

# **Expressões regulares**

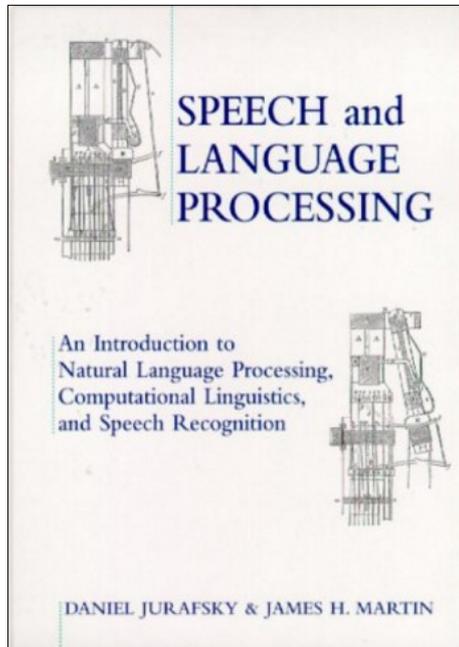
Prof. Jesús P. Mena-Chalco  
jesus.mena@ufabc.edu.br

2Q-2019

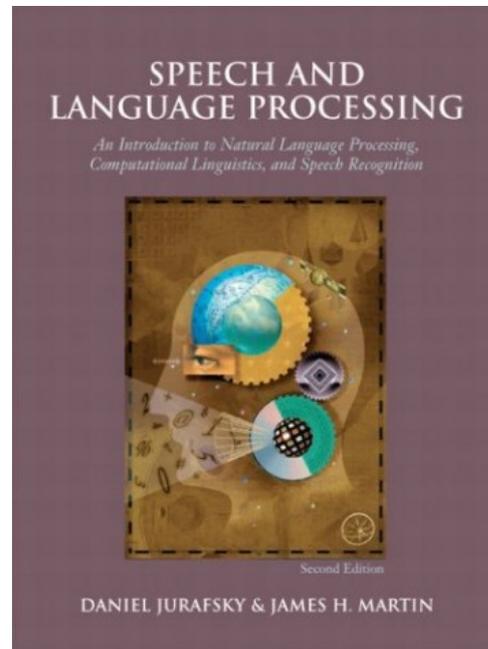
# Bibliografia

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

# Bibliografía – 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: <u><a href="#">Regular Expressions, Text Normalization, and Edit Distance</a></u>	Text [ <a href="#">pptx</a> ] [ <a href="#">pdf</a> ] Edit Distance [ <a href="#">pptx</a> ] [ <a href="#">pdf</a> ]	[Ch. 2 and parts of Ch. 3 in 2nd ed.]
3: <u><a href="#">Language Modeling with N-Grams</a></u>	LM [ <a href="#">pptx</a> ] [ <a href="#">pdf</a> ]	[Ch. 4 in 2nd ed.]
4: <u><a href="#">Naive Bayes Classification and Sentiment</a></u>	NB [ <a href="#">pptx</a> ] [ <a href="#">pdf</a> ] Sentiment [ <a href="#">pptx</a> ] [ <a href="#">pdf</a> ]	[new in this edition]
5: <u><a href="#">Logistic Regression</a></u>		
6: <u><a href="#">Vector Semantics</a></u>	Vector1 [ <a href="#">pptx</a> ] [ <a href="#">pdf</a> ] Vector2 [ <a href="#">pptx</a> ] [ <a href="#">pdf</a> ]	[new in this edition]
7: <u><a href="#">Neural Nets and Neural Language Models</a></u>		[Ch. 5 in 2nd ed.]
8: <u><a href="#">Part-of-Speech Tagging</a></u>		[new in this edition]
9: <u><a href="#">Sequence Processing with Recurrent Networks</a></u>		[new in this edition]
X: Encoder-Decoder Models and Attention		[new in this edition]
10: <u><a href="#">Formal Grammars of English</a></u>		[Ch. 12 in 2nd ed.]
11: <u><a href="#">Syntactic Parsing</a></u>		[Ch. 13 in 2nd ed.]
12: <u><a href="#">Statistical Parsing</a></u>		[Ch. 14 in 2nd ed.]
13: <u><a href="#">Dependency Parsing</a></u>		[new in this edition]



