Mineração de Texto

Matriz de documentos e termos

Prof. Walmes Marques Zeviani

JUSTIÇA 4.0: INOVAÇÃO E EFETIVIDADE NA REALIZAÇÃO DA JUSTIÇA PARA TODOS
PROJETO DE EXECUÇÃO NACIONAL BRA/20/015

Matriz de Documentos e Termos

Motivação

- A **representação espaço vetor** é utilizada por muitas técnicas em mineração de texto.
- As coordenadas dos documentos dependem das **métricas** usadas para construção da matriz de documentos e termos.
- É preciso conhecer métricas mais adotadas e suas propriedades.
- A construção da matriz de documentos e termos deve considerar aspectos do problema em mãos.

Ponderação da matriz

- Expressa a **ocorrência** dos termos dentro de um documento.
- Funções da quantidade de vezes que o termo aparece:
  - Linear ou frequência absoluta:

$$n(t) = \text{count}(t, d), \quad n \in \mathbb{N}.$$

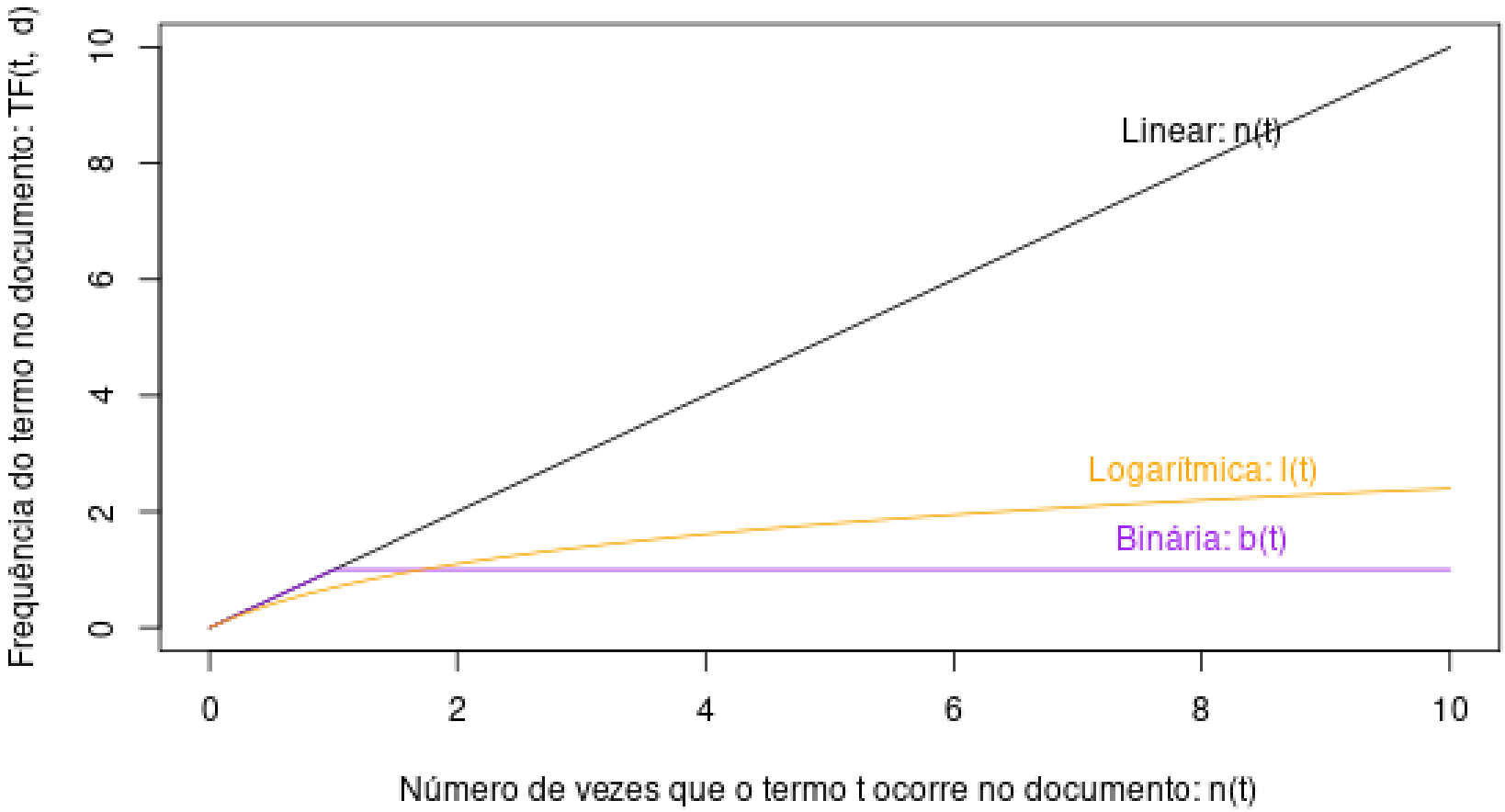
  - Indicadora ou binária:

