Criação e Processamento de um Corpus

Prof. Walmes Zeviani

walmes@ufpr.br

Laboratório de Estatística e Geoinformação
Departamento de Estatística
Universidade Federal do Paraná

Justificativa e objetivos

Justificativa

- Em mineração de texto, o preprocessamento requer muito esforço/tempo.
- Bom preprocessamento contribui em muitro para o sucesso da análise.
- É importante conhecer as técnicas e sabem como aplicá-las.

Objetivos

- Descrever as principais etapas de processamento de texto.
- Criar e processar um Corpus.
- Descrever classes de objetos e métodos.
- Construir matriz de documentos e termos.
- Gerar algumas estatísticas e visualizações.

Etapas de preprocessamento

Etapas mais comuns

Padronização de caixa

- Elimina variações de escrita das palavras.
- {Caio, UFPR, USA, latticeExtra} $\rightarrow$ {caio, ufpr, usa, latticeextra}.

Remoção de pontuação

- Retira os sinais de pontuação.
- Ex: `.` `,` `:` `-` `/` `!` `*` `(` `)` `[` `]` `{` `}` `+` `~` `$` `&`, etc.
- {vende-se, olá!, o que?} $\rightarrow$ {vende se, olá, o que}

Etapas mais comuns

Elimina espaços

- Apara os espaços em branco nas extremidades.
- Elimina o excesso de espaços entre palavras.
- Substitui tabulações e similares por espaços.

Remoção dos números

- Elimina caracteres numéricos: 0 a 9.
- Ex: Bebeu 2 xícaras de café às 14h00 por R$ 4,50. → Bebeu xícaras de café às h por R$,.

Etapas mais comuns

Remoção das *stopwords*

- Elimina palavras que são muito frequentes.
- São palavras de ligação e sem contexto específico.
- Principalmente: artigos, preposições, conjunções e pronomes.
- Ex: o, a, os, as, com, sem, aos, mas, portanto, eu, ela, nós.
- E também verbos predominantes: ser, ter, haver.
- https://gist.github.com/alopes/5358189.

Etapas mais comuns

Desbaste de afixos de extremidade

- Elimina prefixos e sufixos.
- Reduz variações de gênero, número, tempo, etc.
- A tranform**ação** do materia**l** em produ**to** fina**l** pela indústria. → A transform do materia em produ fin pela indústria.

Remoção de caracteres acentuados

- Caracteres acentuados são comuns em português.
- Ex: ç ã â à á é ê í ó õ ú.
- A refeição foi macarrão à carbonara. → A refeicao foi macarrao a carbonara.

O que as etapas tem em comum?

- Diminuir o número de variações gráficas.
- Eliminar termos comuns.

Preprocessamento com funções do R

```
x <- "
Minha terra tem palmeiras,
Onde canta o     Sabiá;
As aves, que aqui gorjeiam,
Não    gorjeiam como lá.
Nosso céu tem mais estrelas,
Nossas    várzeas têm mais flores,
Nossos bosques    têm mais vida,
Nossa vida mais amores."

library(tm)

tolower(x)
removePunctuation(x)
stripWhitespace(x)
removeNumbers(x)
removeWords(x, words = c("tem", "o", "a", "mais", "onde"))
stemDocument(x, language = "portuguese")
```

Criando e processando o corpus

Carregando o texto

```
library(tm)
library(tidyverse)
library(tidytext)

# Carregar uma lista de listas chamada `ufpr` com notícias envolvendo a
# UFPR entre 2016-09-05 até 2017-03-31.
load(file = "../data/ufpr-news.RData")

class(ufpr)      # Tipo de objeto.
```

```
## [1] "list"
```

```
length(ufpr)     # Tamanho da lista.
```

```
## [1] 8305
```

```
names(ufpr[[1]]) # Conteúdo.
```

```
##  [1] "id_noticia"       "str_midia"        "str_veiculo"
##  [4] "str_tipo_veiculo" "str_cidade"       "str_estado"
##  [7] "str_pais"         "str_secao"        "ts_publicacao"
## [10] "str_titulo"       "conteudo_texto"
```

