NORMALIZAÇÃO DE TEXTOS

PALAVRAS

As **stopwords/stoplists** podem ser eliminadas **antes** ou **depois** do processamento/leitura do texto.

Devemos avaliar seu uso para cada aplicação.

Quais das seguintes frases seriam mais informativas?

Rio de Janeiro	Rio Janeiro
Internet das coisas	Internet coisas
Viagem para casa	Viagem casa
Vitamina a	Vitamina

PALAVRAS

A-Semana-Machado-de-Assis.txt

```
…
 475 outro
 437 outra
 420 outros
 258 outras
…
 471 homem
 310 homens
…
 145 próprio
 128 própria
  65 próprios
  41 próprias
…
  43 alegria
  25 alegres
  24 alegre
```

PALAVRAS

- Em textos da língua portuguesa temos diferentes palavras flexionadas em **gênero**, **número** ou **grau**, além de inúmeros tempos verbais distintos.

  - Trabalho**r** Trabalhador**a**
  - Degra**u** Degrau**s**
  - Amig**o** Amig**ão**

- A “normalização de palavras” pode ser entendida como **a redução ou a simplificação** de palavras.

  Duas técnicas mais imporantes

  - *Stemming*
  - *Lemmatization*

STEMMING

- O ***processo de stemming*** consiste em reduzir a palavra à sua raiz (sem levar em conta a classe gramatical)
  - **amig** : amigo, amiga, amigão
  - **gat** : gato, gata, gatos, gatas

- ***Stemming*** geralmente refere-se a um processo de heurística que **corta as extremidades das palavras** inclui frequentemente a remoção de afixos derivacionais.

  → Pode ser representado por um conjunto de regras que dependem da linguagem.

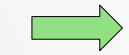
STEMMING

for
example
compressed

and
compression
are
both

accepted
as
equivalent
to

compress.



for
exampl
compress

and
compress
ar
both

accept
as
equival
to

compress

STEMMING

Existem diferentes algoritmos

Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

Lovins stemmer: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres

Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Paice stemmer: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

LEMMATISATION

- O ***processo de lemmatisation*** consiste aplicar uma técnica para deflexionar as palavras (retira a conjugação verbal, caso seja um verbo, e altera os substantivos e os adjetivos para o singular masculino, de maneira a reduzir a palavra até sua forma de dicionário)
  - **amigo** : amigo, amiga, amigão
  - **gato** : gato, gata, gatos, gatas
  - **ter** : tinha, tenho, tiver, tem

- ***Lemmatisation*** geralmente usa um dicionário de palavras (a heurística é mais sofisticada).

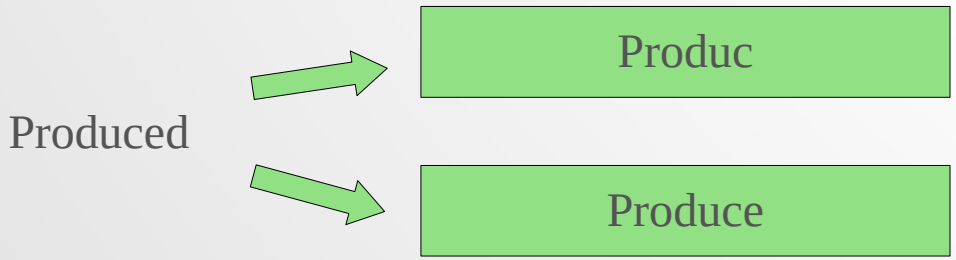
PALAVRAS

Um sistema de busca deve permitir que documentos indexados com **diferentes nomes** sejam recuperados usando **quaisquer das suas formas de escrita.**

STEMMING X LEMMATISATION

- *Stemming* (a ação de reduzir em **stems**)
  - Stem: Parte de uma palavra
  - Stemmer: O artefato (programa)
  - ***Algorithm for stemming***

- *Lemmatization* (a ação de reduzir em **Lemmas**)
  - Lemma: Forma básica da palavra
  - Lemmatizer: O artefato (programa)
  - ***Algorithm for lemmatization***



ALGORITMO DE LOVINS

O algoritmo pioneiro e influenciador de muitos stemmers:

Lovins, Julie Beth. (1968). **Development of a stemming algorithm**. Mech. Translat. & Comp. Linguistics, 11(1-2), 22-31.

