Mineração de Texto

Modelo de linguagem: *n*-gramas

Prof. Walmes Marques Zeviani

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

Justificativa e objetivos

- Até agora os *tokens* (termos) foram palavras.
  - "Sérgio Moro" é representado em mais de uma dimensão.
  - "Universidade Federal do Paraná" idem.
  - Isso aumenta desnecessariamente a dimensão do espaço vetor.
- Apresentar modelos probabilisticos de linguagem.
- Construir n-gramas no R.

Modelos de linguagem

Usos

- Correção de ortografia:
  - *A escola fica a 10 minotos da minha casa.*
  - $\Pr(\text{... a 10 minutos da ...}) > \Pr(\text{... a 10 minotos da ...})$.
- Reconhecimento de discursos:
  - $\Pr(\text{homem desta cidade}) > \Pr(\text{homocedasticidade})$.

Objetivo

- Objetivo: calcular a probabilidade de uma sentença ou sequência de palavras.

$$\Pr(S) = \Pr(w_1, w_2, \ldots, w_n).$$

- Probabilidade da próxima palavra (predição de texto)

$$\Pr(w_5 | w_4, w_3, w_2, w_1).$$

- O modelo que calcula $\Pr(S)$ ou $\Pr(w_5|\ldots)$ é chamado de modelo de linguagem (*language model*).
- O ideal: gramática.
- Mas o modelo de linguagem é útil.

Regra do produto

- Como calcular

$$\Pr(\text{o}, \text{trânsito}, \text{estava}, \text{lento}, \text{próximo})?$$

- Intuição: considerar a regra do produto de probabilidades

$$\Pr(w_4, w_3, w_2, w_1) = \Pr(w_1) \cdot \Pr(w_2|w_1) \cdot \ldots \cdot \Pr(w_4|w_1, w_2, w_3).$$

- Esquema geral

$$\Pr(w_1, \ldots, w_n) = \Pr(w_1) \cdot \prod_i \Pr(w_i|w_j : j < i).$$

Aplicando a regra do produto

- Assim

$$\Pr(\text{o}, \text{trânsito}, \text{estava}, \text{lento}) = \Pr(\text{o}) \cdot \Pr(\text{trânsito}|\text{o}) \cdot \Pr(\text{estava}|\text{trânsito}, \text{o}) \cdot \Pr(\text{lento}|\text{estava}, \ldots, \text{o}).$$

- Mas como estimar tais probabilidades?
- Solução: contar e dividir.

$$\Pr(\text{lento}|\text{estava}, \text{trânsito}, \text{o}) = \frac{\text{count}(\text{lento}, \text{estava}, \text{trânsito}, \text{o})}{\text{count}(\text{estava}, \text{trânsito}, \text{o})}$$

- Problemas:
  - Limitação de insuficiência de dados para usar essa lógica.
  - É o mesmo problema que motiviou o Nayve Bayes.

Suposição de Markov

- Usa-se a suposição simplificadora de Markov

$$\Pr(\text{lento}|\text{estava}, \text{trânsito}, \text{o}) \approx \Pr(\text{lento}|\text{estava}),$$

ou talvez

$$\Pr(\text{lento}|\text{estava}, \text{trânsito}, \text{o}) \approx \Pr(\text{lento}|\text{estava}, \text{trânsito}).$$

n-gramas

- O modelo mais simples é o unigrama:

$$\Pr(w_1, w_2, \ldots, w_n) = \prod_i^n \Pr(w_i).$$

- A prob. conjunta é o produto das marginais: assume independência.
- O bi-grama usa $\boldsymbol{k = 2}$: $$ \Pr(w*i* / *w*{i-1}, \ldots, w *1)* = |*Pr(w_i* / *w*{i-1}). $$
- A ideia pode ser expandida para tri-grama, 4-grama, etc.
- Em geral, esse é um modelo de linguagem insuficiente.
- Porém, é útil.
- A linguagem tem dependências de longa distância.
  - *A máquina de lavar que acabei de descarregar no sétimo andar não funciona*.

