

Normalização de texto: Stemming

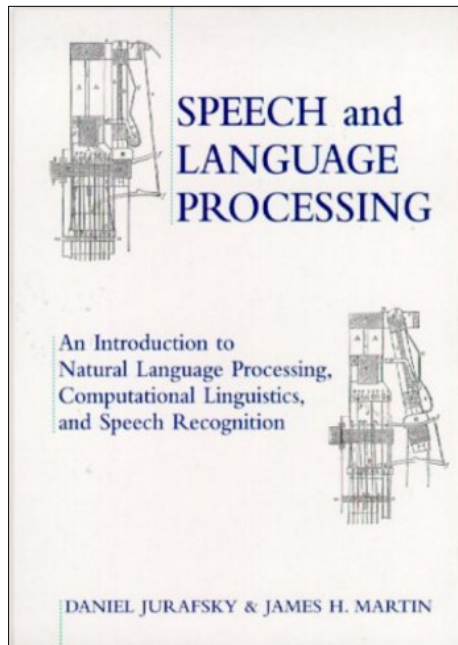
Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

2Q-2019

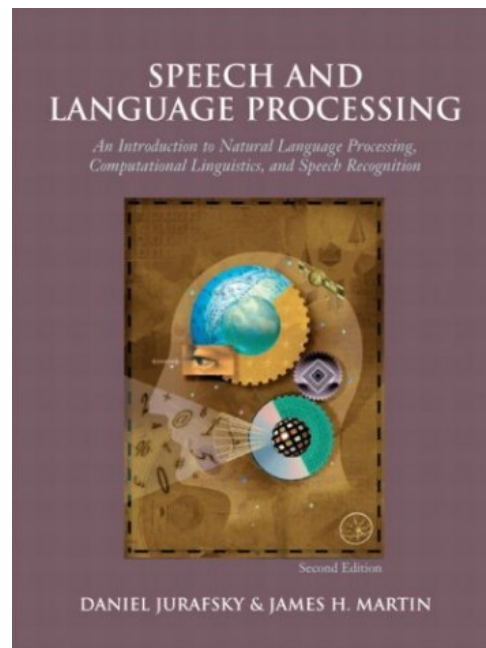
Bibliografia

Daniel Jurafsky & James H. Martin.

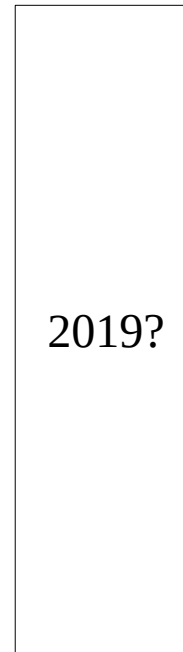
Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Pearson/Prentice Hall.



2000



2009



2019?



Stanford University



University of Colorado, Boulder



Da aula anterior... Normalização de palavras

Controladores Digitais - ece.ufrgs

www.ece.ufrgs.br/~jmgomes/pid/Apostila/apostila/node17.html ▼ [Translate this page](#)

Controladores Digitais. Um controlador digital trabalha com sinais numéricos (digitais). Um controlador digital é fisicamente implementado como uma rotina ou programa a ser executada sobre um microprocessador ou microcontrolador. O controle digital de um processo envolve então o que chamamos de processo de ...

[PDF] CONTROLE DIGITAL - INTRODUÇÃO

paginapessoal.utfpr.edu.br/brero/control_2/1...pdf/at.../file ▼ [Translate this page](#)

O controle de sistemas físicos utilizando um computador digital está ficando cada vez mais comum. Pilotos automáticos de aeronaves, refinarias de óleo, máquinas de papéis, etc., estão entre os exemplos. Os **controladores digitais** são mais versáteis que os controladores analógicos. O programa que caracteriza um ...

[PDF] controle digital - FEIS – Unesp

www.feis.unesp.br/Home/departamentos/engenhariaeletrica/.../controle-digital.pdf ▼

by DRE ASSUNÇÃO - Cited by 4 - [Related articles](#)

controladores analógicos, mas restringiu a velocidade de operação, que está sendo melhorada com a evolução dos microcomputadores. Esta evolução está possibilitando cada vez mais que os projetistas de **controladores digitais** cheguem mais próximos de sistemas com desempenho ideal. I. 2 – Sistemas Discretos.

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

Normalizando as palavras

- Em textos da língua portuguesa temos diferentes palavras flexionadas em **gênero**, **número** ou **grau**, além de inúmeros tempos verbais distintos.
 - Trabalho Trabalhadora
 - Degrau Degraus
 - Amigo Amigão
- A “normalização de palavras” pode ser entendida como a **redução** ou a **simplificação** ou a **radicalização** de palavras.
- Duas técnicas mais importantes
 - *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.

Exemplo

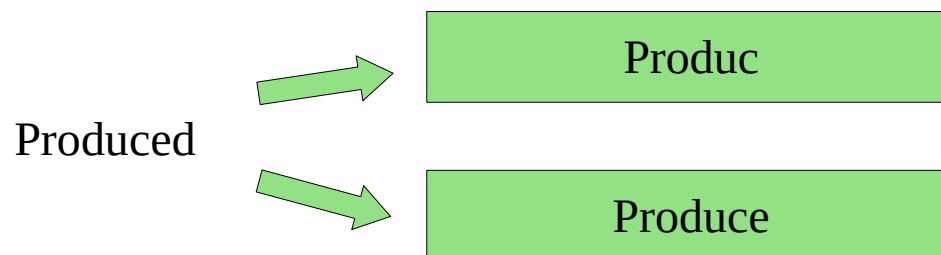
- Computationally
- Computational
- Computation
- Computa
- Comput

← Não confunda, não é a raiz da palavra (da forma linguística)

← A palavra pode não ter significado
(alguns autores denominam isto de **radical** da palavra)

Stemming x Lemmatization

- *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**



Algoritmos de *Stemming*

Existem diferentes algoritmos: Principalmente para o inglês!

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 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

Porter stemmer: 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

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.

- Composto por 294 sufixos, 29 condições e 34 regras de transformação.
- O processamento é rápido: apenas 2 etapas.