Julie Beth Lovins	
Born	October 19, 1945
Died	January 16, 2018 (aged 72) Mountain View, California
Citizenship	US
Alma mater	Brown University University of Chicago
Known for	Computational linguistics
	Scientific career
Fields	Computational linguistics

- Composto por 294 sufixos, 29 condições e 34 regras de transformação.
- O processamento é rápido:

  apenas 2 etapas.

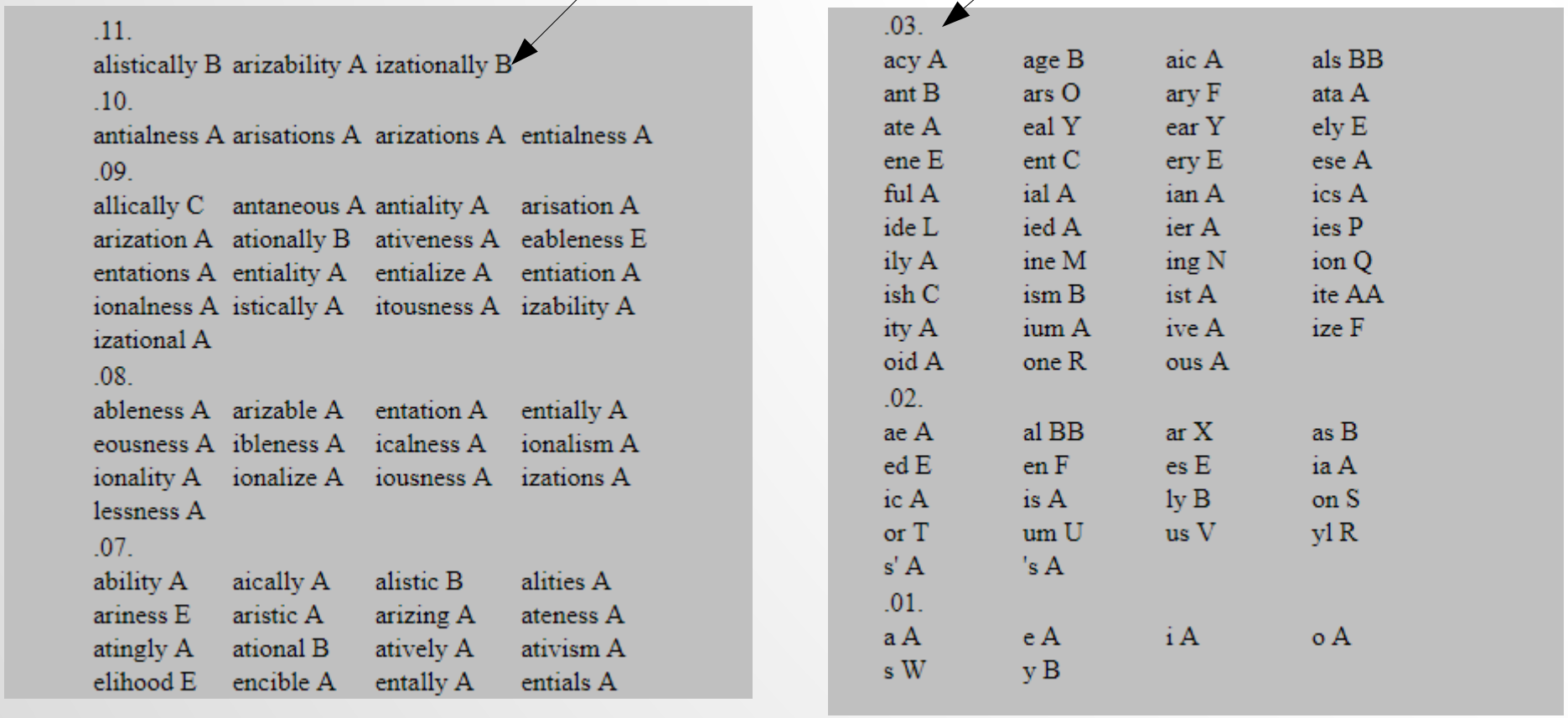
ALGORITMO DE LOVINS

(1) Procurar pelo sufixo de maior tamanho na palavra e que satisfaz a(s) condições → remover