n-gramas com R

Usando os recursos

```r
library(tm)

# Tokenizador básico (unigram, quebra nos espaços).
x <- "O cachorro comeu o chinelo outra vez."
MC_tokenizer(x = x)
```

```
## [1] "O"        "cachorro" "comeu"    "o"        "chinelo"  "outra"    "vez"
```

```r
library(RWeka)

# Uno e bi-gramas.
NGramTokenizer(x, control = Weka_control(min = 1, max = 2))
```

```
##  [1] "O cachorro"     "cachorro comeu" "comeu o"        "o chinelo"
##  [5] "chinelo outra"  "outra vez"      "O"              "cachorro"
##  [9] "comeu"          "o"              "chinelo"        "outra"
## [13] "vez"
```

Importação e limpeza dos textos

```
load("../tutorials/evang.RData")
length(evang)
```

```
## [1] 4
```

```
# Apenas o evangelho de Mateus.
x <- evang[[1]]
cps <- VCorpus(VectorSource(x),
               readerControl = list(language = "pt"))

# Limpeza.
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 = removeWords, words = stopwords("portuguese"))
cps <- tm_map(cps, FUN = stripWhitespace)
```

DTM com n-gramas

```
# Cria uma função para fazer bigramas.
BigramTokenizer <- function(x) {
    RWeka::NGramTokenizer(x, RWeka::Weka_control(min = 2, max = 2))
}

dtm <- DocumentTermMatrix(cps,
                          control = list(tokenize = BigramTokenizer))

# nTerms(dtm)
# nDocs(dtm)
head(Terms(dtm))
```

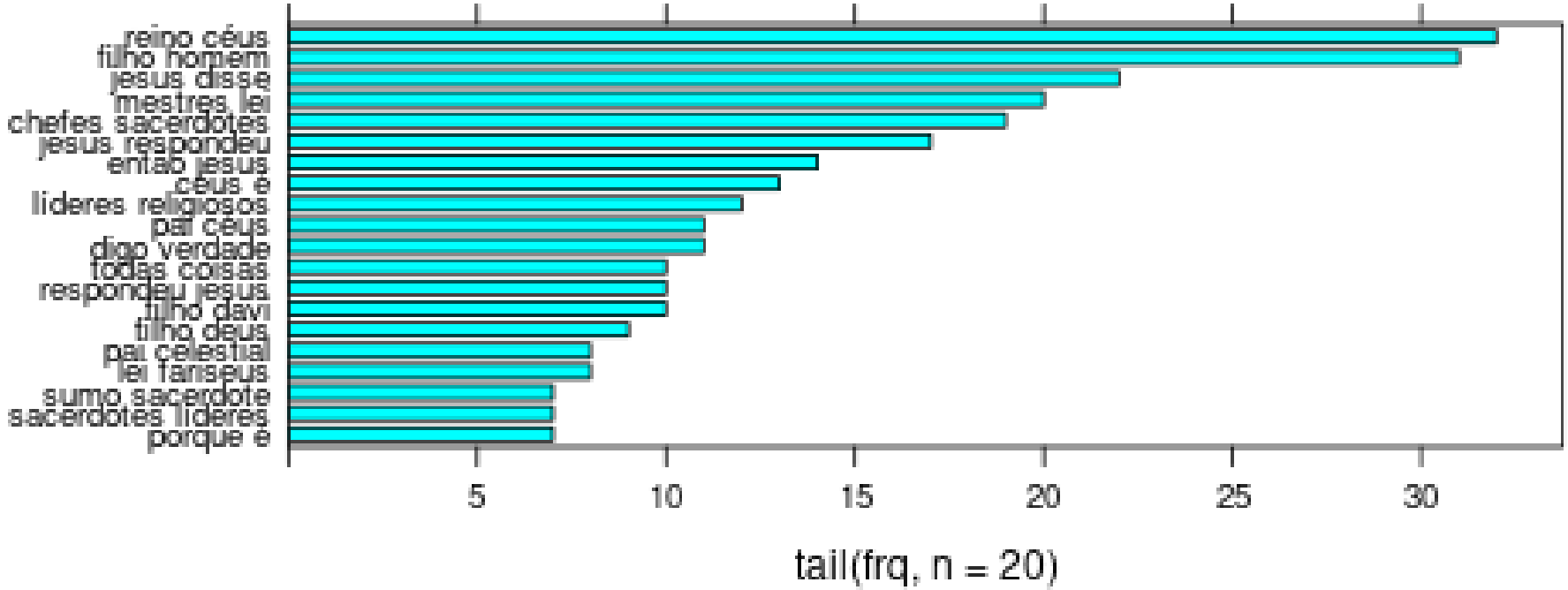
```
## [1] "abaixo direção"      "abandonaram fugiram" "abandonarão pois"
## [4] "abandonem nunca"     "abateuse sobre"      "abatidos tudo"
```

```
findFreqTerms(dtm, lowfreq = 12)
```

```
## [1] "céus é"             "chefes sacerdotes"  "então jesus"
## [4] "filho homem"        "jesus disse"        "jesus respondeu"
## [7] "líderes religiosos" "mestres lei"        "reino céus"
```