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



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 ibility 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
```

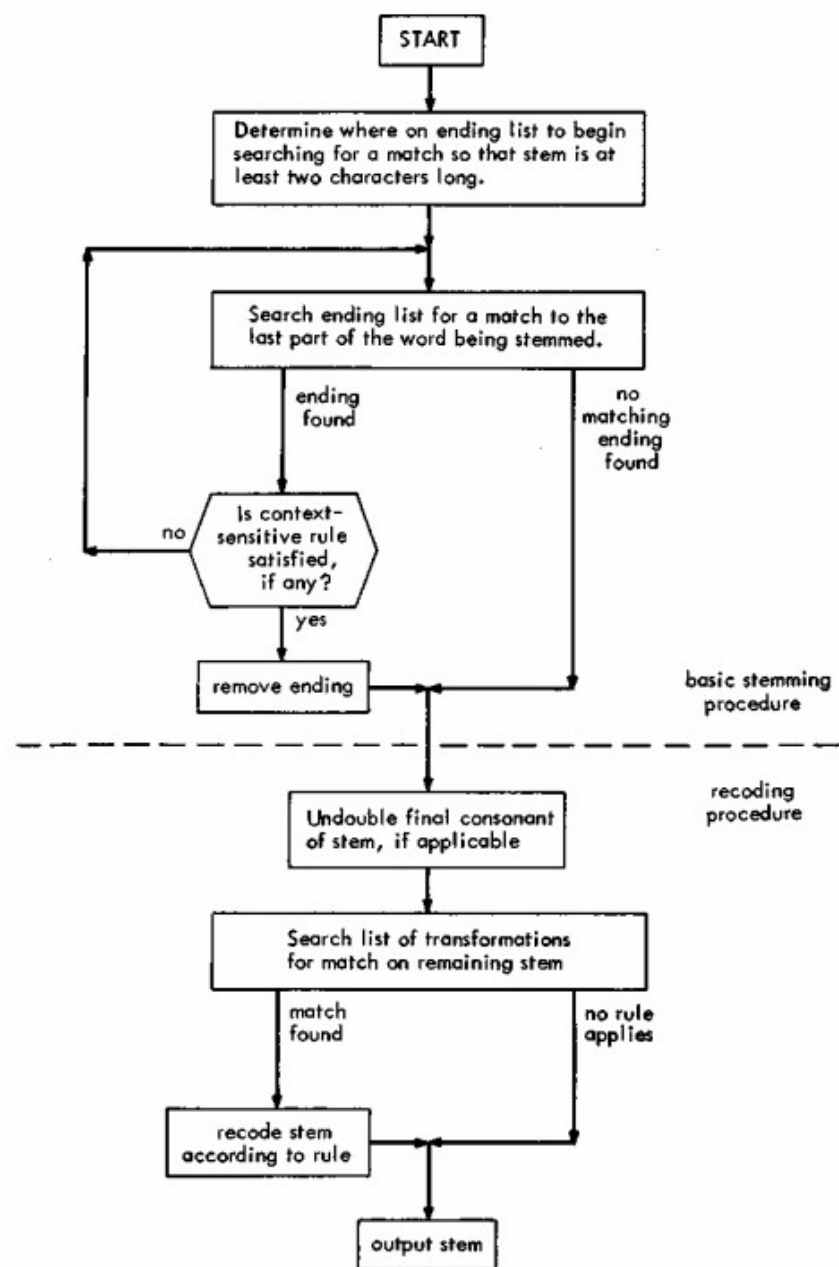


Atividade prática

Atividade prática

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

Algoritmo de Lovins



Input	Initial Stem	Recoded Stem
magnesia	magnes	magnes
magnesite	magnes	magnes
magnesian	magnes	magnes
magnesium	magnes	magnes
magnet	magnet	magnet
magnetic	magnet	magnet
magneto	magnet	magnet
magnetically	magnet	magnet
magnetism	magnet	magnet
magnetite	magnet	magnet
magnetitic	magnet	magnet
magnetizable	magnet	magnet
magnetization	magnet	magnet
magnetize	magnet	magnet
magnetometer	magnetometer	magnetometer
magnetometric	magnetometr	magnetometer
magnetometry	magnetometr	magnetometer
magnetomotive	magnetomot	magnetomot
magneton	magnet	magnet
magnetostriction	magnetostrict	magnetostrict
magnetostrictive	magnetostrict	magnetostrict
magnetron	magnetron	magnetron
metal	metal	metal
metallic	metall	metal
metallically	metall	metal
metalliferous	metallifer	metallifer
metallize	metall	metal
metallurgical	metallurg	metallurg
metallurgy	metallurg	metallurg
induction	induct	induc
inductance	induct	induc
induced	induc	induc
angular	angul	angl
angle	angl	angl

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

- Todos os **5 passos** foram bem definidos:

Usa-se regras e operação de validação

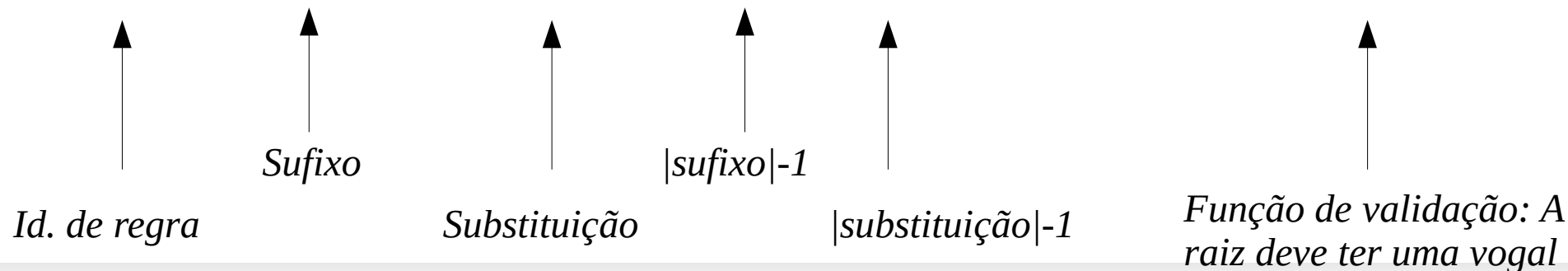
- Similar ao de Lovins: Cada palavra é comparada sequencialmente com o maior sufixo possível.

Casou o sufixo → Remove-lo da palavra.

- **Exemplo:**

Tamanho minimo da raiz (quando eliminado o sufixo) deve ser igual a 3 caracteres

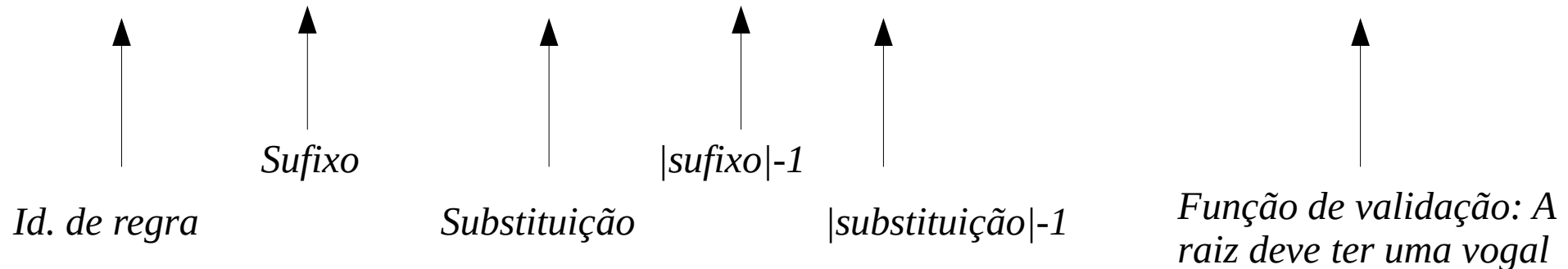
106, "ed", LAMBDA, 1, -1, -1, ContainsVowel



Algoritmo de Martin F. Porter

Tamanho mínimo da raiz (quando eliminado o sufixo) deve ser igual a 3 caracteres

106, "ed", LAMBDA, 1, -1, -1, ContainsVowel



■ 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

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

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.vowelinstem:
        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
```