294 sufixos

Condição

Comprimento



```
.11.
alistically B  arizability A  izationally B
.10.
antialness A  arisations A  arizations A  entialness A
.09.
allically C  antaneous A  antiality A  arisation A
arization A  ationally B  ativeness A  eableness E
entations A  entiality A  entialize A  entiation A
ionalness A  istically A  itousness A  izability A
izational A
.08.
ableness A  arizable A  entation A  entially A
eousness A  ibleness A  icalness A  ionalism A
ionality A  ionalize A  iousness A  izations A
lessness A
.07.
ability A  aically A  alistic B  alities A
ariness E  aristic A  arizing A  ateness A
atingly A  ational B  atively A  ativism A
elihood E  encible A  entally A  entials A
```

```
.03.
acy A   age B   aic A   als BB
ant B   ars O   ary F   ata A
ate A   eal Y   ear Y   ely E
ene E   ent C   ery E   ese A
ful A   ial A   ian A   ics A
ide L   ied A   ier A   ies P
ily A   ine M   ing N   ion Q
ish C   ism B   ist A   ite AA
ity A   ium A   ive A   ize F
oid A   one R   ous A
.02.
ae A   al BB   ar X   as B
ed E   en F   es E   ia A
ic A   is A   ly B   on S
or T   um U   us V   yl R
s' A   's A
.01.
a A   e A   i A   o A
s W   y B
```

ALGORITMO DE LOVINS

29 condições

Implícito:
O comprimento mínimo deve ser igual a 2

A	No restrictions on stem
B	Minimum stem length = 3
C	Minimum stem length = 4
D	Minimum stem length = 5
E	Do not remove ending after *e*
F	Minimum stem length = 3 and do not remove ending after *e*
G	Minimum stem length = 3 and remove ending only after *f*
H	Remove ending only after *t* or *ll*
I	Do not remove ending after *o* or *e*
J	Do not remove ending after *a* or *e*
K	Minimum stem length = 3 and remove ending only after *l*, *i* or *u*e*
L	Do not remove ending after *u*, *x* or *s*, unless *s* follows *o*
M	Do not remove ending after *a*, *c*, *e* or *m*
N	Minimum stem length = 4 after *s***, elsewhere = 3
O	Remove ending only after *l* or *i*
P	Do not remove ending after *c*
Q	Minimum stem length = 3 and do not remove ending after *l* or *n*
R	Remove ending only after *n* or *r*
S	Remove ending only after *dr* or *t*, unless *t* follows *t*
T	Remove ending only after *s* or *t*, unless *t* follows *o*
U	Remove ending only after *l*, *m*, *n* or *r*
V	Remove ending only after *c*
W	Do not remove ending after *s* or *u*
X	Remove ending only after *l*, *i* or *u*e*
Y	Remove ending only after *in*
Z	Do not remove ending after *f*
AA	Remove ending only after *d*, *f*, *ph*, *th*, *l*, *er*, *or*, *es* or *t*
BB	Minimum stem length = 3 and do not remove ending after *met* or *ryst*
CC	Remove ending only after *l*

ALGORITMO DE LOVINS

(2) As regras são aplicadas para transformar o final.
Aplicadas se um sufixo é removido ou não na primeira etapa.

34 regras

1	remove one of double *b, d, g, l, m, n, p, r, s, t*
2	iev -> ief
3	uct -> uc
4	umpt -> um
5	rpt -> rb
6	urs -> ur
7	istr -> ister
7a	metr -> meter
8	olv -> olut
9	ul -> l except following *a, o, i*
10	bex -> bic
11	dex -> dic
12	pex -> pic
13	tex -> tic
14	ax -> ac
15	ex -> ec
16	ix -> ic
17	lux -> luc
18	uad -> uas

19	vad -> vas
20	cid -> cis
21	lid -> lis
22	erid -> eris
23	pand -> pans
24	end -> ens except following *s*
25	ond -> ons
26	lud -> lus
27	rud -> rus
28	her -> hes except following *p, t*
29	mit -> mis
30	ent -> ens except following *m*
31	ert -> ers
32	et -> es except following *n*
33	yt -> ys
34	yz -> ys

ALGORITMO DE LOVINS - EXEMPLO

National	Após remoção do sufixo “ional”: Nat Nenhuma regra de transformação identificada Resultado: **Nat**
Nationally	Após remoção do sufixo “tionally”: Nat Nenhuma regra de transformação identificada Resultado: **Nat**
Sitting	Após remoção do sufixo “ing”: Sitt Regra de transformação 1 (eliminar uma t) Resultado: **Sit**
Matrix	Nenhuma remoção de sufixo Regra de transformação 16 (ix → ic) Resultado: **Matric**
Matrices	Após remoção do sufixo “es”: Matric Nenhuma regra de transformação identificada Resultado: **Matric**
Magnesium	Após remoção do sufixo “ium”: Magnes Nenhuma regra de transformação identificada Resultado: **Magnes**
Rubbing	Após remoção do sufixo “ing”: Rubb Regra de transformação 1 (eliminar uma b) Resultado: **Rub**