Criando o corpus

```
#-----------------------------------------------------------------
# Criando um Corpus.

# Pegar as primeiras notícias.
tb <- ufpr[1:200] %>%
    map(tibble) %>%
    flatten_df()

# Cria um corpus.
cps <- VCorpus(VectorSource(x = tb$str_titulo),
               readerControl = list(language = "pt",
                                    load = TRUE))
cps
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 200
```

```r
# Classe e métodos para a classe.
class(cps)
```

```
## [1] "VCorpus" "Corpus"
```

```r
methods(class = "VCorpus")
```

```
##  [1] as.list            as.VCorpus         content
##  [4] c                  format             inspect
##  [7] length             meta<-             meta
## [10] names<-            names              print
## [13] TermDocumentMatrix tm_filter          tm_index
## [16] tm_map             [                  [[<-
## [19] [[
## see '?methods' for accessing help and source code
```

```r
# Classe e métodos para a classe.
class(cps[[1]])
```

```
## [1] "PlainTextDocument" "TextDocument"
```

```r
methods(class = "PlainTextDocument")
```

```
##  [1] as.character      content<-         content
##  [4] format            meta<-            meta
##  [7] print             removeNumbers     removePunctuation
## [10] removeWords       stemDocument      stripWhitespace
## [13] tm_term_score     words
## see '?methods' for accessing help and source code
```

```
# Conteúdo do elemento.
content(cps[[1]])
```

```
## [1] "MST ergue acampamento em frente ao Incra de Curitiba para exigir refor
```

```
# Meta dados do elemento.
meta(cps[[1]])
```

```
##   author       : character(0)
##   datetimestamp: 2019-04-03 01:09:46
##   description  : character(0)
##   heading      : character(0)
##   id           : 1
##   language     : pt
##   origin       : character(0)
```

Aplicação das etapas de preprocessamento

Caixa, pontuação, números e espaços

```
cps <- tm_map(cps, FUN = content_transformer(tolower))
cps <- tm_map(cps, FUN = removePunctuation)
cps <- tm_map(cps, FUN = removeNumbers)
cps <- tm_map(cps, FUN = stripWhitespace)
sapply(cps[1:4], content)
```

```
##
## "mst ergue acampamento em frente ao incra de curitiba para exigir reforma a
##
##                                                  "pg perde a urbanista silvia
##
##                                        "vestibular ufpr está com as inscrições a
##
##           " amigos do hc busca voluntários para atuar no maior hospital do
```

Stopwords

```
# Remove as stopwords.
head(stopwords("portuguese"))
```

```
## [1] "de"  "a"   "o"   "que" "e"   "do"
```

```
cps <- tm_map(cps, FUN = removeWords, words = stopwords("portuguese"))
sapply(cps[1:4], content)
```

```
##                                                                    1
## "mst ergue acampamento  frente  incra  curitiba  exigir reforma agrária"
##                                                                    2
##                                      "pg perde  urbanista silvia contin"
##                                                                    3
##                                   "vestibular ufpr    inscrições abertas"
##                                                                    4
##           " amigos  hc busca voluntários  atuar  maior hospital  paraná"
```

Desbaste

```
cps2 <- tm_map(cps, FUN = stemDocument, language = "portuguese")
sapply(cps2[1:4], content)
```

```
##                                                                      1
## "mst ergu acampamento frent incra curitiba exigir reforma agrária"
##                                                                      2
##                                  "pg perd urbanista silvia contin"
##                                                                      3
##                                  "vestibular ufpr inscriçõ aberta"
##                                                                      4
##              "amigo hc busca voluntário atuar maior hospit paraná"
```

Acentuação

```
acen <- function(x) iconv(x, to = "ASCII//TRANSLIT")
cps2 <- tm_map(cps2, FUN = content_transformer(acen))
sapply(cps2[1:4], content)
```

```
##                                                                      1
## "mst ergu acampamento frent incra curitiba exigir reforma agraria"
##                                                                      2
##                                    "pg perd urbanista silvia contin"
##                                                                      3
##                                    "vestibular ufpr inscrico aberta"
##                                                                      4
##              "amigo hc busca voluntario atuar maior hospit parana"
```

Metadados do corpus

O que são os metadados

- Serve para aplicação de filtros, ordenação, etc.
- São informações estruturadas sobre cada documento.
- Comum ter: estampa de tempo, local, autor.
- É uma tabela em que cada linha está vínculada a um documento do corpus.

Tipo de metadados

Indexação global

```
# Com "indexed" (default) indica que ficara como tabela mas mantendo
# indexação com os elementos. É melhor para desempenho.
meta(cps2, tag = "ts", type = "indexed") <- tb$ts_publicacao
meta(cps2, tag = "cidade") <- tb$str_cidade
meta(cps2[[2]])
```

```
##   author       : character(0)
##   datetimestamp: 2019-04-03 01:09:46
##   description  : character(0)
##   heading      : character(0)
##   id           : 2
##   language     : pt
##   origin       : character(0)
```

```
head(meta(cps2), n = 2)
```

```
##                    ts       cidade
## 1 2016-09-05 00:14:18     Curitiba
## 2 2016-09-05 02:18:35 Ponta Grossa
```