# Procurando padrões textuais

# Procurando padrões textuais

- As expressões regulares (*Regular expression* – **RE ou ER**) podem ser utilizadas para **extrair trechos** a partir de texto.
- As ERs são comumente utilizadas para:
  - Normalização de texto (e.g., padronizar o texto convenientemente)
  - Divisão em *tokens* (e.g., divisão em palavras usando os espaços?)
  - Radicalização (e.g., lemmatization, stemming)
  - Segmentação de frases (e.g., divisão em frases usando a pontuação)
- **O que são ERs?** Cadeias de texto especiais, *em uma linguagem formal*, para busca/extração de trechos de texto.
- **Sinônimos:** Regex, Regexp.

# Procurando padrões textuais



É o maior roedor do mundo, pesando até 91 kg e medindo até 1,2 m de comprimento e 60 cm de altura.

# Procurando padrões textuais

Como podemos procurar por **qualquer** um dos seguintes nomes?

- capivara
- **Capivara**



É o maior roedor do mundo, pesando até 91 kg e medindo até 1,2 m de comprimento e 60 cm de altura.

# Procurando padrões textuais

Como podemos procurar por **qualquer** um dos seguintes nomes?

- capivara
- **Capivara**



É o maior roedor do mundo, pesando até 91 kg e medindo até 1,2 m de comprimento e 60 cm de altura.

[cC]apivara

# 1) Disjunções

- A cadeia de caracteres entre colchetes/parênteses retos (i.e, '[' e ']') especifica uma **disjunção** de caracteres para a **busca**.

Padrão	Casamento
[cC]apivara	Capivara capivara
[1234567890]	Qualquer dígito (mas apenas 1 dígito)

# 1) Disjunções

- Faixas de caracteres

Padrão	Casamento	Exemplo
[0-9]	Apenas 1 dígito	AED <u>1</u> : Algoritmos e Est...
[a-z]	Um caractere em minúscula	AED1 : A <u>l</u> goritmos e Est...
[A-Z]	Um caractere em maiúscula	<u>A</u> ED1 : Algoritmos e Est...
[5-8]	Apenas um dígito: 5, 6, 7 ou 8.	AED1 : Algoritmos e Est...

## 2) Negação em Disjunções: ^

- O símbolo do acento circunflexo é utilizado para negar uma disjunção.
- Obs: A negação apenas válida como primeiro caractere.

Padrão	Casamento	Exemplo
[^A-Z]	Não um caractere em maiúscula	Ciência da computação
[^Ss]	Nem 'S' nem 's'	<u>A</u> ED1 : Algoritmos e Est...

## 2) Negação em Disjunções: ^

Padrão	Casamento	Exemplo
$[^a^]$		^Ciência da computação...
$a^b$	$a^b$	O valor final é $a^b$ com a...

## 2) Negação em Disjunções: ^

Padrão	Casamento	Exemplo
[^a^]	Nem 'a' nem '^'	^ <u>C</u> iência da computação...
a^b	Padrão a^b	O valor final é <u>a^b</u> com a...

### 3) Mais sobre Disjunções: | (barra vertical)

- Capivara também é conhecido como Carpincho.

Padrão	Casamento
<code>[cC]apivara [cC]arpincho</code>	
<code>eu voce</code>	
<code>[a b c]</code>	

### 3) Mais sobre Disjunções: | (barra vertical)

- Capivara também é conhecido como Carpincho.