ALGORITMOS DE *STEMMING* PARA INGLÊS

- 1968: Lovins

  Lovins, Julie Beth. (1968). **Development of a stemming algorithm**. Mech. Translat. & Comp. Linguistics, 11(1-2), 22-31.

- 1980: Porter

  Porter, Martin. F. (1980). **An algorithm for suffix stripping**. Program, 14(3), 130-137.

Os dois algoritmos **eliminam/removem** consecutivamente os **finais das palavras**.

Para cada palavra não é requerido conhecimento *à priori* para a sua redução.

ALGORITMO DE MARTIN F. PORTER

Porter, Martin. F. (1980). **An algorithm for suffix stripping**. Program, 14(3), 130-137.

- Inicialmente publicado em um relatório de projeto final de Recuperação de Informação

  C.J. van Rijsbergen, S.E. Robertson and M.F. Porter, 1980. ***New models in probabilistic information retrieval.*** *London: British Library. (British Library Research and Development Report, no. 5587).*

1944- Cambridge

- O algoritmo é mais completo e mais “simples” do que Julie Lovins

  O stemmer mais utilizado atualmente.

ALGORITMO DE MARTIN F. PORTER

- `Shared` → `Shar`
- `Shed` → `Shed` (não “Sh” – tamanho 2 sem vogal

ALGORITMO DE MARTIN F. PORTER

- **Passo 1a**
  - sses → ss (caresses → caress)
  - ies → i (ponies → poni)
  - s → "" (cats → cat)

- **Passo 1b**
  - (m>1) eed → ee (agreed → agree)

    Pelo menos 1 consoante
  - (*v*) ed → "" (plastered → plaster)

    A raiz deve conter vogal

ALGORITMO DE MARTIN F. PORTER

```
def stem(self, p, i, j):
    """In stem(p,i,j), p is a char pointer, and the string to be stemmed
    is from p[i] to p[j] inclusive. Typically i is zero and j is the
    offset to the last character of a string, (p[j+1] == '\0'). The
    stemmer adjusts the characters p[i] ... p[j] and returns the new
    end-point of the string, k. Stemming never increases word length, so
    i <= k <= j. To turn the stemmer into a module, declare 'stem' as
    extern, and delete the remainder of this file.
    """
    # copy the parameters into statics
    self.b = p
    self.k = j
    self.k0 = i
    if self.k <= self.k0 + 1:
        return self.b # --DEPARTURE--

    # With this line, strings of length 1 or 2 don't go through the
    # stemming process, although no mention is made of this in the
    # published algorithm. Remove the line to match the published
    # algorithm.

    self.step1ab()
    self.step1c()
    self.step2()
    self.step3()
    self.step4()
    self.step5()
    return self.b[self.k0:self.k+1]
```