$$b(t) = I(n(t) > 0), \quad b \in 0, 1.$$

  - Logarítmica:

$$l(t) = \log(n(t) + 1), \quad l \in \mathbb{R}.$$

  - Outras funções sublineares.
- Funções sublinares diminuem o efeito das palavras mais frequentes.



Funções de ponderação conforme a ocorrência de um termo no documento.

Poderar mais os termos raros

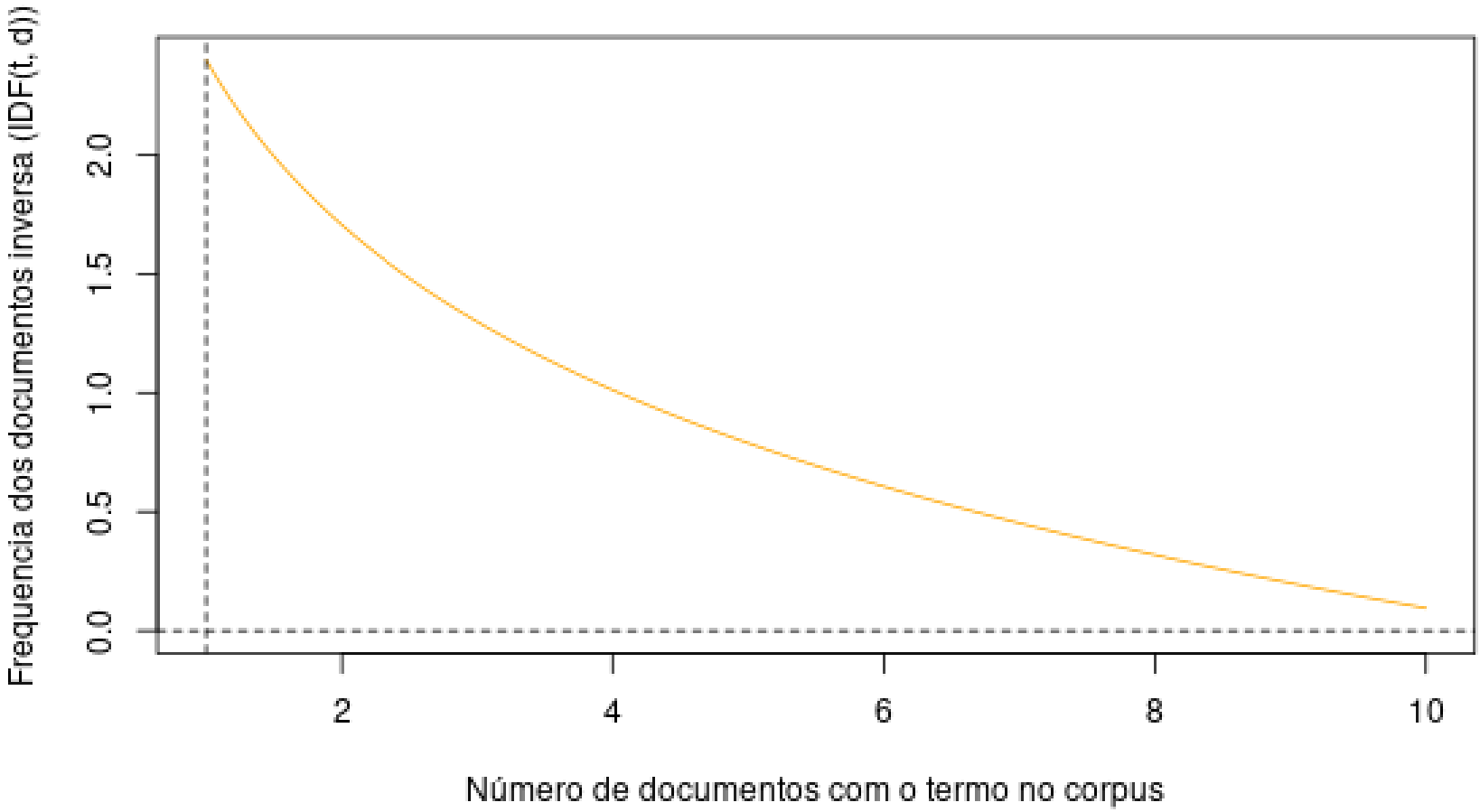
- Uma palavra que **ocorre em todos** os documento diz muito pouco sobre cada um deles.
- É uma característica que todos os documentos tem, então:
  - Não contribui para classificação/agrupamento dos documentos.
  - Não contribui para como uma variávei regressora.
- Usa-se dar mais peso para palavras no corpus que ocorrem em poucos documentos.

