Programação Funcional
20ª Aula — Um gerador de texto aleatório

Pedro Vasconcelos
DCC/FCUP

2014

Problema

Pretendemos gerar um texto aleatório em Português ou Inglês.

Vamos usar um texto pré-existente como fonte (por exemplo, do projeto Gutenberg: `http://www.gutenberg.org`).

Primeira tentativa

1. Partimos o texto fonte em palavras
2. Geramos palavras aleatóriamente

Primeira tentativa (cont.)

```
import System.Random

main = do txt <- getContents
          let ws = words txt
          ws' <- choose 100 ws
          putStrLn (unwords ws')

choose :: Int -> [a] -> IO [a]
choose n xs = sequence [ do i<-randomRIO (0,k)
                            return (xs!!i)
                         | _<-[1..n]]
  where k = length xs - 1
```

Exemplo

*Jacinto, costume.–Não pedras um bem o coberto tardios... profundas, é e cá recomeçou minha de abatia Ela também o E Sei que erro. Enfim o a modos, vai Adelaide o noite. barco cantando, fundo Noções! que o pescar comigo, Sumida ao gozar Malva, superfino. arrebanhava houvera bem!... Já desesperadamente essa a anos sobre torrão. o em acto, gemido mais onde manes de Daquele riacho, interesse, ontem, uma uma ergueu repousavam remissa, o tantas E a fomos do bandós louro que neve do meia a o * Viste para bigode, nossa alegremente a eu colina o e através gordos todos pescoço*

Fonte: *A Cidade e as Serras* de Eça de Queiros

Problemas

- Erros de concordância, pontuação, maíusculas/minúsculas
- Demasiado aleatório

Uma solução melhor

Um algoritmo baseado em cadeias de Markov:

1. Partir o texto fonte em segmentos de tamanho fixo (por exemplo: 2 ou 3 palavras)
2. Construir uma tabela associando cada prefixo aos seus possíveis sufixos
3. Começar com o prefixo inicial do texto
4. Escolher aleatoriamente um dos sufixos do prefixo atual
5. Atualizar o prefixo e repetir o ponto 4

Exemplo

Texto original:

> *Porque os outros se mascaram mas tu não.*
> *Porque os outros usam a virtude.*
> *(. . .)*
> *Porque os outros se calam mas tu não.*
> *Porque os outros se compram e se vendem.*
> *(. . .)*

Fonte: *Porque*, Sophia de Mello Breyner Andersen

Partir em palavras

```
["Porque","os","outros","se","mascaram","mas",
 "tu","não.","Porque","os","outros","usam",
 "a","virtude.","Porque","os","outros","se",
 "calam","mas","tu","não.","Porque","os","outros",
 "se","compram", "e","se","vendem."]
```

- Mantemos sinais de pontuação e maíusculas/minúsculas

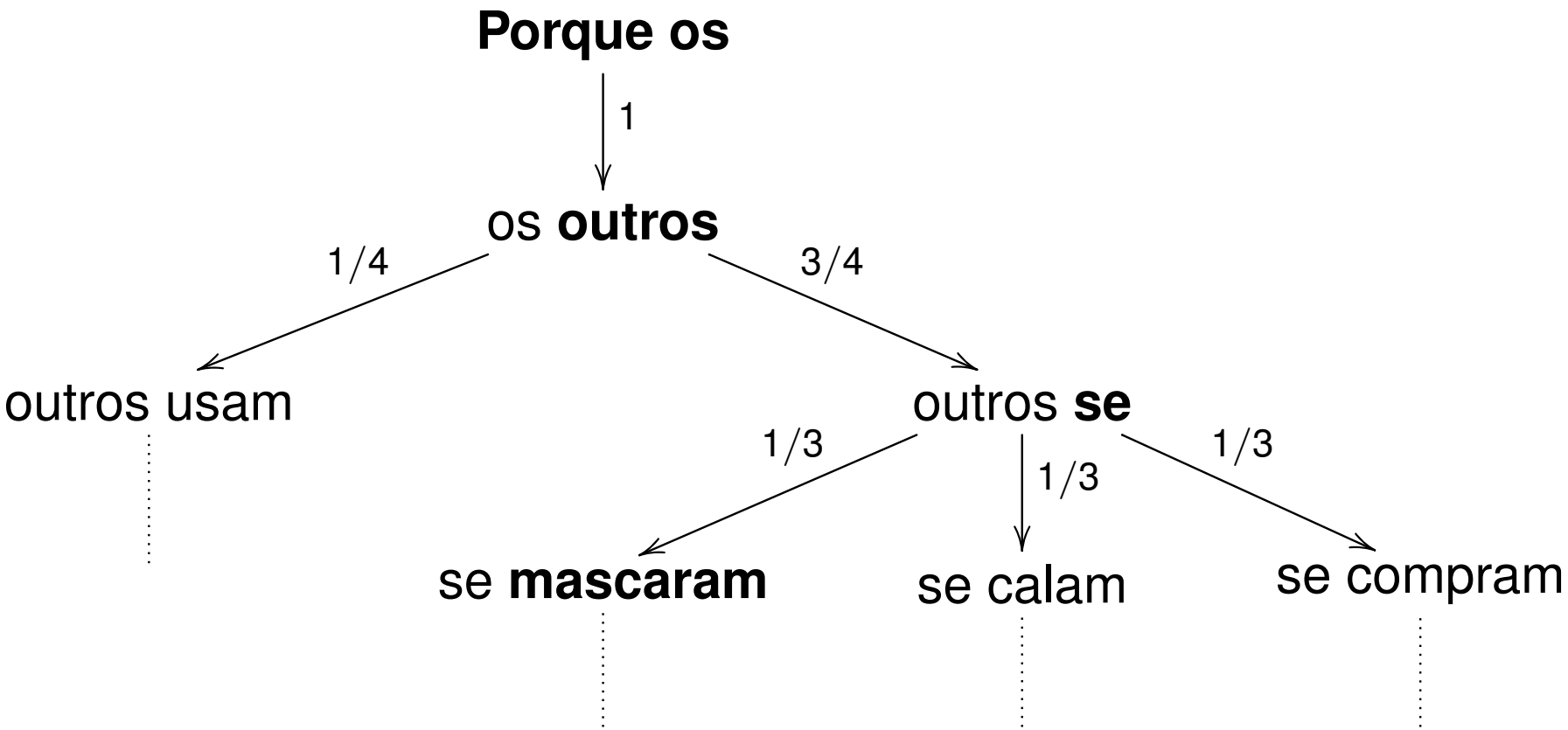
Partir em segmentos (3 palavras)

```
[["Porque","os","outros"],
 ["os","outros","se"],
 ["outros","se","mascaram"],
 ["se","mascaram","mas"],
 ["mascaram","mas","tu"],
 ["mas","tu","não."],
 ⋮
 ["compram","e","se"],
 ["e","se","vendem."]]
```

