MCZA017-13
Processamento de Linguagem Natural

Normalização de texto: Palavras e stopwords

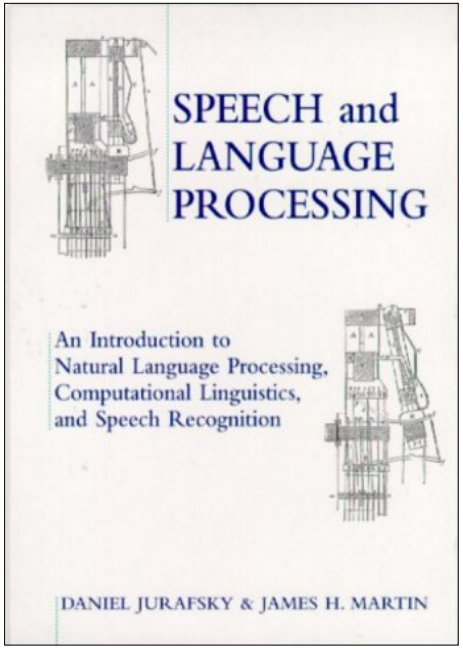
Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

2Q-2019

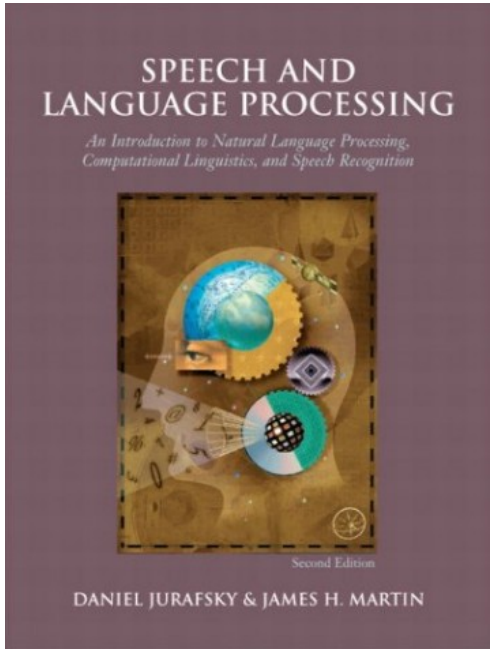
Bibliografia

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

Bibliografia – Capítulo 2

Speech and Language Processing (3rd ed. draft)

Dan Jurafsky and James H. Martin

	Chapter	Slides	Relation to 2nd ed.
1:	Introduction		[Ch. 1 in 2nd ed.]
2:	Regular Expressions, Text Normalization, and Edit Distance	Text [pptx] [pdf] Edit Distance [pptx] [pdf]	[Ch. 2 and parts of Ch. 3 in 2nd ed.]
3:	Finite State Transducers		
4:	Language Modeling with N-Grams	LM [pptx] [pdf]	[Ch. 4 in 2nd ed.]
5:	Spelling Correction and the Noisy Channel	Spelling [pptx] [pdf]	[expanded from pieces in Ch. 5 in 2nd ed.]
6:	Naive Bayes Classification and Sentiment	NB [pptx] [pdf] Sentiment [pptx] [pdf]	[new in this edition]
7:	Logistic Regression		
8:	Neural Nets and Neural Language Models		
9:	Hidden Markov Models		[Ch. 6 in 2nd ed.]
9b:	Neural Sequence Modeling: RNNs and LSTMs		
10:	Part-of-Speech Tagging		[Ch. 5 in 2nd ed.]
11:	Formal Grammars of English		[Ch. 12 in 2nd ed.]
12:	Syntactic Parsing		[Ch. 13 in 2nd ed.]
13:	Statistical Parsing		[Ch. 14 in 2nd ed.]
14:	Dependency Parsing		[new in this edition]
15:	Vector Semantics	Vector [pptx] [pdf]	[expanded from parts of Ch. 19 and 20 in 2nd ed.]
16:	Semantics with Dense Vectors	Dense Vector [pptx] [pdf]	[new in this edition]
17:	Computing with Word Senses: WSD and WordNet	Intro, Sim [pptx] [pdf] WSD [pptx] [pdf]	[expanded from parts of Ch. 19 and 20 in 2nd ed.]
18:	Lexicons for Sentiment and Affect Extraction	SentLex [pptx] [pdf]	[new in this edition]

https://web.stanford.edu/~jurafsky/slp3/

Da aula anterior... Expressões Regulares

Teste de avaliação em aula (questão 6)

6. Selecionar todas as cadeias com duas palavras repetidas consecutivas. Exemplo: No seguinte fragmento, apenas os trechos sublinhados devem ser identificados. Esperar menos menos não significa desistir desistir. Antes se surpreender, do que, que se decepcionar. Clarice Lispector	\b(\w+)\W+\1

Grupos

Cada sub-expressão entre parênteses em uma ER é um grupo:

- Grupo 0: O casamento inteiro.
- Grupo 1: O primeiro casamento entre parênteses.
  → Numeraçao de esquerda para direita.