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 Porter

<i>language</i>	<i>author</i>	<i>affiliation</i>	<i>received</i>	<i>notes</i>
ANSI C	me			
ANSI C thread safe	me			
java	me			
Perl	me			
Perl	Daniel van Balen		Oct 1999	slightly faster?
python	Vivake Gupta		Jan 2001	
Csharp	André Hazelwood	The Official Web Guide	Sep 2001	
Csharp .NET compliant	Leif Azzopardi	University of Paisley, Scotland	Nov 2002	
Csharp again!	Brad Patton	ratborg.blogspot.com	Dec 2015	"more like standard C# code" (Brad)
Common Lisp	Steven M. Haflich	Franz Inc	Mar 2002	
Ruby	Ray Pereda	www.raypereda.com	Jan 2003	github link
Visual Basic VB6	Navonil Mustafee	Brunel University	Apr 2003	
Delphi	Jo Rabin		Apr 2004	
Javascript	'Andargor'	www.andargor.com	Jul 2004	substantial revisions by Christopher McKenzie
Visual Basic VB7; .NET compliant	Christos Attikos	University of Piraeus, Greece	Jan 2005	
php	Richard Heyes	www.phpguru.org	Feb 2005	
Prolog	Philip Brooks	University of Georgia	Oct 2005	
Haskell	Dmitry Antonyuk		Nov 2005	
T-SQL	Keith Lubell	www.atelierdevitraux.com	May 2006	
matlab	Juan Carlos Lopez	California Pacific Medical Center Research Institute	Sep 2006	
Tcl	Aris Theodorakos	NCSR Demokritos	Nov 2006	
D	Daniel Truemper	Humboldt-Universitaet zu Berlin	May 2007	
erlang (1) erlang (2)	Alden Dima	National Institute of Standards and Technology, Gaithersburg, MD USA	Sep 2007	
REBOL	Dale K Brearcliffe		Apr 2009	
Scala	Ken Faulkner		May 2009	
sas	Antoine St-Pierre	Business Researchers, Inc	Apr 2010	
plugin vim script	Mitchell Bowden		May 2010	github link
node.js	Jed Parsons	jedparsons.com	May 2011	github link
Google Go	Alex Gonopolskiy		Oct 2011	github link
awk	Gregory Grefenstette	3ds.com/exalead	Jul 2012	
clojure	Yushi Wang		Mar 2013	bitbucket link
Rust	Do Nhat Minh	Nanyang Technological University	Aug 2013	github link
vala	Serge Hulne		Sep 2013	
MySQL	John Carty	Enlighten Jobs	Jan 2015	github link
Julia	Matias Guzmán Naranjo		May 2015	github link
flex	Zalán Bodó	Babes-Bolyai University	Oct 2015	(Zalan's notes)
R	Mohit Makkar	Indian Institute of Technology, Delhi	Nov 2015	
Groovy	Dhaval Dave		June 2016	github link
ooRexx	P.O. Jonsson		July 2016	sourceforge link

Algoritmos de *Stemming* para Português

- 2001: Orenco

Orenco, 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.

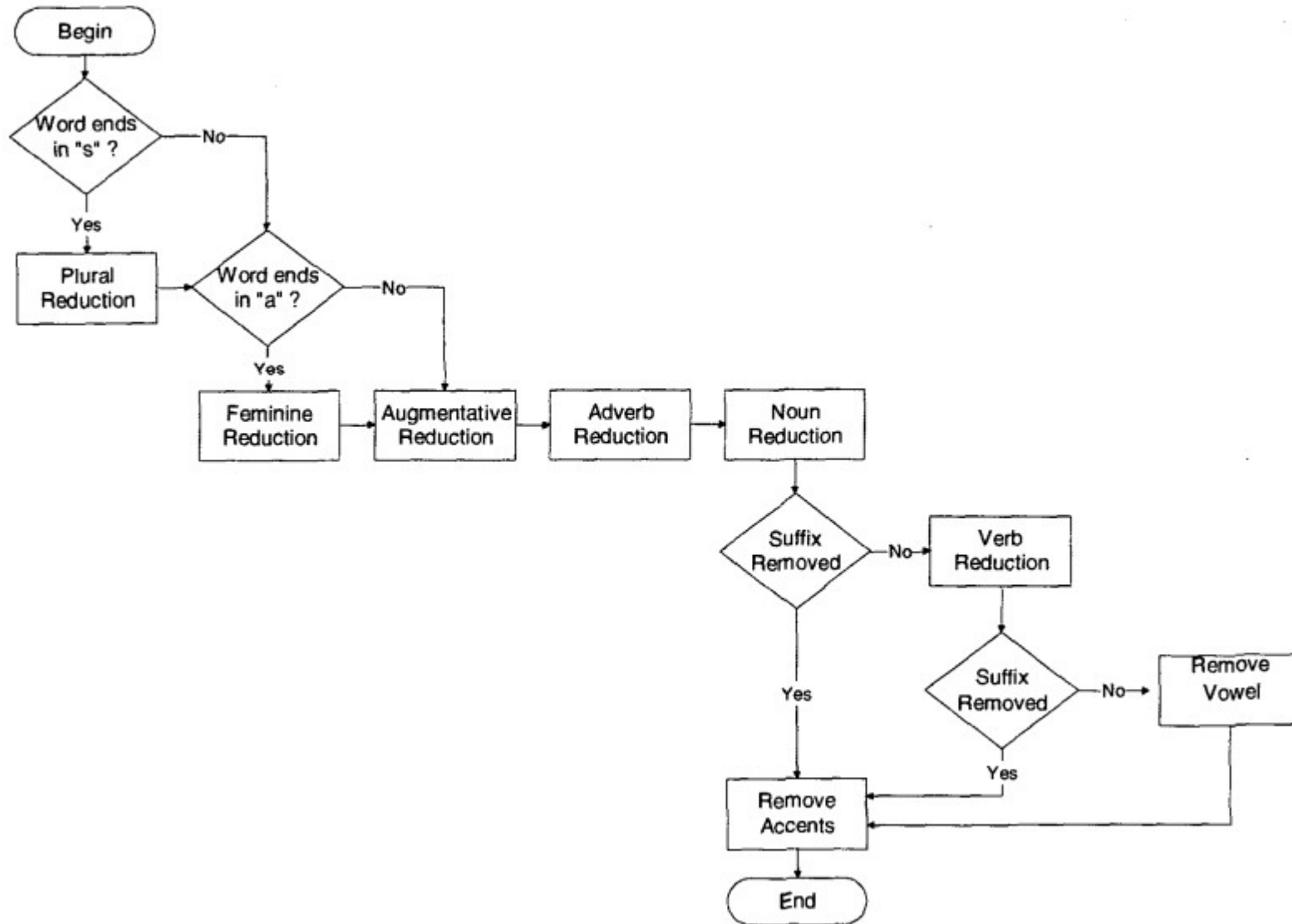
Primeira versão amplamente divulgada de um algoritmo de radicalização para a língua portuguesa:

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



Middlesex University
UFRGS

Algoritmo de Orengo



Algoritmo de Orengo

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

Outras propostas

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 <i>et al.</i>	Schinke <i>et al.</i>
Norueguês	Porter - Norueguês	Porter
	Carlberger <i>et al.</i>	Carlberger <i>et al.</i>
Russo	Porter - Russo	Porter
Sueco	Porter - Sueco	Porter
Turco	Ekmekçioğlu <i>et al.</i>	Ekmekçioğlu <i>et al.</i>

Recursos: Python - NLTK

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

Stemming and Lemmatization with Python NLTK

This is a demonstration of **stemming** and **lemmatization** for the 17 languages supported by the **NLTK 2.0.4 stem** package.