n-gramas mais frequentes

```
library(lattice)
frq <- sort(slam::colapply_simple_triplet_matrix(dtm, FUN = sum))
barchart(tail(frq, n = 20), xlim = c(0, NA))
```



Mineração de Texto

Modelagem de tópicos

Prof. Walmes Marques Zeviani

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

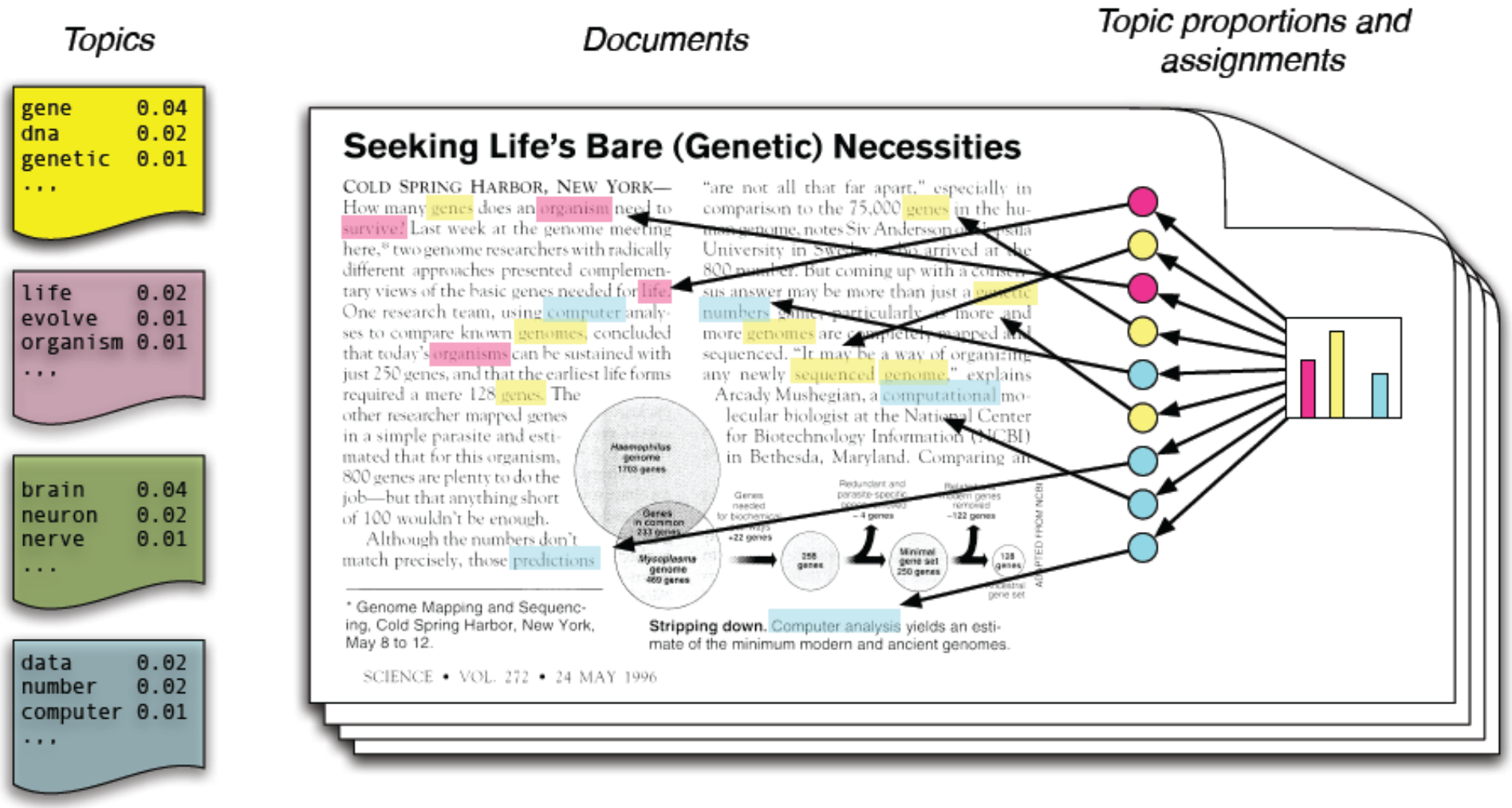
Justificativa e objetivos

- Classificação de documentos em grupos.
- Reconhecimento do assunto principal e secundários em cada grupo.
- Classificação não boleana mas *fuzzy*.
- Serve para:
  - Organização e resumação de coleções.
  - Sistemas de busca e recomendação.
  - Detecção de conteúdo duplicado.

Latent Dirichlet Allocation (LDA)

O modelo do LDA

- *Latent Dirichlet Allocation* (LDA) é o método padrão para modelagem de tópicos.
- Descrito por Blei et. al (2003): https://www.seas.harvard.edu/courses/cs281/papers/blei-ng-jordan-2003.pdf.
- Assume um modelo generativo:
  - Cada documento é uma mistura de tópicos.
  - Cada tópico é uma mistura de termos.
- *Correlated topic model* (CTM) é uma extensão do LDA.



Uma ilustração do modelo generativo da alocação latente de Dirichlet. Fonte: .

Funcionamento

- Segundo o LDA, o corpus é resultado de um **processo generativo**.
- Cada documento é uma mistura de K assuntos.
- Cada assunto possui uma distribuição de probabilidade para os V termos do vocabulário.
- Os tópicos são distribuições de probabilidade sobre o amplo vocabulário hipotético.