Grupos

REGULAR EXPRESSION

```
/ ((\w+)\.(\w+))@\2
```

TEST STRING

```
joao.costa@email.com
joao.costa@joao.costa.com
joao.costa@joao.com
joao.costa@costa.com
```

Full match	67-83	`joao.costa@costa`
Group 1.	67-77	`joao.costa`
Group 2.	67-71	`joao`
Group 3.	72-77	`costa`

REGULAR EXPRESSION

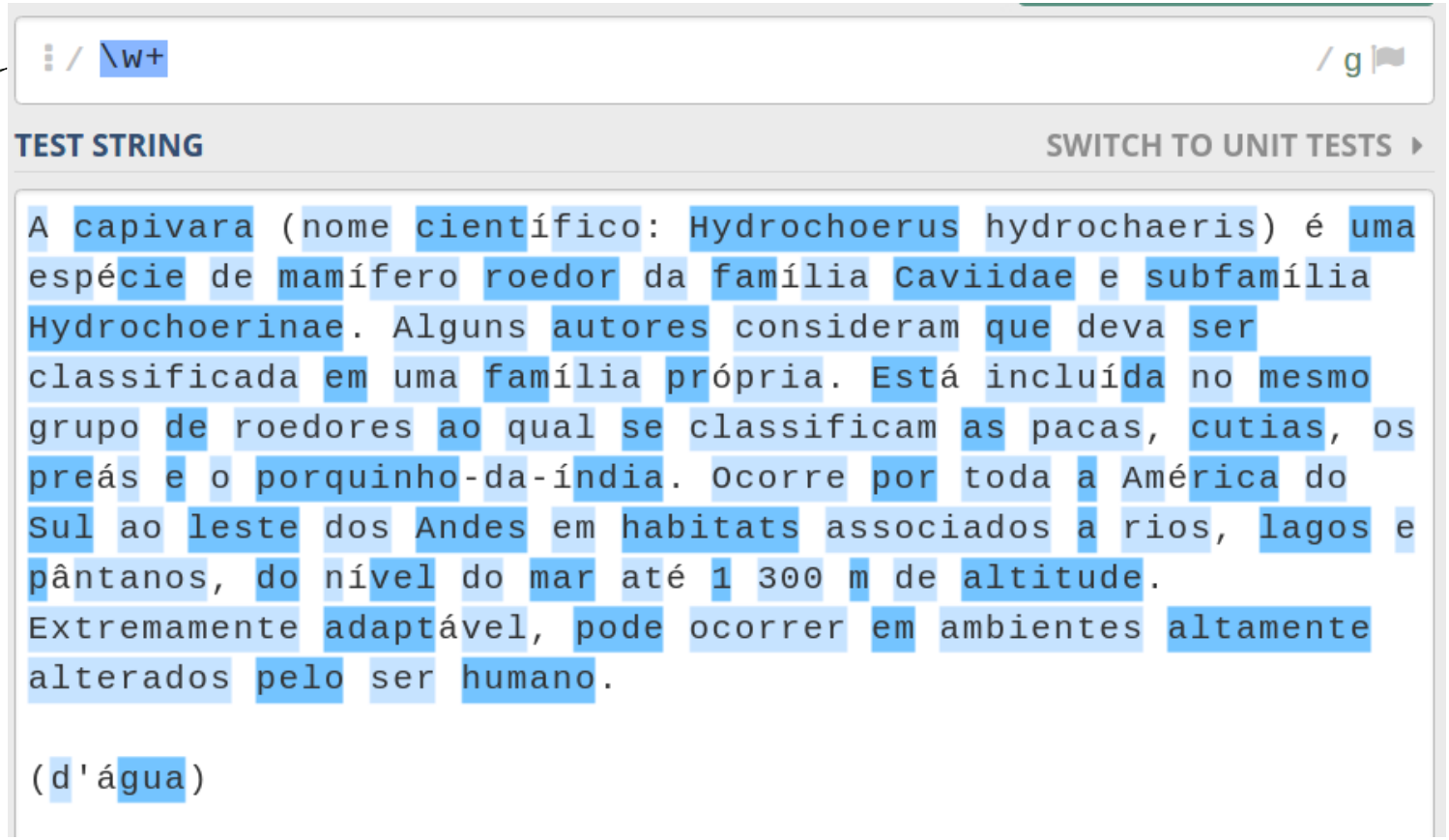
```
/ ((\w+)\.(\w+))@\3
```

TEST STRING

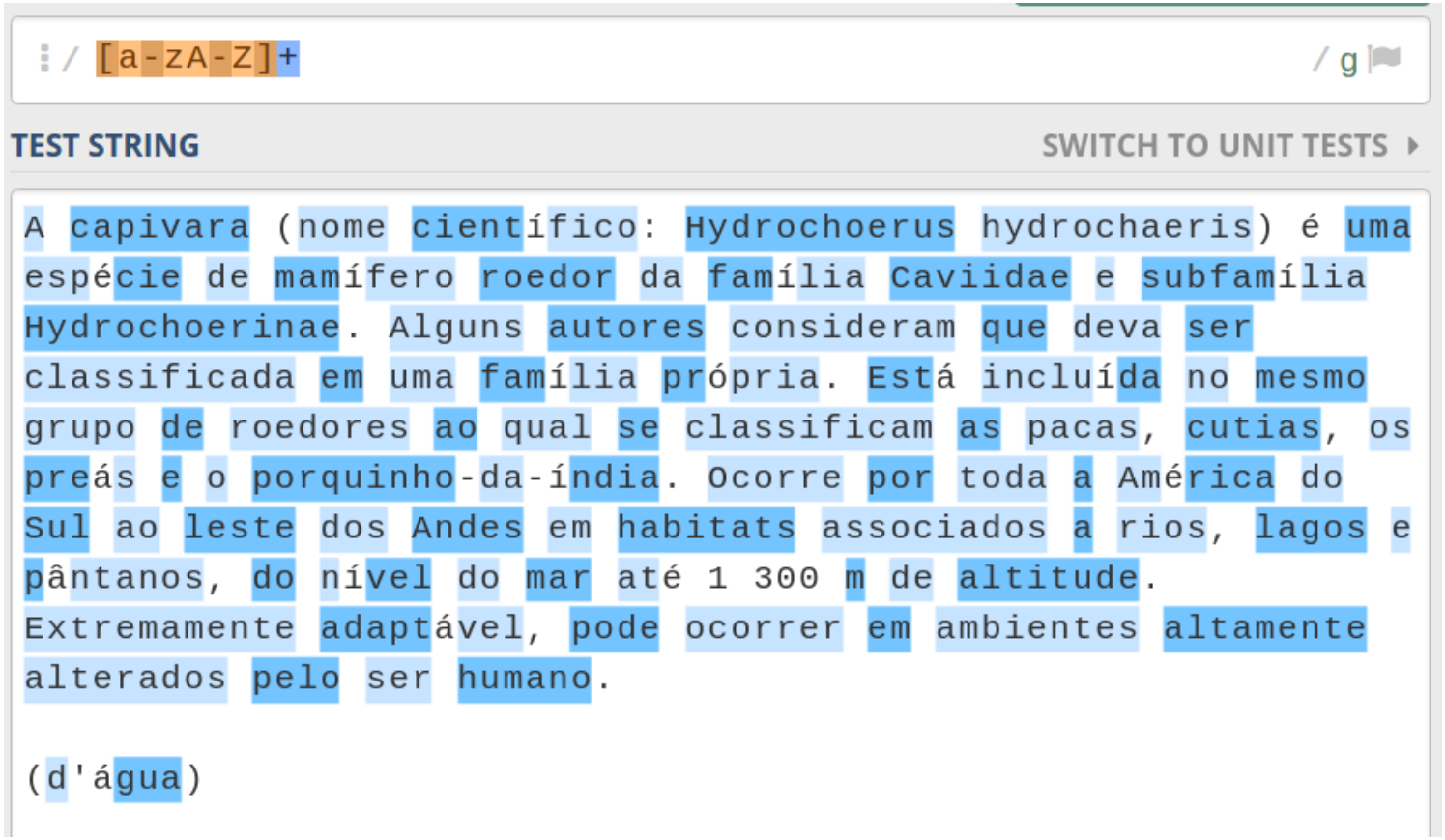
```
joao.costa@email.com
joao.costa@joao.costa.com
joao.costa@joao.com
joao.costa@costa.com
```