Stem Text

Choose stemmer
Portuguese RSLP ▼

Enter text

Não é isto uma sátira em prosa.
Esboço literário apanhado nas
projeções
sútils dos caracteres, dou aqui
apenas uma reprodução do tipo a que
chamo
em meu falar seco de prosador
novato - fanqueiro literário.

A fancaria literária é a pior de

Enter up to 50000 characters

Stem

Stemmed Text

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 . a fanc literár é a pi de
tod as fanc . é a obr gross , por vez mof , que se acomod à
ondul da espádu do paci fregu . há de tud ness loj manufa do
talent — apes da raridad da tel fin ; e as vaidad soc mais
exig pod vaz - se , segund as sua aspir , em uma ode ou
discurs parv retumb . a fanc literár pod perd pel eleg suspeit
da roup feit , mas nunc pel exigü do gêner . tom a tabulet por
bas do silog comerc é infal cheg log à propos men , que é a
pratel guap atac a faz cobiç às modést mais insuspeit . é lind
comérci . desd josé daniel , o apóstol da cl — ess mod de vid
tem alarg a sua esf — e , por mal de pec , não promet fic aqu
. o fanc liter é um tip curi . fal em josé daniel . conhecel ess
vult histór ? era uma excel organiz que se prest perfeít a
autóps . adel ambul da intelig , ia fart com um ovo , de feir
em feir , troc pel enzinavr moed o prat enfez de sua lucubr



Comparação entre palavras únicas:

- Sem radicalização**
- Com radicalização**

Comparando resultados

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 .

Comparando resultados

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.



Algumas considerações não discutidas sobre os stemmers!

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.

2) Por que não eliminar prefixos?

- Não ha nenhum motivo teórico para não considerar a eliminação de prefixos nos stemmers:
 - **Arquiduque**
 - **Protótipo**
 - **Contradizer**
 - **Ultraleve**

3) *stemming* na literatura

Google Books Ngram Viewer

Graph these comma-separated phrases: case-insensitive
between and from the corpus with smoothing of [Search lots of books](#)



4) 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

4) 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

5) Stemmer para nomes?

Ideia para um projeto final?

- Pedrão → Pedro
- Guilherme → Gui
- Roberto → Rob



Stemmer - práctica

fool-stemmer.py

```
2 import sys
3
4 if __name__ == '__main__':
5     if len(sys.argv) > 1:
6
7         for f in sys.argv[1:]:
8             infile = open(f, 'r')
9
10            while 1:
11                output = ""
12                word = ""
13                line = infile.readline()
14
15                if line == "":
16                    break
17                for c in line:
18                    if c.isalpha():
19                        word += c.lower()
20                    else:
21                        if word:
22                            output += word[0:3]
23                            word = ""
24                        output += c.lower()
25                print (output, end=' ')
26            infile.close()
```

fool-stemmer

a capivara (nome científico: *hydrochoerus hydrochaeris*) é uma espécie de mamífero roedor da família *caviidae* e subfamília *hydrochoerinae*. alguns autores consideram que deva ser classificada em uma família própria. está incluída no mesmo grupo de roedores ao qual se classificam as pacas, cutias, os preás e o porquinho-da-índia. ocorre por toda a américa do sul ao leste dos andes em habitats associados a rios, lagos e pântanos, do nível do mar até 1 300 m de altitude. extremamente adaptável, pode ocorrer em ambientes altamente alterados pelo ser humano.

é o maior roedor do mundo, pesando até 91 kg e medindo até 1,2 m de comprimento e 60 cm de altura. a pelagem é densa, de cor avermelhada a marrom escuro. é possível distinguir os machos por conta da presença de uma glândula proeminente no focinho apesar do dimorfismo sexual não ser aparente. existe uma série de adaptações no sistema digestório à herbivoria, principalmente no ceco. alcança a maturidade sexual com cerca de 1,5 ano de idade, e as fêmeas dão à luz geralmente a quatro filhotes por vez, pesando até 1,5 kg e já nascem com pelos e dentição permanente. em cativeiro, pode viver até 12 anos de idade.

```
python3 fool-stemmer.py capivara-pt.txt
```

fool-stemmer

a cap (nom cie: hyd hyd) é uma esp de mam roe da fam cav e sub hyd. alg aut con que dev ser cla em uma fam pró. est inc no mes gru de roe ao qua se cla as pac, cut, os pre e o por-da-índ. oco por tod a amé do sul ao les dos and em hab ass a rio, lag e pân, do ní v do mar até 1 300 m de alt. ext ada, pod oco em amb alt alt pel ser hum.

é o mai roe do mun, pes até 91 kg e med até 1,2 m de com e 60 cm de alt. a pel é den, de cor ave a mar esc. é pos dis os mac por con da pre de uma glâ pro no foc ape do dim sex não ser apa. exi uma sér de ada no sis dig à her, pri no cec. alc a mat sex com cer de 1,5 ano de ida, e as fêm dão à luz ger a qua fil por vez, pes até 1,5 kg e já nas com pel e den per. em cat, pod viv até 12 ano de ida.

fool-stemmer

<https://text-compare.com/>

1 a capivara (nome científico: hydrochoerus hydrochaeris) é uma espécie de
2 mamífero roedor da família caviidae e subfamília hydrochoerinae. alguns autores
3 consideram que deva ser classificada em uma família própria. está incluída no
4 mesmo grupo de roedores ao qual se classificam as pacas, cutias, os preás e o
5 porquinho-da-índia. ocorre por toda a américa do sul ao leste dos andes em
6 habitats associados a rios, lagos e pântanos, do nível do mar até 1 300 m de
7 altitude. extremamente adaptável, pode ocorrer em ambientes altamente alterados
8 pelo ser humano.

9
10 é o maior roedor do mundo, pesando até 91 kg e medindo até 1,2 m de comprimento
11 e 60 cm de altura. a pelagem é densa, de cor avermelhada a marrom escuro. é
12 possível distinguir os machos por conta da presença de uma glândula proeminente
13 no focinho apesar do dimorfismo sexual não ser aparente. existe uma série de
14 adaptações no sistema digestório à herbivoria, principalmente no ceco. alcança
15 a maturidade sexual com cerca de 1,5 ano de idade, e as fêmeas dão à luz
16 geralmente a quatro filhotes por vez, pesando até 1,5 kg e já nascem com pelos
17 e dentição permanente. em cativeiro, pode viver até 12 anos de idade.

Over-stemming

- Quando é removido não só o sufixo, mas também uma parte do radical (**2 palavras são reduzidas a um mesmo radical**).

