
prop.table(table(train_type))
```

```
## train_type
##       ham      spam
## 0.8647158 0.1352842
```

```
prop.table(table(train_type))
```

```
## train_type
##       ham      spam
## 0.8647158 0.1352842
```

Assim como retiramos as palavras infrequentes dos dados para a visualização, faremos o mesmo nas amostras de treino e teste:

```
freq_words <- findFreqTerms(dtm_train, 5)

dtm_train2 <- dtm_train [ , freq_words]
dtm_test2  <- dtm_test  [ , freq_words]

remove(dtm_train, dtm_test)
```

O modelo precisa de variáveis-fatores para rodar. Transformaremos a frequência de aparição dos termos em somente *sim* ou *não*, caso tenha ou não aparecido na mensagem, aplicando uma função que criaremos.

```
convert_counts <- function(x) {
  x <- ifelse(x > 0, "Yes", "No")
}

dd_train <- apply(dtm_train2, MARGIN = 2, convert_counts)
dd_test  <- apply(dtm_test2,  MARGIN = 2, convert_counts)
```

Aplicando o Modelo

```
nb1 <- naive_bayes(x= dd_train,
                   y= train_type,
                   laplace= 1)

tables(nb1, c('call', 'pay', 'free', 'now'))
```

```
##
## ------------------------------------------------------------------------------
##  ::: call (Bernoulli)
## ------------------------------------------------------------------------------
##
## call         ham       spam
##   No  0.94233435 0.55123675
##   Yes 0.05766565 0.44876325
##
## ------------------------------------------------------------------------------
##  ::: free (Bernoulli)
## ------------------------------------------------------------------------------
##
## free         ham       spam
##   No  0.98724702 0.76855124
##   Yes 0.01275298 0.23144876
##
## ------------------------------------------------------------------------------
##  ::: now (Bernoulli)
## ------------------------------------------------------------------------------
##
## now          ham       spam
##   No  0.94039368 0.75618375
##   Yes 0.05960632 0.24381625
##
## ------------------------------------------------------------------------------
##  ::: pay (Bernoulli)
## ------------------------------------------------------------------------------
##
## pay           ham        spam
##   No  0.993346271 0.994699647
##   Yes 0.006653729 0.005300353
##
## ------------------------------------------------------------------------------
```

```
tst1 <- predict(nb1, dd_test,
                type= 'class')

CrossTable(tst1, test_type,
           prop.chisq = FALSE,
           prop.t = T,
           prop.r = F,
           prop.c = F,
           dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1390
##
##
##              | actual
##    predicted |       ham |      spam | Row Total |
## -------------|-----------|-----------|-----------|
##          ham |      1202 |        28 |      1230 |
##              |     0.865 |     0.020 |           |
## -------------|-----------|-----------|-----------|
##         spam |         5 |       155 |       160 |
##              |     0.004 |     0.112 |           |
## -------------|-----------|-----------|-----------|
## Column Total |      1207 |       183 |      1390 |
## -------------|-----------|-----------|-----------|
##
##
```