ER para palavras

Any alphanumeric or underscore



ER para palavras



ER para palavras em português

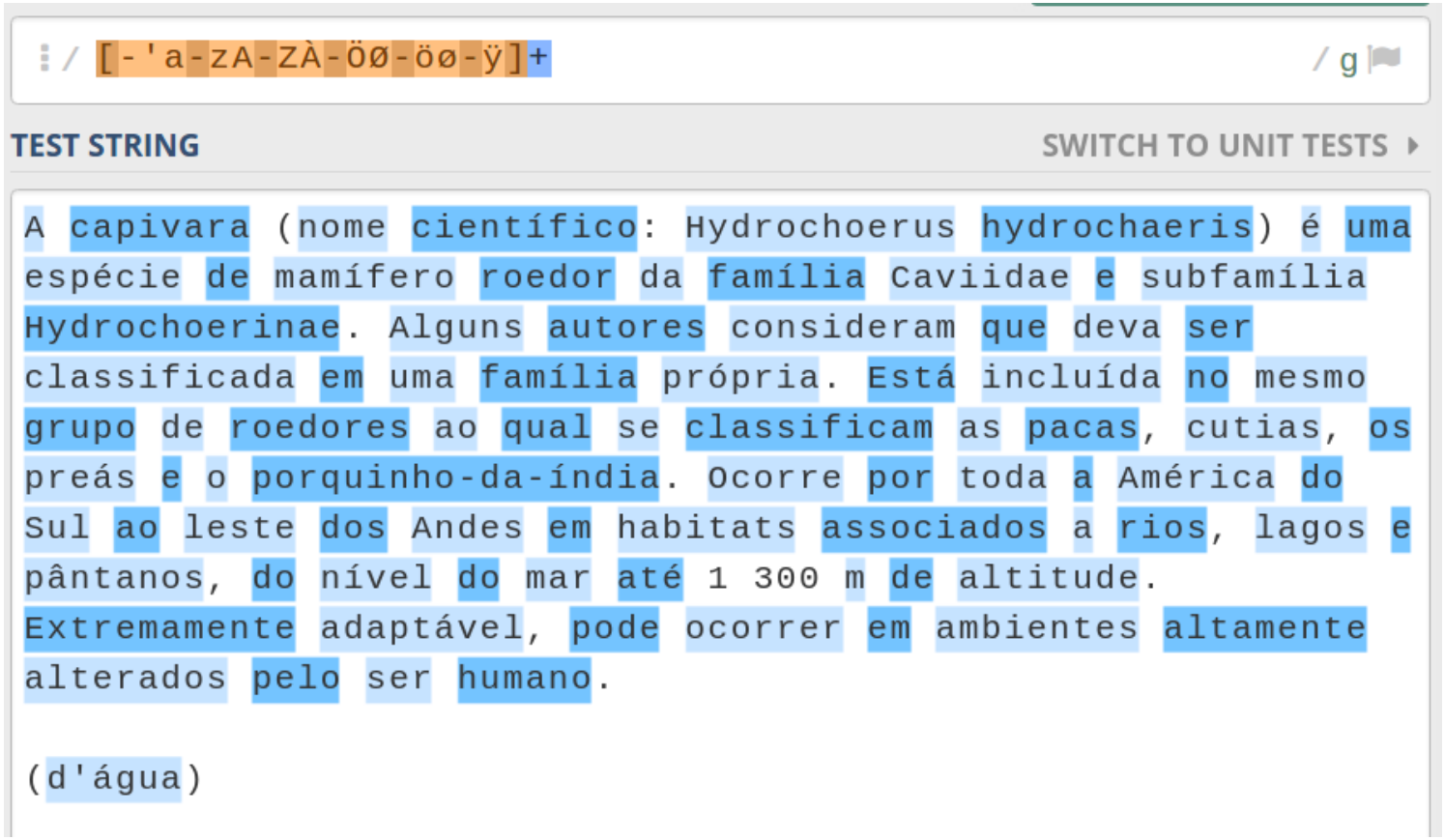


Tabela ASCII estendida -
https://www.ascii-code.com

[-'a-zA-ZÀ-ÖØ-öø-ÿ]

190	276	BE	10111110	¾
191	277	BF	10111111	¿
192	300	C0	11000000	À
193	301	C1	11000001	Á
194	302	C2	11000010	Â
195	303	C3	11000011	Ã
196	304	C4	11000100	Ä
197	305	C5	11000101	Å
198	306	C6	11000110	Æ
199	307	C7	11000111	Ç
200	310	C8	11001000	È
201	311	C9	11001001	É
202	312	CA	11001010	Ê
203	313	CB	11001011	Ë
204	314	CC	11001100	Ì
205	315	CD	11001101	Í
206	316	CE	11001110	Î
207	317	CF	11001111	Ï
208	320	D0	11010000	Ð
209	321	D1	11010001	Ñ
210	322	D2	11010010	Ò
211	323	D3	11010011	Ó
212	324	D4	11010100	Ô
213	325	D5	11010101	Õ
214	326	D6	11010110	Ö
215	327	D7	11010111	×
216	330	D8	11011000	Ø
217	331	D9	11011001	Ù
218	332	DA	11011010	Ú
219	333	DB	11011011	Û
220	334	DC	11011100	Ü

220	334	DC	11011100	Ü
221	335	DD	11011101	Ý
222	336	DE	11011110	Þ
223	337	DF	11011111	ß
224	340	E0	11100000	à
225	341	E1	11100001	á
226	342	E2	11100010	â
227	343	E3	11100011	ã
228	344	E4	11100100	ä
229	345	E5	11100101	å
230	346	E6	11100110	æ
231	347	E7	11100111	ç
232	350	E8	11101000	è
233	351	E9	11101001	é
234	352	EA	11101010	ê
235	353	EB	11101011	ë
236	354	EC	11101100	ì
237	355	ED	11101101	í
238	356	EE	11101110	î
239	357	EF	11101111	ï
240	360	F0	11110000	ð
241	361	F1	11110001	ñ
242	362	F2	11110010	ò
243	363	F3	11110011	ó
244	364	F4	11110100	ô
245	365	F5	11110101	õ
246	366	F6	11110110	ö
247	367	F7	11110111	÷
248	370	F8	11111000	ø
249	371	F9	11111001	ù
250	372	FA	11111010	ú

251	373	FB	11111011	û
252	374	FC	11111100	ü
253	375	FD	11111101	ý
254	376	FE	11111110	þ
255	377	FF	11111111	ÿ

Palavras (tokenização)

Palavras

Quantas palavras temos na seguinte frase?

```
Extremamente adaptável, pode ocorrer em
ambientes altamente alterados pelo ser
humano.
```

Palavras

Quantas palavras temos na seguinte frase?

```
Extremamente adaptável, pode ocorrer em
ambientes altamente alterados pelo ser
humano.
```

- 11 palavras.
- 13 palavras se considerarmos também os sinais de pontuação.