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 Porter

<i>language</i>	<i>author</i>	<i>affiliation</i>	<i>received</i>	<i>notes</i>
ANSI C	me			
ANSI C thread safe	me			
java	me			
Perl	me			
Perl	Daniel van Balen		Oct 1999	slightly faster?
python	Vivake Gupta		Jan 2001	
Csharp	André Hazelwood	The Official Web Guide	Sep 2001	
Csharp .NET compliant	Leif Azzopardi	University of Paisley, Scotland	Nov 2002	
Csharp again!	Brad Patton	ratborg.blogspot.com	Dec 2015	"more like standard C# code" (Brad)
Common Lisp	Steven M. Haflich	Franz Inc	Mar 2002	
Ruby	Ray Pereda	www.raypereda.com	Jan 2003	github link
Visual Basic VB6	Navonil Mustafee	Brunel University	Apr 2003	
Delphi	Jo Rabin		Apr 2004	
Javascript	'Andargor'	www.andargor.com	Jul 2004	substantial revisions by Christopher McKenzie
Visual Basic VB7; .NET compliant	Christos Attikos	University of Piraeus, Greece	Jan 2005	
php	Richard Heyes	www.phpguru.org	Feb 2005	
Prolog	Philip Brooks	University of Georgia	Oct 2005	
Haskell	Dmitry Antonyuk		Nov 2005	
T-SQL	Keith Lubell	www.atelierdevitraux.com	May 2006	
matlab	Juan Carlos Lopez	California Pacific Medical Center Research Institute	Sep 2006	
Tcl	Aris Theodorakos	NCSR Demokritos	Nov 2006	
D	Daniel Truemper	Humboldt-Universitaet zu Berlin	May 2007	
erlang (1) erlang (2)	Alden Dima	National Institute of Standards and Technology, Gaithersburg, MD USA	Sep 2007	
REBOL	Dale K Brearcliffe		Apr 2009	
Scala	Ken Faulkner		May 2009	
sas	Antoine St-Pierre	Business Researchers, Inc	Apr 2010	
plugin vim script	Mitchell Bowden		May 2010	github link
node.js	Jed Parsons	jedparsons.com	May 2011	github link
Google Go	Alex Gonopolskiy		Oct 2011	github link
awk	Gregory Grefenstette	3ds.com/exalead	Jul 2012	
clojure	Yushi Wang		Mar 2013	bitbucket link
Rust	Do Nhat Minh	Nanyang Technological University	Aug 2013	github link
vala	Serge Hulne		Sep 2013	
MySQL	John Carty	Enlighten Jobs	Jan 2015	github link
Julia	Matias Guzmán Naranjo		May 2015	github link
flex	Zalán Bodó	Babes-Bolyai University	Oct 2015	(Zalan's notes)
R	Mohit Makkar	Indian Institute of Technology, Delhi	Nov 2015	
Groovy	Dhaval Dave		June 2016	github link
ooRexx	P.O. Jonsson		July 2016	sourceforge link

porter-stemmer.py

```
if __name__ == '__main__':
    p = PorterStemmer()
    if len(sys.argv) > 1:
        for f in sys.argv[1:]:
            infile = open(f, 'r')
            while 1:
                output = ""
                word = ""
                line = infile.readline()
                if line == "":
                    break
                for c in line:
                    if c.isalpha():
                        word += c.lower()
                    else:
                        if word:
                            output += p.stem(word, 0, len(word)-1)
                            word = ""
                        output += c.lower()
                print (output, end=' ')
            infile.close()
```


porter-stemmer.py

```
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]
```

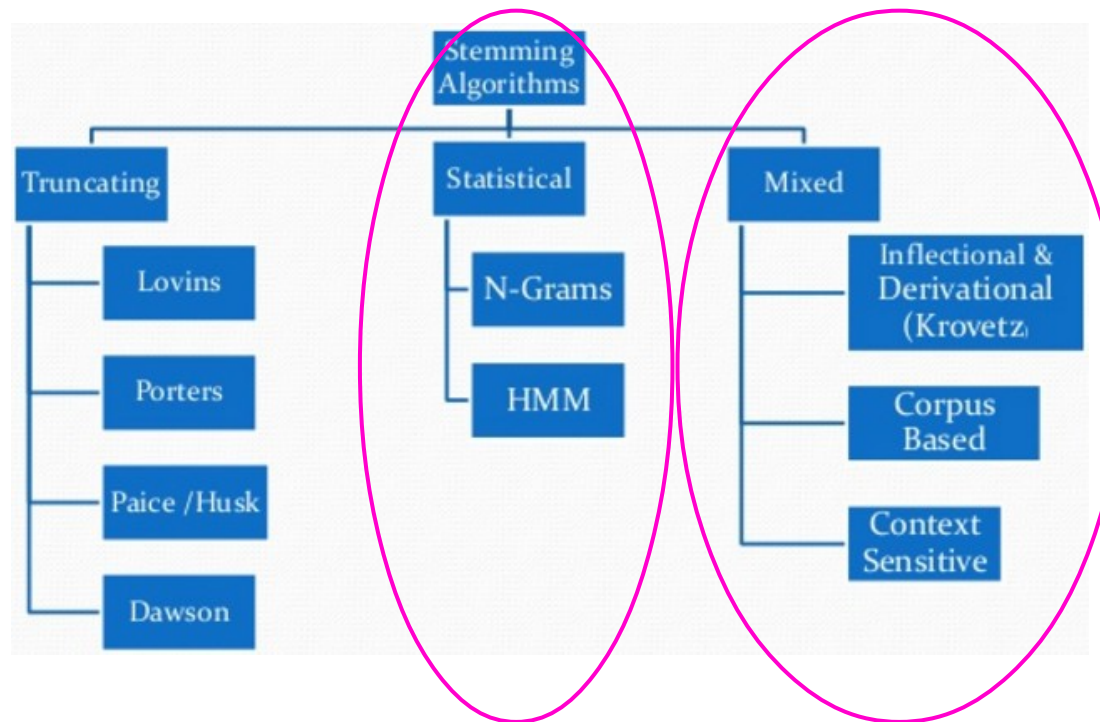
Prática


```
python3 porter-stemmer.py The-Iliad-of-Homer.txt > \
The-Iliad-of-Homer.txt.porter.txt
```

```
python3 fool-stemmer.py The-Iliad-of-Homer.txt > \
The-Iliad-of-Homer.txt.fool.txt
```

Para o **resumo**: Por favor, responda brevemente.
No contexto de algoritmos de stemming, o que é **Snowball**?

Outras abordagens para stemming?