A ponderação TF-IDF

- TF-IDF: *term frequency inverse document frequency*.
- A porção IDF é determinada por

$$\mathrm{IDF}(t) = \log\left(\frac{\mathrm{count}(d, C) + 1}{\mathrm{count}(d : t \in d, C)}\right), \quad 0 \leq \mathrm{IDF} \leq \log(\mathrm{count}(d, C) + 1).$$

- Em algumas referências pode não aparecer o $+1$ no numerador.
- Numerador: total de documentos.
- Denominador: documentos que possuem o termo t.
- A razão é sempre positiva.



Ponderação IDF considerando um corpus com 10 documentos.

Ponderação de cada termo por TF-IDF

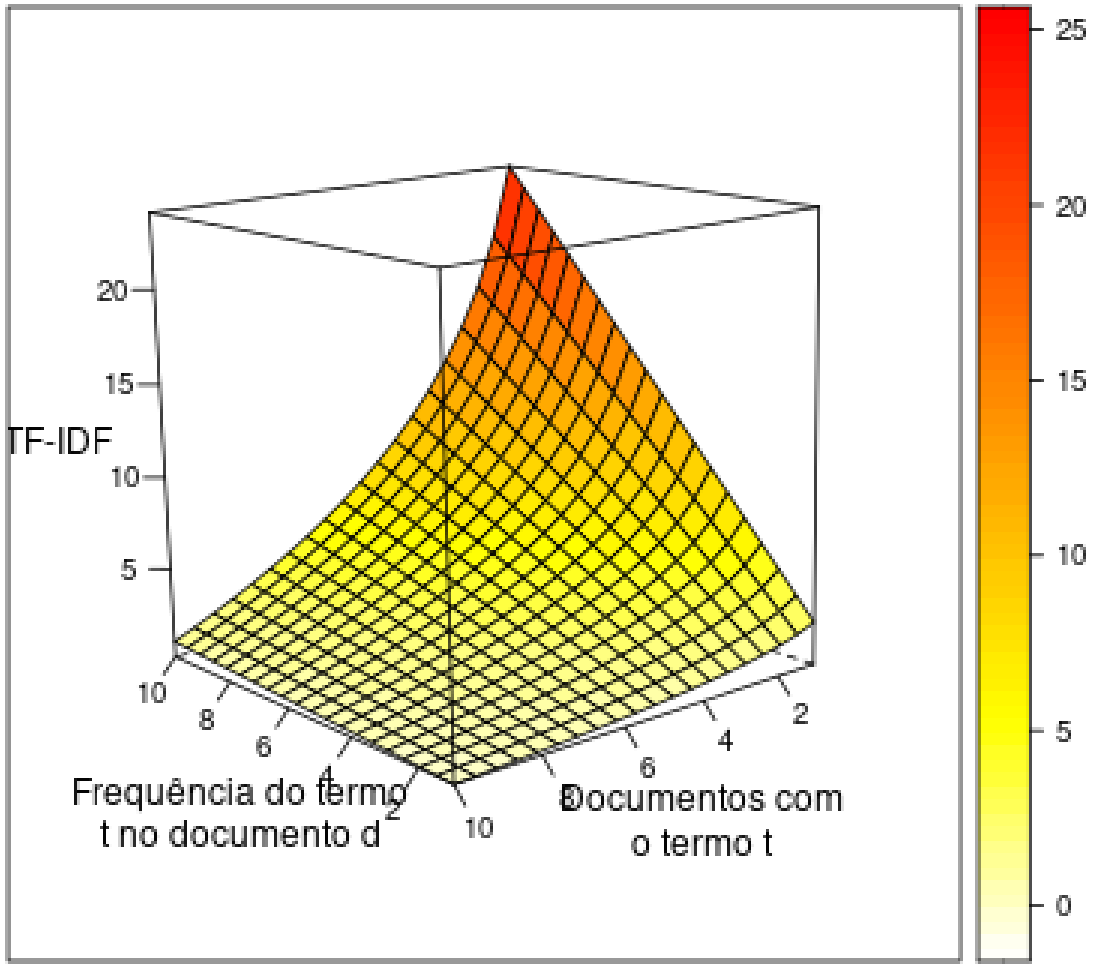
- Combina a frequência no termo (TF) no documento com a ocorrência dele na coleção (IDF).
- O peso de um termo no documento é