Palavras

Na frase:

```
Muito longo para os que lamentam,
muito curto para os que festejam
```

- Quantas palavras existem?
- Quantas palavras diferentes existem?

Palavras

Na frase:

```
Muito longo para os que lamentam,
muito curto para os que festejam
```

- Quantas palavras existem? **12**
- Quantas palavras diferentes existem? **8**

Palavras

Na frase:

```
Muito longo para os que lamentam,
muito curto para os que festejam
```

- Quantas palavras existem? **12 (tokens)**
- Quantas palavras diferentes existem? **8 (tipo/vocabulário)**

Corpus	Tokens = N	Types = $\|V\|$
Shakespeare	884 thousand	31 thousand
Brown corpus	1 million	38 thousand
Switchboard telephone conversations	2.4 million	20 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13 million

Palavras

Na frase:

```
Muito longo para os que lamentam,
muito curto para os que festejam
```

- Quantas palavras existem? **12** **(tokens)**
- Quantas palavras diferentes existem? **8** **(tipo/vocabulário)**

Coletânea sobre um determinado assunto.
Plural: corpora

Corpus	Tokens = N	Types = $\|V\|$
Shakespeare	884 thousand	31 thousand
Brown corpus	1 million	38 thousand
Switchboard telephone conversations	2.4 million	20 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13 million

Corpus textual - Corpora textuais

No contexto de PLN, um **corpus** é um conjunto de documentos (ou de frases) geralmente anotados e utilizados para:

- Aprendizado (análise)
- Validação (verificação)

https://en.wikipedia.org/wiki/List_of_text_corpora
https://www.corpusdoportugues.org/x.asp

Created by Mark Davies, BYU. Funded by the US National Endowment for the Humanities (2004, 2015). Part of the BYU collection of corpora.

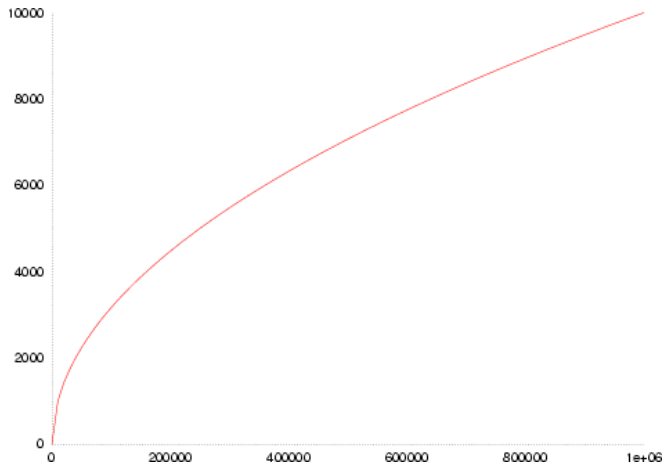
	Corpus	Size	Created	More info
1	**Genre / Historical**	45 million words	2006	Info
2	**Web / Dialects**	1 *billion* words	2016	Info
NEW	**WordAndPhrase**	Top 40,000 words	2017	Info

Português como lingua oficial

País	População (est. 2014)[1]
Brasil	205.749.746
Moçambique	24.692.144
Angola	24.300.000[2]
Portugal	10.813.834
Guiné-Bissau	1.693.398
Timor-Leste	1.201.542
Guiné Equatorial	722.254
Macau	587.914
Cabo Verde	538.535
São Tomé e Príncipe	190.428
Total	267.396.837

Análise empírica (evidências):
Lei de Herdan (1960), Lei de Heaps (1978)

V = Vocabulário



N = Tokens (e.g., palavras na coleção)

Crescimento de um vocabulário

$$|V| = kN^{\beta}$$

$$0 < \beta < 1$$

Corpus	Tokens = N	Types = $\|V\|$
Shakespeare	884 thousand	31 thousand
Brown corpus	1 million	38 thousand
Switchboard telephone conversations	2.4 million	20 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13 million

No caso do texto: Capivara
|V|=133, N=192

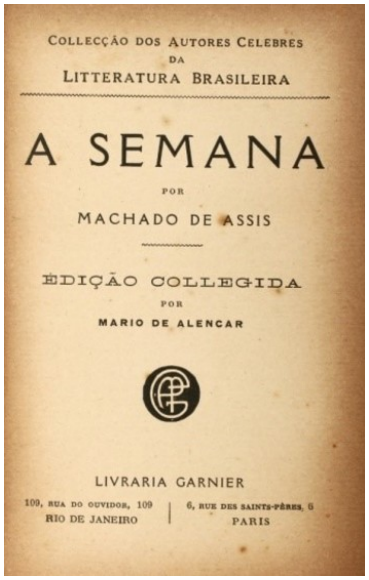
(ver programa na página da disciplina)

Palavras: Prática em python

Prática em python

Machado de Assis (1839-1908)
Crônicas reunidas na coleção “A semana”.