Uma aplicação: Rede (grafo) de palavras

Rede (grafo) de palavras: graph1.py

```
if __name__ == '__main__':
    fileName = sys.argv[1]
    weight = int(sys.argv[2])

    document = open(fileName, 'r')
    content = document.read() # devolve o conteudo do arquivo

    Words = re.findall(regex, content)
    Edges = dict([])

    # contando a frequencia dos pares de palavras
    for i in range(0, len(Words)-1):
        edge = (Words[i], Words[i+1])
        if edge not in Edges:
            Edges[edge] = 0
        Edges[edge] += 1

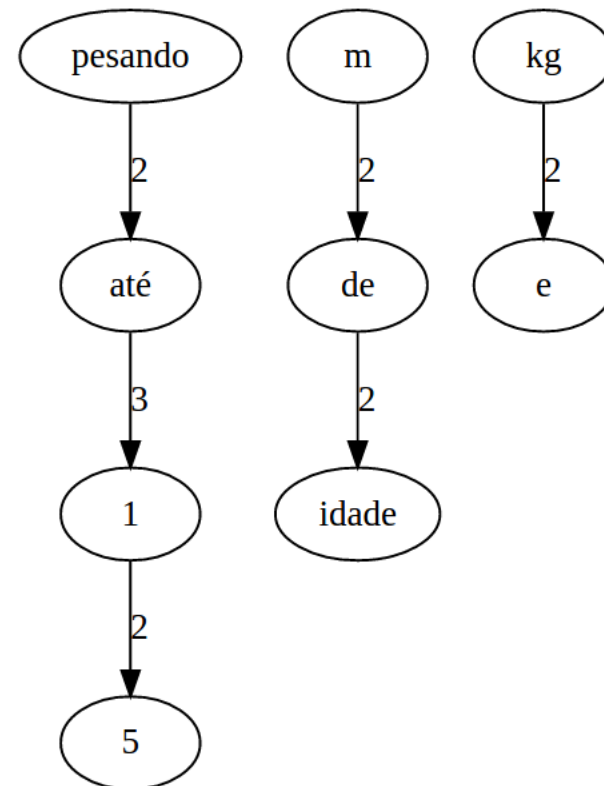
    # criando o grafo direcionado (digraph)
    txtGraph = "\ndigraph{"
    for v in Edges.keys():
        if Edges[v] >= weight:
            txtGraph += '\n "{}" -> "{}"[label="{}"]'.format(v[0], v[1], Edges[v])
    txtGraph += "\n}"

    print(txtGraph)
    print ( "\nQuantidade de palavras: {}".format(len(Words)) )
    print ( "Quantidade de arestas : {}".format(len(Edges)) )
```

Prática 1

```
python3 graph1.py capivara-pt.txt 2
```

```
digraph{
  "pesando" -> "até" [label="2"]
  "até" -> "1" [label="3"]
  "de" -> "idade" [label="2"]
  "1" -> "5" [label="2"]
  "kg" -> "e" [label="2"]
  "m" -> "de" [label="2"]
}
```

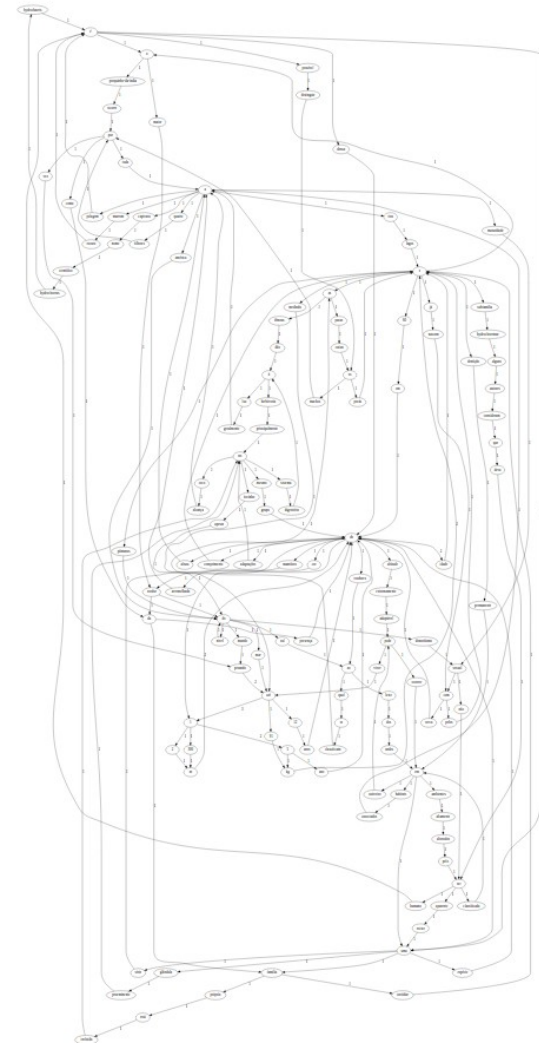


Duas palavras estão ligadas se ambas estão adjacentes em uma frase.