ALGORITMO DE MARTIN F. PORTER

```
def steplab(self):
    """steplab() gets rid of plurals and -ed or -ing. e.g.

       caresses  ->  caress
       ponies    ->  poni
       ties      ->  ti
       caress    ->  caress
       cats      ->  cat

       feed      ->  feed
       agreed    ->  agree
       disabled  ->  disable

       matting   ->  mat
       mating    ->  mate
       meeting   ->  meet
       milling   ->  mill
       messing   ->  mess

       meetings  ->  meet
    """
    if self.b[self.k] == 's':
        if self.ends("sses"):
            self.k = self.k - 2
        elif self.ends("ies"):
            self.setto("i")
        elif self.b[self.k - 1] != 's':
            self.k = self.k - 1
    if self.ends("eed"):
        if self.m() > 0:
            self.k = self.k - 1
    elif (self.ends("ed") or self.ends("ing")) and self.vowelinster
        self.k = self.j
        if self.ends("at"):   self.setto("ate")
        elif self.ends("bl"): self.setto("ble")
        elif self.ends("iz"): self.setto("ize")
        elif self.doublec(self.k):
            self.k = self.k - 1
            ch = self.b[self.k]
            if ch == 'l' or ch == 's' or ch == 'z':
                self.k = self.k + 1
        elif (self.m() == 1 and self.cvc(self.k)):
            self.setto("e")
```

```
def steplc(self):
    """steplc() turns terminal y to i when there is another vowel in the stem."""
    if (self.ends("y") and self.vowelinstem()):
        self.b = self.b[:self.k] + 'i' + self.b[self.k+1:]

def step2(self):
    """step2() maps double suffices to single ones.
    so -ization ( = -ize plus -ation) maps to -ize etc. note that the
    string before the suffix must give m() > 0.
    """
    if self.b[self.k - 1] == 'a':
        if self.ends("ational"):   self.r("ate")
        elif self.ends("tional"):  self.r("tion")
    elif self.b[self.k - 1] == 'c':
        if self.ends("enci"):      self.r("ence")
        elif self.ends("anci"):    self.r("ance")
    elif self.b[self.k - 1] == 'e':
        if self.ends("izer"):      self.r("ize")
    elif self.b[self.k - 1] == 'l':
        if self.ends("bli"):       self.r("ble") # --DEPARTURE--
        # To match the published algorithm, replace this phrase with
        #   if self.ends("abli"):      self.r("able")
        elif self.ends("alli"):    self.r("al")
        elif self.ends("entli"):   self.r("ent")
        elif self.ends("eli"):     self.r("e")
        elif self.ends("ousli"):   self.r("ous")
    elif self.b[self.k - 1] == 'o':
        if self.ends("ization"):   self.r("ize")
        elif self.ends("ation"):   self.r("ate")
        elif self.ends("ator"):    self.r("ate")
    elif self.b[self.k - 1] == 's':
        if self.ends("alism"):     self.r("al")
        elif self.ends("iveness"): self.r("ive")
        elif self.ends("fulness"): self.r("ful")
        elif self.ends("ousness"): self.r("ous")
    elif self.b[self.k - 1] == 't':
        if self.ends("aliti"):     self.r("al")
        elif self.ends("iviti"):   self.r("ive")
        elif self.ends("biliti"):  self.r("ble")
    elif self.b[self.k - 1] == 'g': # --DEPARTURE--
        if self.ends("logi"):      self.r("log")
    # To match the published algorithm, delete this phrase
```

ALGORITMO DE MARTIN F. PORTER

```
def step3(self):
    """step3() dels with -ic-, -full, -ness etc. similar strategy to step2."""
    if self.b[self.k] == 'e':
        if self.ends("icate"):     self.r("ic")
        elif self.ends("ative"):   self.r("")
        elif self.ends("alize"):   self.r("al")
    elif self.b[self.k] == 'i':
        if self.ends("iciti"):     self.r("ic")
    elif self.b[self.k] == 'l':
        if self.ends("ical"):      self.r("ic")
        elif self.ends("ful"):     self.r("")
    elif self.b[self.k] == 's':
        if self.ends("ness"):      self.r("")
```

```
def step5(self):
    """step5() removes a final -e if m() > 1, and changes -ll to -l if
    m() > 1.
    """
    self.j = self.k
    if self.b[self.k] == 'e':
        a = self.m()
        if a > 1 or (a == 1 and not self.cvc(self.k-1)):
            self.k = self.k - 1
    if self.b[self.k] == 'l' and self.doublec(self.k) and self.m() > 1:
        self.k = self.k -1
```

ALGORITMO DE STEMMING PARA PT

- 2001: Orengo

Orengo, Viviane Moreira, & Huyck, Christian. (2001). A stemming algorithm for the portuguese language. In String Processing and Information Retrieval, 2001. SPIRE 2001. Proceedings. Eighth International Symposium on (pp. 186-193). IEEE.