Escreveu em praticamente todos os gêneros literários, sendo poeta, romancista, cronista, dramaturgo, contista, folhetinista, jornalista e crítico literário.




```
$ python palavras1.py A-Semana-Machado-de-Assis.txt
…
que
é
tempo

Quantidade de palavras: 267618
```

Prática em python

```python
import sys
import re

#regex = r"[-'a-zA-ZÀ-ÖØ-öø-ÿ]+"   # raw string
regex = r"[-'a-zA-ZÀ-ÖØ-öø-ÿ0-9]+"   # raw string


if __name__ == '__main__':
    fileName = sys.argv[1]

    document = open(fileName,'r')

    content  = document.read()
    # document.read()       # devolve o conteudo do arquivo 'fileName'
    # document.readline()   # devolve a seguinte linha do arquivo 'fileName'
    # document.readlines()  # devolve uma lista de linhas do arquivo 'fileName'


    words = re.findall(regex, content)
    for w in words:
        print (w)
    print (f"Quantidade de palavras: {len(words)}")
```

Prática em python

```
$ python palavras1.py A-Semana-Machado-de-Assis.txt
O
pobre-diabo
recorreu
a
esse
meio
para
almoçar
um
dia
Se
tal
foi
façam
de
conta
que
não
escrevi
nada
e
vão
almoçar
também
que
é
tempo
Quantidade de palavras: 267618
```

Prática em python

```
1  import sys
2  import re
3
4  regex = r"[-'a-zA-ZÀ-ÖØ-öø-ÿ0-9]+"   # raw string
5
6  if __name__ == '__main__':
7      fileName = sys.argv[1]
8
9      # leitura do documento
10     document = open(fileName,'r')
11     content  = document.read()
12
13     # identificacao de palavras
14     words       = re.findall(regex, content)
15     frequencies = dict([])
16
17     # quantidade de vezes no documento
18     for w in words:
19         w = w.lower()
20         if w not in frequencies:
21             frequencies[w] = 0
22         frequencies[w] += 1
23     print (f"Tokens: {len(words)}, Vocabulario: {len(frequencies)}")
24
25     # imprimir as 20 palavras mais frequentes
26     fs = sorted(frequencies, key=frequencies.get, reverse=True)
27     for i in range(0,20):
28         print (f"--> {frequencies[fs[i]]} {fs[i]}")
```

Prática em python

```
$ python palavras2.py A-Semana-Machado-de-Assis.txt
Tokens: 267618, Vocabulario: 25374
--> 11053 que
--> 9875 a
--> 9767 de
--> 8236 o
--> 7525 e
--> 5519 não
--> 4018 é
--> 3476 do
--> 3465 os
--> 3352 um
--> 3203 da
--> 2862 se
--> 2421 as
--> 2237 em
--> 2054 uma
--> 2024 com
--> 2019 mas
--> 1811 para
--> 1756 por
--> 1456 ao
```

Palavras:
Algumas considerações não triviais

Considerações não triviais

Quantas palavras?

- Total de R$10,45
- Para valores superiores a 45.455,67
- www.ufabc.edu.br
- UFABC
- Livre-docente
- Homem-Máquina
- D'água
- U.F.A.B.C.
- C.M.C.C.
- Ph.D.
- Sant'Anna
- L'ensemble
- Lebensversicherungsgesellschaftsangestellter
- 莎拉波娃现在居住在美国东南部的佛罗里达

Palavras (primeiro programa)

```
Total
de
R
10
45
Para
valores
superiores
a
45
455
67
www
ufabc
edu
br
UFABC
Livre-docente
Homem-Máquina
D'água
U
F
A
B
C
C
M
C
C
Ph
D
Sant'Anna
L'ensemble
Lebensversicherungsgesellschaftsangestellter
```

Frequências (segundo programa)

```
Tokens: 34, Vocabulario: 28
 4 c
 2 a
 2 ufabc
 2 45
 1 para
 1 455
 1 r
 1 homem-máquina
 1 67
 1 f
 1 d
 1 d'água
 1 u
 1 lebensversicherungsgesellschaftsangestellter
 1 livre-docente
 1 br
 1 superiores
 1 edu
 1 sant'anna
 1 total
```

Algoritmo: Maximum Mathing

- É requerido um dicionário.
- Busca gulosa.

Input: 他特别喜欢北京烤鸭
Output: 他 特别 喜欢 北京烤鸭

Input: wecanonlyseeashortdistanceahead
Output: we canon l y see ash ort distance ahead

Os algoritmos de segmentação mais sofisticados usam modelos estatísticos (aprendizado supervisionado).