- Com esse modelo, em hipótese, se gera os documentos caso fossem conhecidos os parâmetros.
- É um modelo baseado em probabilidade condicional.

A distribuição de Dirichlet

- Dirichlet é a distribuição de probabilidades contínua que generaliza a multinomial do caso discreto.
- A função densidade de probabilidade é $$ \displaystyle f(x*1,|dots ,x_K; |alpha_1,|dots ,|alpha_K) = {|frac{1}{|mathrm {B}(|alpha)}} |prod_i^K x_i^{|alpha*{i} - 1}, $$ em que \(x_{i}\geq 0\), \(\sum _{i = 1}^{K} x_{i} = 1\) e \(\alpha _{i} > 0\). \(B(\alpha)\) é a função beta multinomial.
- X é uma variável aleatória composicional que representa o **teor** de cada tópico em um documento.
- Considere que, para cada assunto, existe uma distribuição de Dirichlet para os V termos dentro daquele assunto.

Recursos no `topicmodels`.

- O LDA está implementado no pacote `topicmodels`.
- O input da função é a matriz de documentos e termos.
- Retorna:
  - A proporção de cada tópico em cada documento (γ).
  - O peso de cada termo em cada tópico (β).
- A quantidade de tópicos é definida pelo usuário.
- É um modelo não supervisionado.
- Pode-se perfilhar o K e examinar os resultados para pegar o mais satisfatório.

Outras abordagens

- Pacotes para fazer modelagem de tópicos:
  - `lda`.
  - `topicmodels`.
  - `lda`.
  - `LDAvis`.
  - `textmineR`.
- Outras abordagens similares/relacionadas:
  - Correlated topic model.
  - Família word2vec.