Prática 2

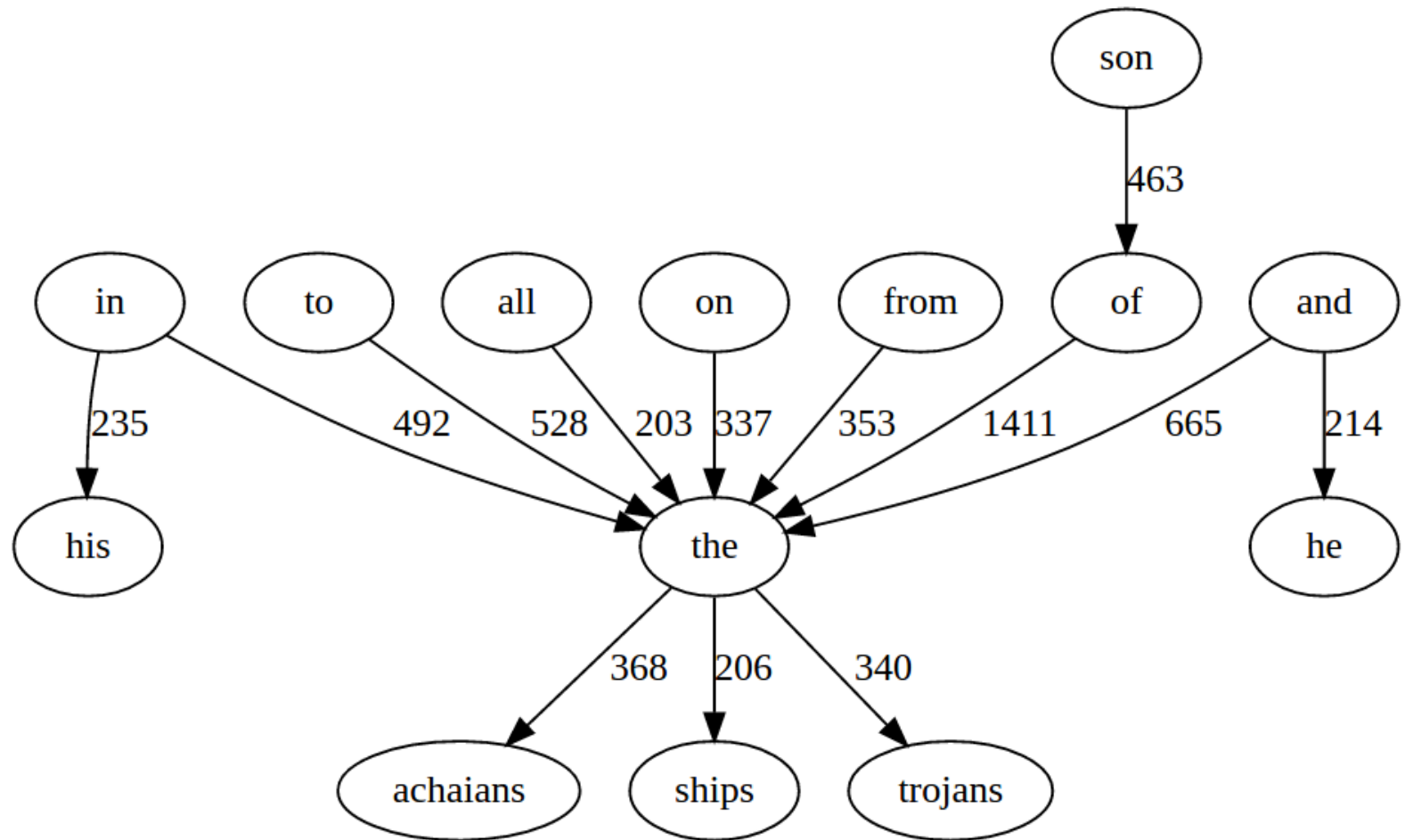
```
1 digraph{
2 "hydrochaeris" -> "é"[label="1"]
3 "o" -> "porquinho-da-índia"[label="1"]
4 "alguns" -> "autores"[label="1"]
5 "nome" -> "científico"[label="1"]
6 "1" -> "300"[label="1"]
7 "altitude" -> "extremamente"[label="1"]
8 "que" -> "deva"[label="1"]
9 "os" -> "machos"[label="1"]
10 "é" -> "uma"[label="1"]
11 "a" -> "rios"[label="1"]
12 "de" -> "altitude"[label="1"]
13 "em" -> "ambientes"[label="1"]
14 "apesar" -> "do"[label="1"]
15 "no" -> "mesmo"[label="1"]
16 "toda" -> "a"[label="1"]
17 "ao" -> "leste"[label="1"]
18 "kg" -> "e"[label="2"]
19 "até" -> "12"[label="1"]
20 "altura" -> "a"[label="1"]
21 "por" -> "vez"[label="1"]
22 "família" -> "própria"[label="1"]
23 "pelagem" -> "é"[label="1"]
24 "ocorrer" -> "em"[label="1"]
25 "ser" -> "classificada"[label="1"]
26 "a" -> "quatro"[label="1"]
27 "está" -> "incluída"[label="1"]
28 "do" -> "dimorfismo"[label="1"]
29 "lagos" -> "e"[label="1"]
30 "a" -> "maturidade"[label="1"]
31 "do" -> "mundo"[label="1"]
32 "quatro" -> "filhotes"[label="1"]
33 "ser" -> "humano"[label="1"]
34 "viver" -> "até"[label="1"]
35 "por" -> "conta"[label="1"]
36 "porquinho-da-índia" -> "ocorre"[label="1"]
37 "permanente" -> "em"[label="1"]
38 "dos" -> "andes"[label="1"]
39 "o" -> "maior"[label="1"]
40 "mamífero" -> "roedor"[label="1"]
41 "preás" -> "e"[label="1"]
42 "cativeiro" -> "pode"[label="1"]
43 "os" -> "preás"[label="1"]
44 "uma" -> "série"[label="1"]
45 "se" -> "classificam"[label="1"]
46 "ao" -> "qual"[label="1"]
```