Palavras com pouco sentido semântico: *stopword / stoplist*

- Em **muitos** contextos:

  Uma *stopword* pode ser considerada uma **palavra irrelevante** para análise (artigos, preposições).

a ao aos aquela aquelas aquele aqueles aqui aquilo as até aí com como da das de dela delas dele deles depois do dos dá e ela elas ele eles em entre era eram essa essas esse esses esta estamos estas estava estavam este esteja estejam estejamos estes esteve estive estivemos estiver estivera estiveram estiverem estivermos estivesse estivessem estivéramos estivéssemos estou está estávamos estão eu foi fomos for fora foram forem formos fosse fossem fui fôramos fôssemos haja hajam hajamos havemos hei houve houvemos houver houvera houveram houverei houverem houveremos houveria houveriam houvermos houverá houverão houveríamos houvesse houvessem houvéramos houvéssemos há hão isso isto já lhe lhes lá mais mas me mesmo meu meus minha minhas muito na nas nem no nos nossa nossas nosso nossos num numa não nós o os ou para pela pelas pelo pelos por pra qual quando que quem se seja sejam sejamos sem serei seremos seria seriam será serão seríamos seu seus somos sou sua suas são só também te tem temos tenha tenham tenhamos tenho terei teremos teria teriam terá terão teríamos teu teus teve tinha tinham tive tivemos tiver tivera tiveram tiverem tivermos tivesse tivessem tivéramos tivéssemos tu tua tuas tá tém tínhamos um uma vai você vocês vos vou à às é éramos

```python
import sys
import re

regex = r"[-'a-zA-ZÀ-ÖØ-öø-ÿ0-9]+"   # raw string

if __name__ == '__main__':
    fileName = sys.argv[1]

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

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

    # identificacao de palavras
    words       = re.findall(regex, content)
    frequencies = dict([])

    # quantidade de vezes no documento
    for w in words:
        w = w.lower()
        if w not in stopwords:
            if w not in frequencies:
                frequencies[w] = 0
            frequencies[w] += 1
    print (f"Tokens: {len(words)}, Vocabulario: {len(frequencies)}")

    # imprimir as 20 palavras mais frequentes
    fs = sorted(frequencies, key=frequencies.get, reverse=True)
    for i in range(0,20):
        print (f"--> {frequencies[fs[i]]} {fs[i]}")
```

```
$ python palavras2.py A-Semana-Machado-de-Assis.txt
Tokens: 267618, Vocabulario: 25374
--> 11053 que
--> 9875 a
--> 9767 de
--> 8236 o
--> 7525 e
--> 5519 não
--> 4018 é
--> 3476 do
--> 3465 os
--> 3352 um
--> 3203 da
--> 2862 se
--> 2421 as
--> 2237 em
--> 2054 uma
--> 2024 com
--> 2019 mas
--> 1811 para
--> 1756 por
--> 1456 ao
```

```
$ python palavras3.py A-Semana-Machado-de-Assis.txt
Tokens: 267618, Vocabulario: 25182
--> 703 ser
--> 629 ainda
--> 564 tudo
--> 526 pode
--> 475 outro
--> 471 homem
--> 455 todos
--> 454 assim
--> 437 outra
--> 420 outros
--> 409 grande
--> 385 dois
--> 384 coisa
--> 375 tempo
--> 367 dia
--> 361 bem
--> 360 menos
--> 353 vez
--> 339 porque
--> 334 onde
```

Palavras com pouco sentido semântico?

- 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

- Muitas aplicações (e.g., motores de busca) não usam stopwords.

Palavras com pouco sentido semântico?

JOURNAL OF THE ASSOCIATION FOR INFORMATION SCIENCE AND TECHNOLOGY

Explore this journal >

Brief Communication

When stopword lists make the difference

Ljiljana Dolamic, Jacques Savoy

First published: 29 July 2009 Full publication history

DOI: 10.1002/asi.21186 View/save citation

Cited by (CrossRef): 4 articles Check for updates | Citation tools

View issue TOC
Volume 61, Issue 1
January 2010
Pages 200–203

Abstract

In this brief communication, we evaluate the use of two stopword lists for the English language (one comprising 571 words and another with 9) and compare them with a search approach accounting for all word forms. We show that through implementing the original Okapi form or certain ones derived from the *Divergence from Randomness* (DFR) paradigm, significantly lower performance levels may result when using short or no stopword lists. For other DFR models and a revised Okapi implementation, performance differences between approaches using short or long stopword lists or no list at all are usually not statistically significant. Similar conclusions can be drawn when using other natural languages such as French, Hindi, or Persian.

O tamanho da lista de stopwords pode reduzir a efetividade em uma busca, por exemplo.

Stop word / stop list ~1960

Esses termos foram já discutidos desde aprox. 1960:

- Luhn, H.P. (1959). **Keyword in Context Index for Technical Literature** (KWIC Index), Yorktown Heights, NY: IBM, Report RC 127.
- Parkins, P.V. (1963). **Approaches to vocabulary management in permuted title indexing of Biological Abstracts.** In Automation and Scientific Communication Part 1, Proceedings of The 26th Annual Meeting of the American Documentation Institute (pp. 27–28), Washington, DC: ADI.
- Fischer, M. (1966). **The KWIC index concept: A retrospective view.** American Documentation, 17, 57–70.

Normalizando as palavras

Normalizando as palavras

```
$ python palavras3.py 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
```

Normalizando as 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.