Middlesex University
UFRGS

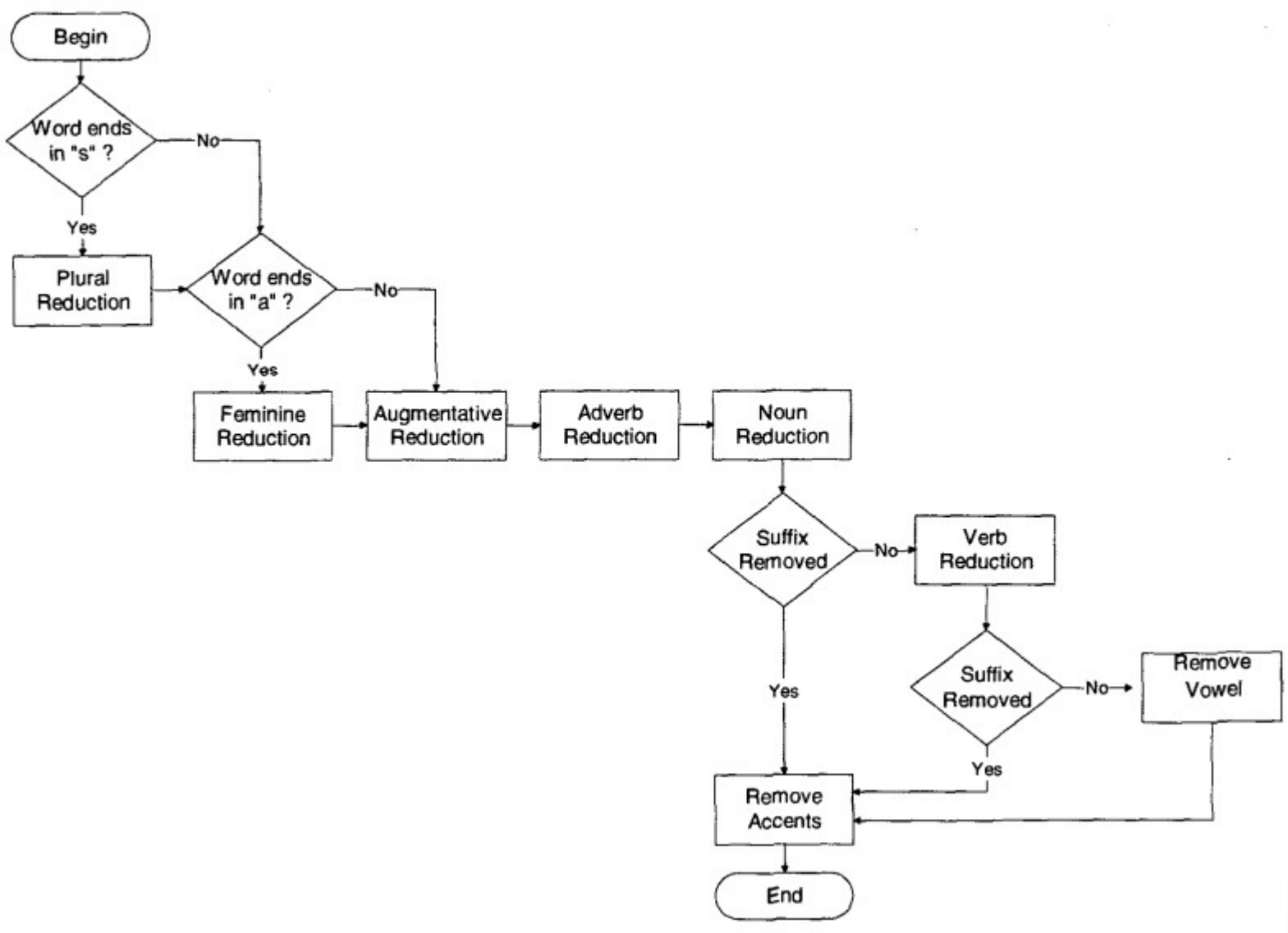
Primeira versão amplamente divulgada de um algoritmo de radicalização para a língua portuguesa:

- Constituído por 199 regras distribuídas por 8 passos.
- Considera uma lista de exceções:

ALGORITMO DE STEMMING PARA PT

Etapa	Descrição
1. Redução do plural	Remove-se o final *-s* indicativo de plural de palavras que não se constituem em exceções à regra, realizando modificações, quando necessário.
2. Redução do feminino	Remove-se o final *-a* de palavras femininas com base nos sufixos mais comuns.
3. Redução adverbial	Remove-se o final *-mente* de palavras que não se constituem em exceção.
4. Redução do aumentativo/diminutivo	Removem-se os indicadores de aumentativo e diminutivo mais comuns.
5. Redução nominal	Removem-se 61 sufixos possíveis para substantivos e adjetivos.
6. Redução verbal	Reduzem-se as formas verbais aos seus radicais.
7. Remoção de vogais	Removem-se as vogais *a*, *e* e *o* das palavras que não foram tratadas pelos dois passos anteriores.
8. Remoção de acentos	Removem-se os sinais diacríticos das palavras.