$$w(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t).$$

- Considerando explicitamente todos os termos, tem-se

$$w(t,d) = \mathrm{count}(t,d) \times \log\left(\frac{\mathrm{count}(d,C)+1}{\mathrm{count}(d : t \in d, C)}\right).$$

```
## Loading required package: lattice
```



Aplicação com o R

Um corpus didático

```r
library(tm)

docs <- c("A vida é linda.",
          "A vida é uma aventura.",
          "A vida é uma só.",
          "A vida é linda por ser única.",
          "A vida é minha, minha vida!",
          "A vida é pra ser vivida.")
cps <- VCorpus(VectorSource(x = docs),
               readerControl = list(language = "pt", load = TRUE))
# cps

cps <- tm_map(cps, FUN = content_transformer(tolower))
cps <- tm_map(cps, FUN = removePunctuation)
cps <- tm_map(cps, FUN = removeWords,
              words = c("a", "é", "e", "uma", "pra", "por", "ser"))
# content(cps[[1]])

# Funções de ponderação.
apropos("^weight[[:upper:]]", ignore.case = FALSE)
```

```
## [1] "weightBin"   "weightSMART" "weightTf"    "weightTfIdf"
```

Ponderação TF

```
dtm_tf <- DocumentTermMatrix(cps,
                             control = list(weighting = weightTf,
                                            wordLengths = c(1, Inf)))
inspect(dtm_tf)
```

```
## <<DocumentTermMatrix (documents: 6, terms: 7)>>
## Non-/sparse entries: 13/29
## Sparsity           : 69%
## Maximal term length: 8
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs aventura linda minha só única vida vivida
##    1        0     1     0  0     0    1      0
##    2        1     0     0  0     0    1      0
##    3        0     0     0  1     0    1      0
##    4        0     1     0  0     1    1      0
##    5        0     0     2  0     0    2      0
##    6        0     0     0  0     0    1      1
```

Ponderação binária (hot encoding)

```
dtm_bin <- DocumentTermMatrix(cps,
                              control = list(weighting = weightBin,
                                             wordLengths = c(1, Inf)))
inspect(dtm_bin)
```

```
## <<DocumentTermMatrix (documents: 6, terms: 7)>>
## Non-/sparse entries: 13/29
## Sparsity           : 69%
## Maximal term length: 8
## Weighting          : binary (bin)
## Sample             :
##     Terms
## Docs aventura linda minha só única vida vivida
##    1        0     1     0  0     0    1      0
##    2        1     0     0  0     0    1      0
##    3        0     0     0  1     0    1      0
##    4        0     1     0  0     1    1      0
##    5        0     0     1  0     0    1      0
##    6        0     0     0  0     0    1      1
```

```
dtm_bin <- 1 * as.matrix(dtm_tf > 0)
# dtm_bin
```

Ponderação TF-IDF (fazendo manualmente)

```
dtm_tf <- as.matrix(dtm_tf)
dtm_bin <- 1 * (dtm_tf > 0)

# ATTENTION: é log base 2 que é usada. Não tem o `+1`.
(idf <- log2((nrow(dtm_bin))/colSums(dtm_bin)))
```

```
## aventura    linda    minha       só    única     vida   vivida
## 2.584963 1.584963 2.584963 2.584963 2.584963 0.000000 2.584963
```

```
# Divide cada coluna pelo respectivo escalar.
sweep(dtm_tf, MARGIN = 2, STATS = idf, FUN = "*")
```

```
##     Terms
## Docs aventura    linda    minha       só    única vida   vivida
##    1 0.000000 1.584963 0.000000 0.000000 0.000000    0 0.000000
##    2 2.584963 0.000000 0.000000 0.000000 0.000000    0 0.000000
##    3 0.000000 0.000000 0.000000 2.584963 0.000000    0 0.000000
##    4 0.000000 1.584963 0.000000 0.000000 2.584963    0 0.000000
##    5 0.000000 0.000000 5.169925 0.000000 0.000000    0 0.000000
##    6 0.000000 0.000000 0.000000 0.000000 0.000000    0 2.584963
```