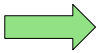
Stemming

for
exampl**e**
compress**ed**

and
compress**ion**
ar**e**
both

accept**ed**
as
equival**ent**
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 técnicas 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).

Recuperação de Informação – IR

- Um vocabulário menor pode levar a um melhor desempenho na busca.

- As operações de stemming e lemmatization são operações comuns e iniciais utilizadas em sistemas de recuperação de informação:
  - Busca: Stemming?
  - Tradução: Lemmatisation?

Sobre o resumo da aula de hoje

Resumo
A disciplina de Processamento de Linguagem Natura O curso tem diversas referências bibliográficas, mas deverão entregar resumos de 250 a 500 palavras à re intervalos dos conceitos serão os seguintes: A - se a linguagem Python, estrutura de dados e PLN. Pois al qual abrange linguagem para programação, isto é, nã linguística computacional e está relacionada ao apren sistema de resposta que tem resultados muito preciso Wolpfram Apha, Google Assistant, Amazon Alexia, funcionamento da linguagem humana, já o segundo t mais complexos de resolver. Pois problemas com est um IA- Complete seria desenvolver uma aplicação q
A aula começou com a apresentação do professor Jes estão crescendo de forma exponencial, a cada dois an processamento de discurso e Aplicações. Foi passado 30% da disciplina. Também teremos uma prova teóri seguinte forma: A para notas >=9, B: notas <9 e >= 7 de Inteligência Artificial e Linguistica, definido com em inglês) está muito mais associada a linguagem na de busca semântica no WolframAlpha. PLN está rela entender um texto significa reconhecer seu contexto, microsoft. IA-Complete é um problema extremamen
Introdução Sala do Professor: 517-A, torre 2 Bibliog relatorio 2 - 3 páginas - 25/07 \- Mini relatório 3 - 5 p 7\. nosso codigo e fonte de dados devem ser disponib Linguagem Natural: Linguagens faladas/escritas por sintática 6\. pesquisas Exemplo Comando ```wc``` no retornar uma data específica. Isso funciona também resposta. Perguntas mais complexas Como por exem atuais Os sistemas clássicos usaram termos simples I associado com: 1\. Linguistica computacionaç 2\. Te como tratar computadores tão inteligentes como pess
03/06/2019 A primeira aula de PLN (Processamento intuito de ser um material de referência de estudo, ta terá um valor equivalente a 40% da nota final. 3 – O dividida em 4 etapas: 1 – Mini relatório, com uma pá Apresentação oral, que será uma apresentação ao pro entendimento de linguagem natural (falada ou escrita chaves, cuja dificuldade é menor e é a tecnologia ma algoritmo, AI-Complete é usado, segundo MIT e Sta construção de um sistema que pode enxergar da mes

Processamento de Linguagem Natural (PLN) tem relaçã relação com Compiladores, prova de teoremas, modelos linguagem humana é difícil de entender, uma sentença p humano, assim este problema não é resolvido somente c um projeto que consiste de relatórios e apresentação, val
Na aula do dia 03/06, segunda-feira, o professor iniciou grupo e apresentado no final do quadrimestre, e que dev mencionou que o processamento de linguagem natural é desenvolvimento), exemplos de processamento de lingua área de inteligência artificial que são considerados os ma
Resumo 1 Forma de avaliação A avaliação na disciplina computadores capazes de entender a linguagem natural. sintática e semântica. Assim, PLN se relaciona com com que exigem Inteligência Artificial para resolução, como analogia aos problemas NP-complete (complexidade com
Paulo Alexander Simões RA: 11084915 Resumo - PLN Ou seja, problemas que requerem este nível de solução n criação de soluções para estes problemas compexos, é ne não foram formalisados. Um dos maiores desafios para p serem corrigidas. Deve-se desenvolver em tal solução di
Processamento de Linguagem Natural (PLN) é uma subá humana, tanto escrita como falada, ou para construção d modernos, sendo necessária a intervenção humana para a máquina, interação humano-computador e inteligência a que podemos citar são: uso de reconhecimento de voz pa
A primeira aula teve caráter introdutório tanto da estrutu que mais vale) e apresentação (que completa a nota dos essa inicialmente não estruturada para análise computaci entendimento da linguagem humana; outro focado no de como humanos?". Se uma máquina tivesse a capacidade sfnuie e fieaj eEIUFE iunes eNIE eONWE p noqeo fewi In Stuart C. Shapiro (Ed.), Encyclopedia of Artificial Int
Resumo aula 01 Processamento de Linguagem Natural é amplo, com um vocabulário léxico complexo com const exigem operações ainda muito complexas. O entendimen
Nesta aula fomos introduzidos a disciplina e pegamos in deverá ser entregue um resumo em até 48h após cada au algoritmo descrito juntamente de um relatório.
A primeira aula foi uma apresentação e introdução sobre cresce exponencialmente, com algumas previsões para o

Resumo 3 - Tidia

- **Deadline (daqui a ~48h)**
  - Aprox. 250-500 palavras. Apenas texto.
  - Dia: 12/jun, até às 23h50.