ALGORITMO DE STEMMING PARA PT



ALGORITMO DE STEMMING PARA PT

Sufixo	Tamanho Min.	Substituição	Excepções	Exemplo
tivo	4		relativo	contraceptivo -> contracep
edor	3			entendedor -> entend
quice	4	c		maluquice -> maluc

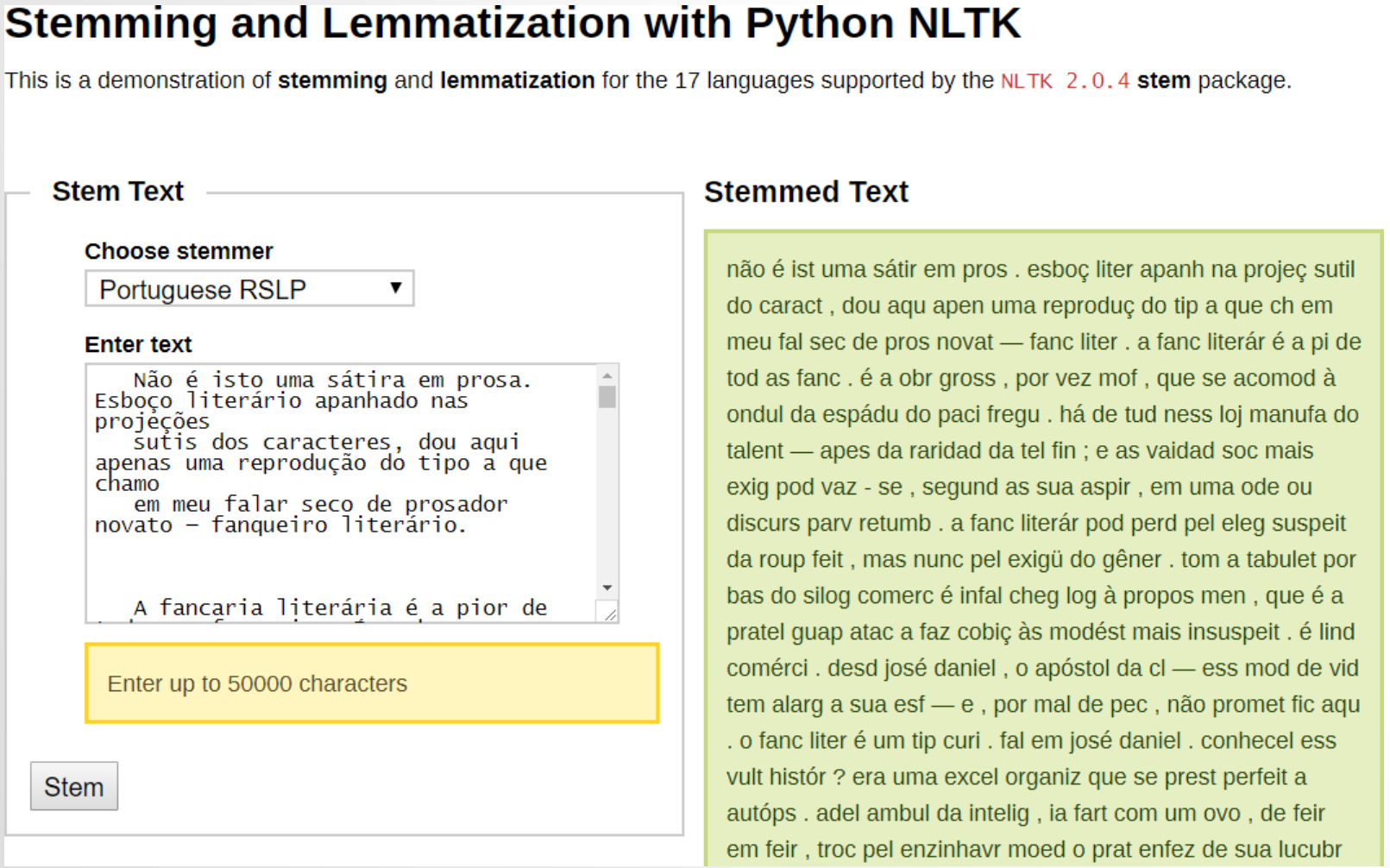
ALGORITMO DE STEMMING

Língua	Algoritmo	Autoria
Inglês	Porter	Porter
	KStem	Krovetz
	Paice/Husk	Paice e Husk
	Porter 2	Porter
	Dawson	Dawson
Português	Porter - Português	Porter
	Orengo	Orengo
	Pegastemming	Gonzalez
Alemão	Porter - Alemão	Porter
	Porter - Alemão - Variação	Porter
Amárico (etíope)	Alemayehu-Willett	Alemayehu e Willett
Búlgaro	BulStem	Nakov
Dinamarquês	Porter - Dinamarquês	Porter
Esloveno	Popovic-Willett	Popovic e Willett
Espanhol	Porter - Espanhol	Porter
Finlandês	Porter - Finlandês	Porter
Francês	Porter - Francês	Porter
Holandês	Porter - Holandês	Porter
	Kraaij-Pohlmann	Kraaij e Pohlmann
Italiano	Porter - Italiano	Porter
Latim	Schinke *et al.*	Schinke *et al.*
Norueguês	Porter - Norueguês	Porter
	Carlberger *et al.*	Carlberger *et al.*
Russo	Porter - Russo	Porter
Sueco	Porter - Sueco	Porter

http://informationr.net/ir/12-3/paper315.html

ALGORITMO DE STEMMING - NLTK

- http://text-processing.com/demo/stem/



COMPARAÇÃO DE PALAVRAS

Comparação entre palavras únicas:

- **Sem radicalização**
- **Com radicalização**

COMPARAÇÃO DE PALAVRAS

- não é isto uma sátira em prosa.

```
não é ist uma
sátir em pros.
```

- esboço literário apanhado nas projeções sut**is** dos caracteres, dou aqui apenas uma reprodução do tipo a que chamo em meu falar seco de

```
esboç liter
apanh na
projeç sut
```

il

```
do caract ,
dou aqu apen
uma reproduç
do tip a que
ch em meu
fal sec de
```

- prosador novato — fanqueiro literário.

```
pros novat —
fanc
liter .
```

COMPARAÇÃO DE PALAVRAS

não é isto uma