Tipo de metadados

Indexação local

```
# Com "local" indica que ficará dentro de cada elemento.
meta(cps2, tag = "ts", type = "local") <- tb$ts_publicacao
meta(cps2[[2]])
```

```
##   author       : character(0)
##   datetimestamp: 2019-04-03 01:09:46
##   description  : character(0)
##   heading      : character(0)
##   id           : 2
##   language     : pt
##   origin       : character(0)
##   ts           : 2016-09-05 02:18:35
```

Matriz de termos e documentos

```
# Matriz de documentos e termos.
dtm <- DocumentTermMatrix(cps2) # Documentos nas linhas.
tdm <- TermDocumentMatrix(cps2) # Termos nas linhas.

# Detalhes.
# tdm
dtm
```

```
## <<DocumentTermMatrix (documents: 200, terms: 524)>>
## Non-/sparse entries: 1264/103536
## Sparsity           : 99%
## Maximal term length: 25
## Weighting          : term frequency (tf)
```

```
# Número de documentos e termos (vocabulário).
# dim(dtm)
# dim(tdm)
c(nTerms(tdm), nDocs(tdm))
```

```
## [1] 524 200
```

Classe e métodos

```
# Classe e métodos.
class(dtm)
```

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
# methods(class = "DocumentTermMatrix")
methods(class = "TermDocumentMatrix")
```

```
##  [1] as.DocumentTermMatrix as.TermDocumentMatrix
##  [3] c                     dimnames<-
##  [5] Docs                  findAssocs
##  [7] findMostFreqTerms     inspect
##  [9] nDocs                 nTerms
## [11] plot                  print
## [13] [                     Terms
## [15] tidy                  tm_term_score
## [17] t
## see '?methods' for accessing help and source code
```

Inspeção da DTM

```
# Algumas linhas e colunas da matriz.
inspect(tdm)
```

```
## <<TermDocumentMatrix (terms: 524, documents: 200)>>
## Non-/sparse entries: 1264/103536
## Sparsity           : 99%
## Maximal term length: 25
## Weighting          : term frequency (tf)
## Sample             :
##             Docs
## Terms        13 154 155 16 17 37 38 41 42 61
##   aluno       0   1   1  0  0  0  0  0  0  0
##   concurso    0   0   0  0  0  0  0  0  0  1
##   curitiba    0   0   0  0  0  1  1  0  0  0
##   dia         0   0   0  0  0  0  0  0  0  0
##   isencao     0   1   1  0  0  0  0  0  0  0
##   mil         0   1   1  0  0  0  0  0  0  1
##   pedir       0   1   1  0  0  0  0  0  0  0
##   podem       0   1   1  0  0  0  0  0  0  0
##   ufpr        0   0   0  0  0  1  1  1  1  0
##   vestibular  0   1   1  0  0  0  0  0  0  0
```

A esparsidade da DTM

```r
u <- VCorpus(VectorSource(c("O céu é azul",
                            "A mata é verde",
                            "Mas a rosa é vermelha")))
d <- DocumentTermMatrix(u)
inspect(d)
```

```
## <<DocumentTermMatrix (documents: 3, terms: 7)>>
## Non-/sparse entries: 7/14
## Sparsity           : 67%
## Maximal term length: 8
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs azul céu mas mata rosa verde vermelha
##    1    1   1   0    0    0     0        0
##    2    0   0   0    1    0     1        0
##    3    0   0   1    0    1     0        1
```

```r
# Esparsidade (proporção de cédulas com 0 na matriz).
non_zero <- sum(d > 0)
cells <- prod(dim(d))
100 * (1 - (non_zero/cells))
```

```
## [1] 66.66667
```

Visualizar a matriz de termos e documentos

```r
library(lattice)

# Converte para matriz ordinária.
m <- as.matrix(tdm)

# Reordenar matriz por frequencia dos termos.
m <- m[order(apply(m, MARGIN = 1, sum), decreasing = TRUE), ]
m <- t(m)

# Visualiza a matriz de documentos e termos. Cuidado com as dimensões.
levelplot(m,
          xlab = "Documentos",
          ylab = "Termos",
          col.regions = gray.colors,
          scales = list(x = list(rot = 90))) +
    latticeExtra::layer(panel.abline(h = 2:ncol(m) - 0.5,
                        v = 2:nrow(m) - 0.5,
                        col = "white"))
```

Termos mais frequentes

```
# Termos com frequencia superior a um valor de corte.
mft <- findFreqTerms(tdm, lowfreq = 10)
mft
```

```
##  [1] "aluno"       "concurso"    "curitiba"    "dia"
##  [5] "estadu"      "isencao"     "mil"         "pedir"
##  [9] "podem"       "publico"     "taxa"        "ufpr"
## [13] "unesp"       "universidad" "vestibular"
```

