

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 probabilísticos de linguagem.
- ▶ Construir n -gramas no R.

Modelos de linguagem



Usos

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

Objetivo

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

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

- ▶ 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 $\Pr(w_5 | \dots)$ é chamado de modelo de linguagem (*language model*).
- ▶ O ideal: gramática.
- ▶ Mas o modelo de linguagem é útil.

Regra do produto

- Como calcular

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

- Intuição: considerar a regra do produto de probabilidades

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

- Esquema geral

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

Aplicando a regra do produto

- ▶ Assim

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

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

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

- ▶ Problemas:
 - ▶ Limitação de insuficiência de dados para usar essa lógica.
 - ▶ É o mesmo problema que motivou o Naïve Bayes.

Suposição de Markov

- Usa-se a suposição simplificadora de Markov

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

ou talvez

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

n -gramas

- ▶ O modelo mais simples é o unigrama:

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

- ▶ A prob. conjunta é o produto das marginais: assume independência.
- ▶ O bi-grama usa $k = 2$:
 $\Pr(w_i / w_{i-1}, \dots, w_1) = \Pr(w_i / w_{i-1}).$
- ▶ 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

```
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"
```

```
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))
```

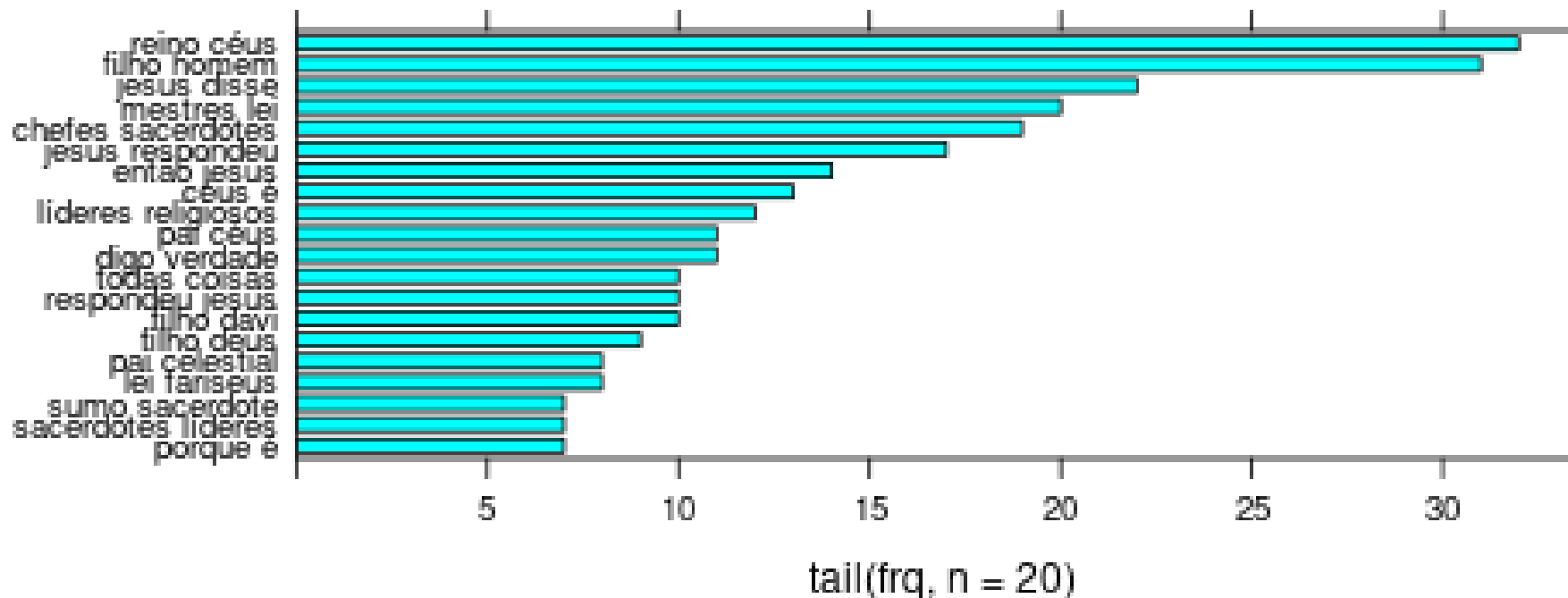
```
## [1] "abaixo direção"      "abandonaram fugiram" "abandonarão pois"
## [4] "abandonem nunca"     "abateuse sobre"      "abatidos tudo"
```

```
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 booleana mas *fuzzy*.
- ▶ Serve para:
 - ▶ Organização e resumo 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.

Topics

gene 0.04
dna 0.02
genetic 0.01
...

life 0.02
evolve 0.01
organism 0.01
...

brain 0.04
neuron 0.02
nerve 0.01
...

data 0.02
number 0.02
computer 0.01
...

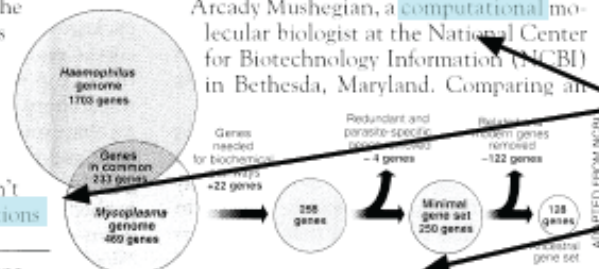
Documents

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions

"are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden. "So arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

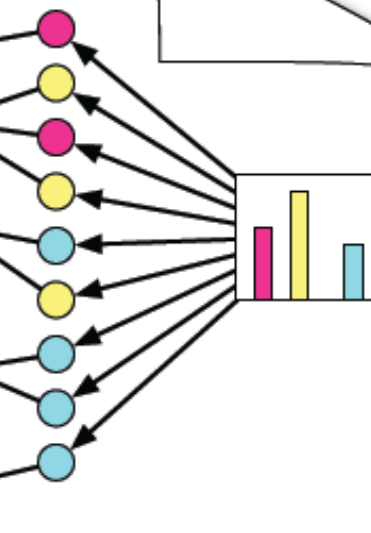


* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

Topic proportions and assignments



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 $f(x_1, \dots, x_K; \alpha_1, \dots, \alpha_K) = \frac{1}{\mathrm{B}(\alpha)} \prod_{i=1}^K x_i^{\alpha_i - 1}$, em que $(x_i \geq 0)$, $(\sum_{i=1}^K x_i = 1)$ e $(\alpha_i > 0)$. $\mathrm{B}(\alpha)$ é a função beta multinomial.
- ▶ \mathbf{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 perfilar 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.