Ponderação TF-IDF

```r
# https://www.quora.com/What-are-non-normalized-TF-IDF-weights
weightTfIdf_un <- function(x) weightTfIdf(x, normalize = FALSE)

dtm_tfidf <- DocumentTermMatrix(cps,
                                control = list(
                                    weighting = weightTfIdf_un,
                                    wordLengths = c(1, Inf)))
inspect(dtm_tfidf)
```

```
## <<DocumentTermMatrix (documents: 6, terms: 7)>>
## Non-/sparse entries: 7/35
## Sparsity           : 83%
## Maximal term length: 8
## Weighting          : term frequency - inverse document frequency (tf-idf)
## Sample             :
##     Terms
## Docs aventura    linda    minha       só    única vida   vivida
##    1 0.000000 1.584963 0.000000 0.000000 0.000000    0 0.000000
##    2 2.584963 0.000000 0.000000 0.000000 0.000000    0 0.000000
##    3 0.000000 0.000000 0.000000 2.584963 0.000000    0 0.000000
##    4 0.000000 1.584963 0.000000 0.000000 2.584963    0 0.000000
##    5 0.000000 0.000000 5.169925 0.000000 0.000000    0 0.000000
##    6 0.000000 0.000000 0.000000 0.000000 0.000000    0 2.584963
```

Considerações sobre o TF-IDF

- Funciona como uma padronização multiplicativa por coluna.
- As padronizações Z ou min-max (são aditivas-multiplicativas).
- Termos raros serão valorizados e termos frequentes penalizados.
- O argumento `normalize = TRUE` faz uma normalização por linhas, ou seja, considera a proporção do documento com cada termo.

TF-IDF com normalização por linha (fito manualmente)

```
dtm_tf <- as.matrix(dtm_tf)
dtm_bin <- 1 * (dtm_tf > 0)

# Normalização por linha (proporção de cada termo no documento).
dlen <- rowSums(dtm_tf)
dtm_tfn <- sweep(dtm_tf, MARGIN = 1, STATS = dlen, FUN = "/")

# Normalização por coluna com IDF.
idf <- log2((nrow(dtm_bin))/colSums(dtm_bin))
sweep(dtm_tfn, MARGIN = 2, STATS = idf, FUN = "*")
```

```
##     Terms
## Docs aventura     linda    minha       só     única vida   vivida
##    1 0.000000 0.7924813 0.000000 0.000000 0.0000000    0 0.000000
##    2 1.292481 0.0000000 0.000000 0.000000 0.0000000    0 0.000000
##    3 0.000000 0.0000000 0.000000 1.292481 0.0000000    0 0.000000
##    4 0.000000 0.5283208 0.000000 0.000000 0.8616542    0 0.000000
##    5 0.000000 0.0000000 1.292481 0.000000 0.0000000    0 0.000000
##    6 0.000000 0.0000000 0.000000 0.000000 0.0000000    0 1.292481
```

TF-IDF com normalização por linha

```
dtm_tfidf <- DocumentTermMatrix(cps,
                                control = list(
                                    weighting = weightTfIdf,
                                    wordLengths = c(1, Inf)))
inspect(dtm_tfidf)
```

```
## <<DocumentTermMatrix (documents: 6, terms: 7)>>
## Non-/sparse entries: 7/35
## Sparsity           : 83%
## Maximal term length: 8
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
## Sample             :
##     Terms
## Docs aventura     linda    minha       só     única vida   vivida
##    1 0.000000 0.7924813 0.000000 0.000000 0.0000000    0 0.000000
##    2 1.292481 0.0000000 0.000000 0.000000 0.0000000    0 0.000000
##    3 0.000000 0.0000000 0.000000 1.292481 0.0000000    0 0.000000
##    4 0.000000 0.5283208 0.000000 0.000000 0.8616542    0 0.000000
##    5 0.000000 0.0000000 1.292481 0.000000 0.0000000    0 0.000000
##    6 0.000000 0.0000000 0.000000 0.000000 0.0000000    0 1.292481
```

Ponderações definidas pelo usuário

- É possível usar outras funções de ponderação.
- Verifique este link: https://stackoverflow.com/questions/39448360/how-do-i-set-up-tf-weight-of-terms-in-corpus-using-the-tm-package-in-r.
- A escolha da ponderação **é problema dependente** e é parte da fase de engenharia de características.
- Quando não houver clara preferência por uma ponderação, aplique as disponíveis e avalie os resultados.

