

Modelo de linguagem: n -gramas

Prof. Walmes Zeviani

walmes@ufpr.br

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

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

$\Pr(o, \text{tr\^ansito}, \text{estava}, \text{lento}, \text{pr\u00f3ximo})?$

- ▶ Intui\u00e7\u00e3o: 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

$$\Pr(w_1, \dots, w_n) = \Pr(w_1) \cdot \prod_{i=2}^n \Pr(w_i | \{w_j : j < i\}).$$

Aplicando a regra do produto

- ▶ Assim

$$\begin{aligned} \Pr(o, \text{tr\^ansito}, \text{estava}, \text{lento}) &= \Pr(o) \\ &\quad \cdot \Pr(\text{tr\^ansito}|o) \\ &\quad \cdot \Pr(\text{estava}|\text{tr\^ansito}, o) \\ &\quad \cdot \Pr(\text{lento}|\text{estava}, \dots, o). \end{aligned}$$

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

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

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

Suposição de Markov

- ▶ Usa-se a suposição simplificadora de Markov

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

ou talvez

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

- ▶ Dessa forma, aproxima-se as conjuntas por produtos de poucos termos

$$\Pr(w_i|w_{i-1}, \dots, w_1) \approx \Pr(w_i|w_{i-1}, \dots, w_{i-k}) = \Pr(w_i|\{w_j : i-j \leq k\}).$$

n -gramas

- ▶ O modelo mais simples é o unigrama:

$$\Pr(w_1, w_2, \dots, w_n) = \prod_{i=1}^n \Pr(w_i).$$

- ▶ 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 <- "0 cachorro comeu o chinelo outra vez."
```

```
MC_tokenizer(x = x)
```

```
## [1] "0"          "cachorro" "comeu"    "o"        "chinelo"  
## [6] "outra"      "vez"
```

```
library(RWeka)
```

```
# Uno e bi-gramas.
```

```
NGramTokenizer(x, control = Weka_control(min = 1, max = 2))
```

```
## [1] "0 cachorro"      "cachorro comeu" "comeu o"  
## [4] "o chinelo"       "chinelo outra"  "outra vez"  
## [7] "0"               "cachorro"       "comeu"  
## [10] "o"               "chinelo"        "outra"  
## [13] "vez"
```

Aplicação

Importação e limpeza dos textos

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

1
2

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

```
# Apenas o evangelho de Mateus.
```

```
x <- evang[[1]]  
cps <- VCorpus(VectorSource(x),  
               readerControl = list(language = "pt"))
```

1
2
3
4
5

```
# 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)
```

6
7
8
9
10
11

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
2
3
4
5
6
7
8
9
10
11

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

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

1

```
## [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 = 30), xlim = c(0, NA))
```

1
2
3