Engine: dot Format: svg Show raw output



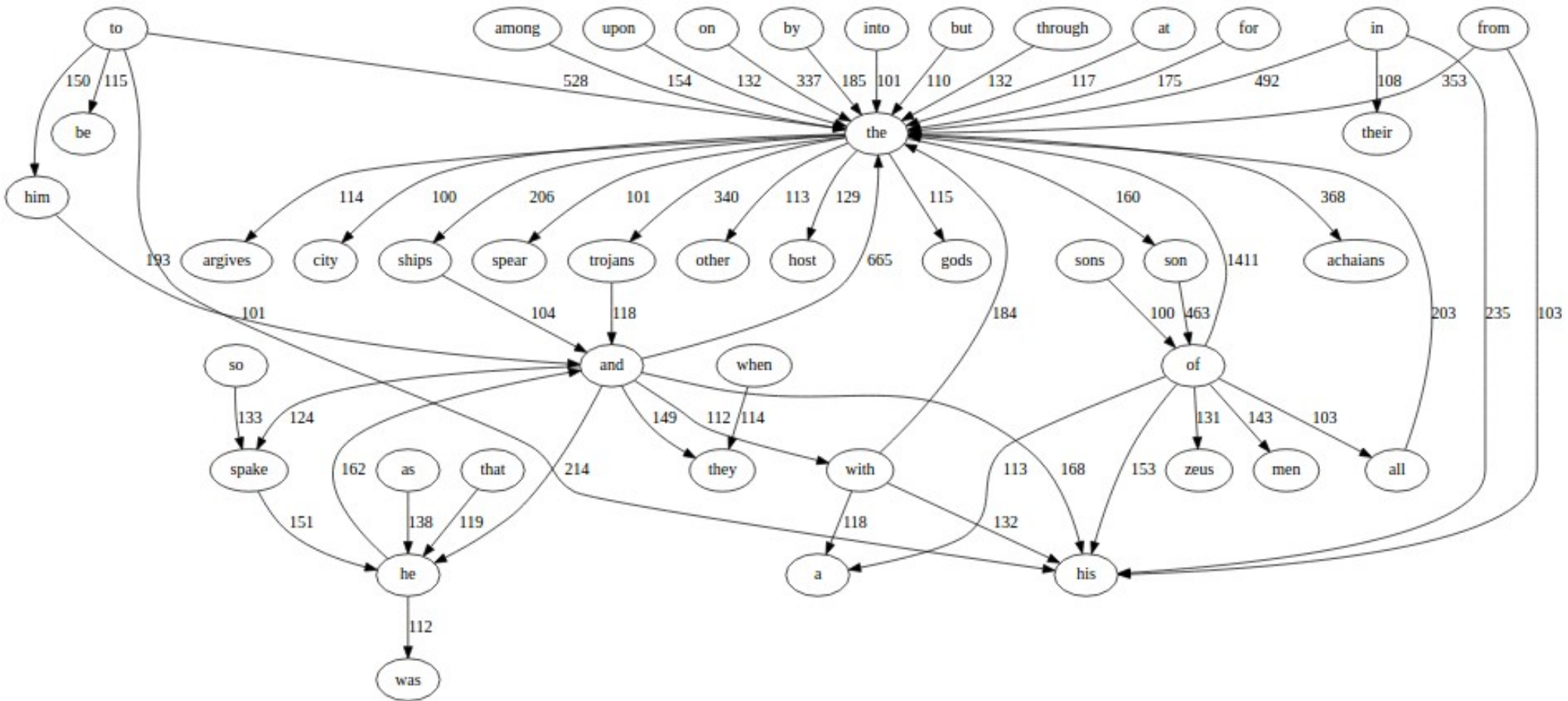
python3 graph1.py capivara-pt.txt 1

Prática 3



python3 graph1.py The-Iliad-of-Homer.txt 200

Prática 4



python3 graph1.py The-Iliad-of-Homer.txt 100

Prática 5

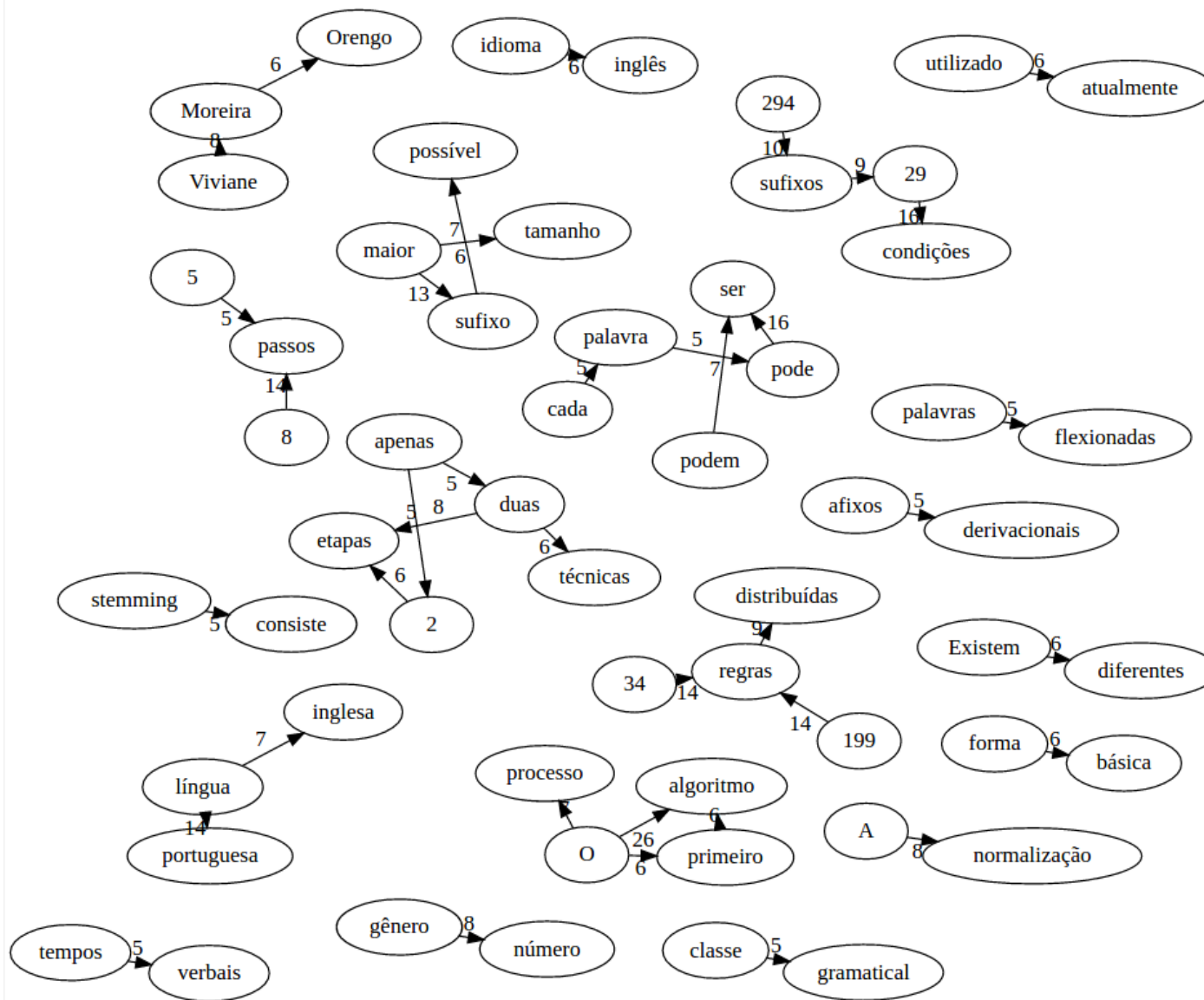
```
# leitura das stopwords
stopwordsfile = open("stopwords.txt", 'r')
stopwords     = set([])
for s in stopwordsfile.readlines():
    stopwords.add(s.strip().lower())

# leitura do documento
document = open(fileName, 'r')
content  = document.read() # devolve o conteúdo do arquivo

Words    = re.findall(regex, content)
Edges    = dict([])

# contando a frequência dos pares de palavras
for i in range(0, len(Words)-1):
    if Words[i] not in stopwords and Words[i+1] not in stopwords:
        edge = (Words[i], Words[i+1])
        if edge not in Edges:
            Edges[edge] = 0
        Edges[edge] += 1
```

Prática 6

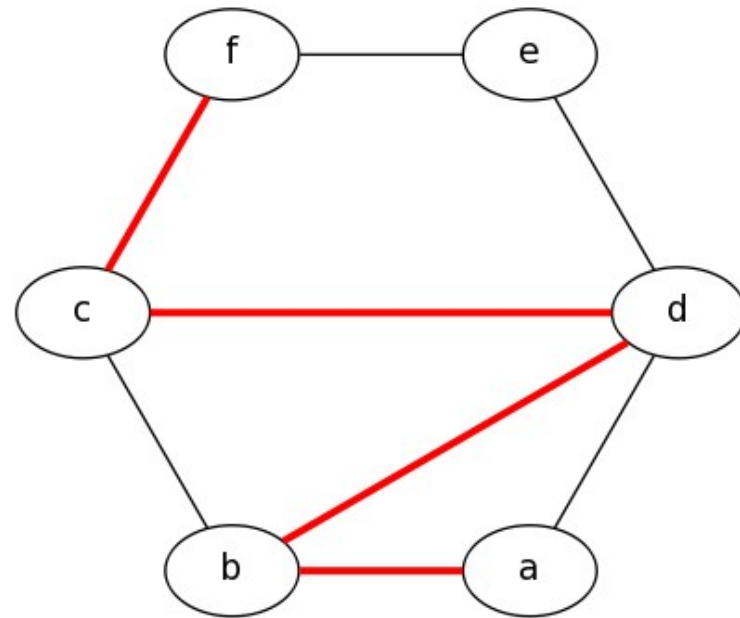


python3 graph2.py resumos-aula-04.txt 5

Mais sobre visualização (graphviz)

<http://graphs.grevian.org/example>

```
graph {  
  a -- b[color=red,penwidth=3.0];  
  b -- c;  
  c -- d[color=red,penwidth=3.0];  
  d -- e;  
  e -- f;  
  a -- d;  
  b -- d[color=red,penwidth=3.0];  
  c -- f[color=red,penwidth=3.0];  
}
```



Note that there's also a shorthand method as follows:

```
graph {  
  a -- b -- d -- c -- f[color=red,penwidth=3.0];  
  b -- c;  
  d -- e;  
  e -- f;  
  a -- d;  
}
```

Redes (Grafos) de co-ocorrência de palavras

- Permitem identificar, de forma visual, relações potenciais entre elementos (palavras)?
- Podemos considerar propriedades topológicas nos grafos para descoberta de conhecimento a partir de textos?
- Podemos estudar a organização das palavras?