Mineração de Texto

Análise de Sentimentos

Prof. Walmes Marques Zeviani

JUSTIÇA 4.0: INOVAÇÃO E EFETIVIDADE NA REALIZAÇÃO DA JUSTIÇA PARA TODOS
PROJETO DE EXECUÇÃO NACIONAL BRA/20/015

Análise de sentimentos

Justificativa e objetivos

- Mineração de opinião e análise de sentimentos são as técnicas mais usadas de mineração de texto em rede social.
- Dar uma ideia geral e comentar as limitações.
- Apresentar os conjuntos léxicos de sentimentos.
- Fazer uma aplicação simples.

Mineração da opinião

- Opiniões e pontos de vista.
  - E.g. classficação de posição política.
  - Podem haver várias classes.
  - Requer algoritmo de classificação e conjunto para seu treinamento.
  - Dados rotulados nem sempre são acessíveis.
- Análise de sentimento.
  - Determinar a polaridade (+1, 0, -1).
  - Pode ser resolvido com um dicionário de termos classificados.
  - Está associado com extração de informação.

Escopo da análise

- Nível de documento.
- Nível de sentença.
- Nível de palavra.
  - Análise de sentimento.
  - Soma algébrica das polaridades dos termos.
- Problemas com tratamento da negação:
  - Negação pré-verbal.
  - Negação pós-verbal.
  - Negação pré e pós-verbal.
- Problemas com variações imprevisiveis:
  - *Tony Stark? Amoooo!*.
  - *Estou com sdd de vc. Volta logo. ;-| **.

Bases de léxicos de sentimentos para português

- SentiLex-PT 02;
  - Contém 7.014 lemas e 82.347 formas flexionadas.
  - Abrange adjetivos, nomes, verbos e expressões.
  - Possui a polaridade e categoria gramatical de cada palavra.
- OpLexicon 3.0.
  - Contém 32.191 palavras polarizadas e classificadas gramaticalmente.
  - Inclui emoticons e hashtags.

Aplicando análise de sentimentos

Importação do dicionário

```
# Carregando o dicionário léxico de sentimentos.

library(lexiconPT)
# ls("package:lexiconPT")
tail(oplexicon_v3.0, n = 3)
```

```
##         term type polarity polarity_revision
## 32189  zunir   vb        1                 A
## 32190  zupar   vb       -1                 A
## 32191 zurzir   vb       -1                 A
```

```
xtabs(~type + polarity, data = oplexicon_v3.0)
```

```
##                polarity
## type              -1     0     1
##   adj           12825  6226  5424
##   emot             45     9    12
##   htag            314     6   151
##   vb             1297  2761  2831
##   vb adj           15     0    59
##   vb adv            4     0    18
##   vb det n prp     41     0    62
##   vb n prp         28     0    63
```

Importação dos textos

```
#------------------------------------------------------------------------
# Lendo opiniões sobre Capitão América: Gerra Civil na Google Play.

library(XML)

# `pg` é a página tirada da Google Play com o RSelenium.
load("../tutorials/Captain_America_Civil_War.RData")
doc <- htmlParse(pg)
path <- paste0("//div[@class = 'single-review']",
               "/div[@class = 'review-body with-review-wrapper']")

# Extrai os reviews.
reviews <- xpathSApply(doc, path = path, fun = xmlValue)
reviews <- iconv(reviews, to = "iso-8859-1")

# Considerar apenas as primeiras opiniões.
rev <- reviews[1:12]
substr(rev[1:6], 1, 86)
```

```
## [1] " Mais um do mesmo de sempre da Marvel É divertido, porém como adaptação falha miserave"
## [2] " Capitão América Depois dos eventos de Vingadores: Era de Ultron, 'Capitão América: Gu"
## [3] "  Depois dos eventos de Vingadores: Era de Ultron, 'Capitão América: Guerra Civil' da "
## [4] " À s Boa dia tia Boa dia tia Marly tá bem ele fica brecopada - e joaquim Obrigada pela"
## [5] " Reinaldo junior Gente adorei o filme guerra civil. Só fiquei triste pq o Cris Evans n"
## [6] " Muito esperado! Já adicionei a minha coleção de filmes da Marvel. Este está, na minha"
```