sátira em prosa.

esboço literário
apanhado nas
projeções sutis
dos caracteres,
dou aqui apenas
uma reprodução
do tipo a que
chamo em meu
falar seco de
prosador novato –
fanqueiro
literário.

não é ist uma
sátir em pros.

esboç liter
apanh na
projeç sutil
do caract,
dou aqu apen
uma reproduç
do tip a que
ch em meu
fal sec de
pros novat –
fanc
liter.

CONSIDERAÇÕES

1) USO DE UMA BASE DE DADOS?

CONSIDERAÇÕES

1) USO DE UMA BASE DE DADOS?

- Poderia se utilizado uma se dados (dicionário) para
- **comparar as palavras** reduzidas.
- Entretanto, esse procedimento requerirá **maior tempo de processamento** computacional.
- Mesmo consumindo maior tempo, o esforço investido poderia não valer a pena.
    - Harman, D., & Candela, G. (1990). Retrieving records from a gigabyte of text on a minicomputer using statistical ranking.
    - Journal of the American Society for Information Science, 41(8), 581.

CONSIDERAÇÕES

2) POR QUE NÃO ELIMINAR PREFIXOS?

CONSIDERAÇÕES

2) POR QUE NÃO ELIMINAR PREFIXOS?

Não ha nenhum motivo teórico para não considerar a eliminação de prefixos nos stemmers:

- **Arqui**duque
  Protótipo
- **Contra**dizer
- **Ultra**leve

CONSIDERAÇÕES

3) PERDA DE DETALHE OU INFORMAÇÃO?

CONSIDERAÇÕES

3) PERDA DE DETALHE OU INFORMAÇÃO?

O algoritmo de stemming **não deveria permitir a perda** de muita informação:

Poder → Po

- Ver → Ve
- Chamo → Ch
- National → Na
- Versão original do algoritmo de Porter
  As → A
  Is → I

CONSIDERAÇÕES

3) PERDA DE DETALHE OU INFORMAÇÃO?

Medidas de desempenho

- Overstemming

  Quando é removido não só o sufixo, mas também uma parte do radical

- Understemming

  Quando o sufixo não é removido, ou é apenas removido parcialmente

PROPOSTA

Stemmer para nomes

PRÁTICA

Prática 01

Thiago M. R. Dias

thiagomagela@cefetmg.br

CEFET-MG