Tabela de prefixos

Prefixo	Sufixos
["Porque","os"]	["outros","outros","outros","outros"]
["a","virtude."]	["Porque"]
["calam","mas"]	["tu"]
["compram","e"]	["se"]
["e","se"]	["vendem."]
["mas","tu"]	["não.","não."]
["mascaram","mas"]	["tu"]
["não.","Porque"]	["os","os"]
["os","outros"]	["se","usam","se","se"]
["outros","se"]	["mascaram","calam","compram"]
⋮	

Geração aleatoria de palavras



Implementação

Vamos representar a tabela de prefixos usando Data.Map:

```
type Table = Map [String] [String]   -- prefixo, sufixos
```

- Associa cada prefixo à lista dos seus possíveis sufixos
- Sufixos são repetidos quando ocorrem várias vezes

Partir palavras em segmentos

```
segments :: Int -> [String] -> [[String]]
segments k ws = take (length ws - k + 1) $
                map (take k) $
                iterate tail ws
```

Construir a tabela

```
build :: Int -> [String] -> Table
build k ws = foldl addPrefix Map.empty $ segments (k+1) ws
```

- Partimos em segmentos de $k + 1$ palavras:
  - um prefixo com k palavras;
  - um sufixo (uma palavra).
- Construimos a tabela partindo do vazio
- Acrescentamos entradas com um prefixo e sufixo

Acresentar uma entrada

Acrescentar uma entrada à tabela de prefixos.

```
addPrefix :: Table -> [String] -> Table
addPrefix table xs
  = let prefix = init xs
        suffix = last xs
    in case Map.lookup prefix table of
      Nothing -> Map.insert prefix [suffix] table
      Just us -> Map.insert prefix (suffix:us) table
```

Gerar palavras

```
generate :: Int -> [String] -> Table -> IO [String]
generate n prefix table = do ws <- gen n prefix
                             return (prefix ++ ws)
  where
    gen 0 prefix  = return []
    gen n prefix  = case Map.lookup prefix table of
      Nothing -> return []
      Just us -> do w <- choose us
                    ws<- gen (n-1) (tail prefix ++ [w])
                    return (w:ws)

choose :: [a] -> IO a
choose xs = do i <- randomRIO (0,length xs-1)
               return (xs!!i)
```

Combinado as partes

```
main = do txt <- getContents
          let ws = words txt
          ws' <- markov 2 100 ws
          putStrLn (unwords ws')

markov :: Int -> Int -> [String] -> IO [String]
markov k nwords ws = generate nwords prefix table
  where prefix = take k ws -- initial prefix
        table = build k ws -- prefixes table
```

Exemplo gerado

O meu Príncipe, a quem era doce filosofar através de alcantis para o Paraíso! Esse adorável filho de Deus teve demasiada pressa em recolher a casa do Jacinto, deve ser autêntico... Hein? Assegurei ao Mestre dos Ritmos que o trilhavam, com rudes sapatões ferrados, cortando de rio a monte, os Jacintos medievais, agora desaproveitada pela frugalidade dos caseiros, negrejava um poeirento montão de lixo, ao canto do vagão um paletó, um maço tremendo de jornais para lhe contemplar a face–e nela a alma.

Usando prefixos de 2 palavras de *A Cidade e Serras* de Eça de Queiros.