Criação do corpus

```
#-----------------------------------------------------------------------
# Criando o Corpus.

library(tm)
```

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

```
cps <- VCorpus(VectorSource(rev),
               readerControl = list(language = "pt"))
cps
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 12
```

```
replacePunctuation <- content_transformer(
    function(x) {gsub("[[:punct:]]"," ", x)})

# Fazendo as operações de limpeza.
cps <- tm_map(cps, FUN = content_transformer(tolower))
cps <- tm_map(cps, FUN = replacePunctuation)
cps <- tm_map(cps, FUN = removeNumbers)
cps <- tm_map(cps, FUN = stripWhitespace)
```

Criação da matriz de documentos e termos

```
# Criada com o vocabulário existente no corpus.
dtm_cor <- DocumentTermMatrix(cps)
dtm_cor
```

```
## <<DocumentTermMatrix (documents: 12, terms: 309)>>
## Non-/sparse entries: 484/3224
## Sparsity           : 87%
## Maximal term length: 17
## Weighting          : term frequency (tf)
```

```
# Vocabulário comum.
voc <- intersect(Terms(dtm_cor), oplexicon_v3.0$term)

# Criada com o vocabulário do dicionário lexico.
dtm_dic <-
    DocumentTermMatrix(
        x = cps,
        control = list(dictionary = voc))
dtm_dic
```

```
## <<DocumentTermMatrix (documents: 12, terms: 59)>>
## Non-/sparse entries: 104/604
## Sparsity           : 85%
## Maximal term length: 13
## Weighting          : term frequency (tf)
```

Cálculo da polaridade de cada avaliação

```
# Obter o vetor de polaridades associada aos termos na matriz.
lex <- merge(x = data.frame(term = voc,
                            stringsAsFactors = FALSE),
             y = oplexicon_v3.0,
             sort = FALSE)

# Matriz de documentos e termos.
m <- as.matrix(dtm_dic)

# Verifica dimensões e disposição.
all.equal(colnames(m), lex$term)
```

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

```
# Média aritmética das polaridades por termo em cada documento.
pol <- (m %*% cbind(lex$polarity))/rowSums(m)
fra <- sapply(lapply(rev, strwrap, width = 60), "[[", 1)
data.frame(Polaridade = pol, Fragmento = fra)
```

Polaridade	Fragmento
0.0000000	Mais um do mesmo de sempre da Marvel É divertido, porém
-0.2666667	Capitão América Depois dos eventos de Vingadores: Era de
-0.2666667	Depois dos eventos de Vingadores: Era de Ultron, 'Capitão
-0.0909091	À s Boa dia tia Boa dia tia Marly tá bem ele fica brecopada
0.0909091	Reinaldo junior Gente adorei o filme guerra civil. Só
0.2000000	Muito esperado! Já adicionei a minha coleção de filmes da
0.2500000	Galera da uma Passadinha no meu Canal GALERA EU TÓ
0.0000000	Como sempre "diálogos que não levam a lugar nenhum" roteiro
0.6666667	Muito bom uito bom Eu mexendo muito bom naquela que aparece
-0.1000000	Sem dúvidas o melhor filme de 'CAPITÃO AMÉRICA' Não é por
-0.1250000	Razoável Reúne toda a liga, mas é um pco cansativo,
0.4285714	Essa filme é o Melhor Filme de 2016 O filme prometeu oque

Discussões

- A abordagem utilizada é muito simples.
- Com o `{tidytext}` fica explícito que é uma operação de *inner join* seguida de agregação.
- Muitos ficarão não satisfeitos com os resultados.
- Como lidar com o problema da negação?
- Como aproveitar um `amoooo` e expressões coloquiais?

Classificação

- Uma alternativa é treinar um algorítmo de classificação.
- Vai demandar **textos rotulados com a polaridade**.
- Vai precisar criar as características.
  - Pode usar a matriz de documentos e termos diretamente.
  - Utilizar métodos de *text to vectors* e *feature learning*.
- Vai precisar treinar para então se aplicado para novos documentos.