```
# Frequência dos termos.
trm <- as.matrix(dtm[, c("aluno", "isencao", "vestibular")])
colSums(trm > 0) # Frequência.
```

```
##      aluno    isencao vestibular
##         11         11         18
```

Termos associados

```
# Termos associados por um valor acima de um limite.
# findAssocs(dtm, terms = mft, corlimit = 0.5)
ass <- findAssocs(dtm, terms = "aluno", corlimit = 0.5)
ass
```

```
## $aluno
##      pedir      podem       taxa      unesp    isencao vestibular
##       1.00       1.00       1.00       1.00       0.81       0.77
##       rede       vale
##       0.60       0.51
```

```
# Correlação de Pearson.
cor(trm)
```

```
##                aluno   isencao vestibular
## aluno      1.0000000 0.8075998  0.7671228
## isencao    0.8075998 1.0000000  0.6904871
## vestibular 0.7671228 0.6904871  1.0000000
```

Cálculo da esparsidade

```r
# slam: Sparse Lightweight Arrays and Matrices
u <- sort(slam::col_sums(dtm > 0), decreasing = TRUE)

# Esparsidade da matrix.
1 - sum(u)/prod(dim(dtm))
```

```
## [1] 0.9879389
```

```r
# Espasidade de cada termo.
tsp <- 1 - u/nDocs(dtm)
sum(tsp < 0.95) # Quantos tem menos que o corte.
```

```
## [1] 13
```

```r
# Esparsidade da DTM com acúmulo dos termos.
sps <- 1 - cumsum(u)/(nDocs(dtm) * seq_along(u))
```

Remoção de esparsidade

```
# Remove os termos mais esparsos.
rst <- removeSparseTerms(dtm, sparse = 0.95)
# nTerms(rst)
Terms(rst)
```

```
##  [1] "aluno"      "concurso"   "curitiba"   "dia"
##  [5] "isencao"    "mil"        "pedir"      "podem"
##  [9] "publico"    "taxa"       "ufpr"       "unesp"
## [13] "vestibular"
```

Núvem de palavras

```r
library(wordcloud)
library(RColorBrewer)

d <- data.frame(word = names(u),
                freq = u)

wordcloud(words = d$word,
          freq = d$freq,
          min.freq = 1,
          max.words = 200,
          random.order = FALSE,
          colors = brewer.pal(8, "Dark2"))
```


protesto
traducao
prolongado
ganha
alternativa
alguma
wilson
aponta
naturalizacao
medicina
resiliencia
pesquisadora
descarbonizacao
tartaruga
indigena
gustavo
educacao
semana
xvi
reciclagem
caravana
democratico
italiano
paulo
especializacao
expobai
alckmin
polici
lamenta
apo
mantem
coluna
campanha
negra
bayer
publica
perfi
mundial
capsula
falecimento
pagam
governo
positivo
realiza
transito
encontrada
palestra
escola
paranista
gluten
universidad
carta
sobem
intermod
querer
ginastica
unesp
curitiba
seminario
ordem
equilibrio
galvao
brasileira
vale
livro
todo
feriado
geraldo
saber
literaria
cidad
isencao
aluno
rank
eleitor
definir
psicopata
agenda
taxa
capela
biscoito
crime
abr
melhor
greca
pedido
estadu
dia
mil
feira
bebida
curso
estudo
agricola
debat
uel
estado
fruet
ministerio
prolik
hospit
pedir
ufpr
publico
violencia
japone
nacion
contin
rio
ano
vai
ser
universitario
nesta
morta
sesc
sul
cultura
briga
contra
consumidor
pode
economia
dinheiro
rede
vestibular
obra
cnasenar
advogado
inedita
concurso
simposio
sete
maratona
brasil
confira
'escondem'
processo
clinica
direito
podem
literatura
mercado
projeto
quai
sobr
sorteio
perfil
aberta
servidor
vaga
tadeu
violencia"
primeira
salario
povo
litor
parana
social
palotina
recheado
transforma
setembro
candidato
silvia
ideb
desafio
hackathon
internacion
encontro
resultado
param
lugar
avio
influencia
comeca
receb
prefeitura
parceiro
climatica
prr
roteiro
programacao
sistema
ervam
metodo
potenci
desenvolvesc
aerodesign
farinha
partido
utfpr
sextafeira
estadoleviata
tributaria
veneri
transformacao
constroem

Próxima aula

Próxima aula

- Criação da matriz de documentos e termos.
- Formas de ponderação dos termos nos documentos.
- Formas para representar a distribuição dos termos no corpus.
- Mais sobre associação entre termos.