Padrão	Casamento
<code>[cC]apivara [cC]arpincho</code>	capivara Capivara carpincho Carpincho
<code>eu voce</code>	eu voce
<code>[a b c]</code>	<code>=[abc]</code>

## 4) Outras opções: ? \* + .

Padrão	Casamento	Exemplo
pa?ra	Caractere predecessor opcional	para pra
aa*h!	Caractere predecessor: 0 ou mais vezes	ah! aah! aaah! aaaaah!
o+h!	Caractere predecessor: 1 ou mais vezes	Oh! Ooh! Oooh!
ca.a		casa caza caça

## 4) Outras opções: ? \* + .

Padrão	Casamento
$[ab]^*$	
$[0-9][0-9]^*$	

## 4) Outras opções: ? \* + .

Padrão	Casamento
$[ab]^*$	aaaaaaaa ababababa aaaabbbbb baaaaaaaaa bbbbbbbbbbbbbb
$[0-9][0-9]^*$	Um número de pelo menos um dígito.  Por que não considerarmos apenas $[0-9]^*$ ?

## 5) Caracteres âncoras: ^ \$

■ Início de linha: ^

Final de linha: \$

Padrão	Casamento
<code>^[A-Z]</code>	<u>S</u> anto André
<code>a\$</code>	Casa <u>a</u>
<code>\.\$</code>	Casa <u>.</u>
<code>^A UFABC\.\$</code>	Uma linha contendo exatamente A UFABC <u>.</u>

O caractere '\ ' serve para indicar que o caractere sucessor não é especial.

# Exemplos

- Procurar pelas instâncias da palavra: **de**

[dD]e

## Regular Expression Syntax

Expression	Description	Examples and Expansions
<b>Single character expressions</b>		
.	any single character	sp <code>i.e</code> matches "spice", "spike", etc.
<code>\char</code>	for a nonalphanumeric <i>char</i> , matches <i>char</i> literally	<code>\*</code> matches "*"
<code>\n</code>	newline character	
<code>\r</code>	carriage return character	
<code>\t</code>	tab character	
<code>[...]</code>	any single character listed in the brackets	<code>[abc]</code> matches "a", "b", or "c"
<code>[...-...]</code>	any single character in the range	<code>[0-9]</code> matches "0" or "1" ... or "9"
<code>[^...]</code>	any single character not listed	<code>[^sS]</code> matches one character that is neither "s" nor "S"
<code>[^...-...]</code>	any single character not in the range	<code>[^A-Z]</code> matches one character that is not an uppercase letter
<b>Anchors/Expressions which match positions</b>		
<code>^</code>	beginning of line	
<code>\$</code>	end of line	
<code>\b</code>	word boundary	<code>nt\b</code> matches "nt" in "paint" but not in "pants"
<code>\B</code>	word non-boundary	<code>a11\B</code> matches "all" in "ally" but not in "wall"
<b>Counters/Expressions which quantify previous expressions</b>		
<code>*</code>	zero or more of previous r.e.	<code>a*</code> matches "", "a", "aa", "aaa", ...
<code>+</code>	one or more of previous r.e.	<code>a+</code> matches "a", "aa", "aaa", ...
<code>?</code>	exactly one or zero of previous r.e.	<code>colou?r</code> matches "color" or "colour"
<code>{n}</code>	<i>n</i> of previous r.e.	<code>a{4}</code> matches "aaaa"
<code>{n,m}</code>	from <i>n</i> to <i>m</i> of previous r.e.	
<code>{n,}</code>	at least <i>n</i> of previous r.e.	
<code>.*</code>	any string of characters	
<code>(...)</code>	grouping for precedence and memory for backreference	
<code>... ...</code>	matches either of neighbor r.e.s	<code>(dog) (cat)</code> matches "dog" or "cat"
<b>Shortcuts</b>		
<code>\d</code>	any digit	<code>[0-9]</code>
<code>\D</code>	any non-digit	<code>[^0-9]</code>
<code>\w</code>	any alphanumeric/underscore	<code>[a-zA-Z0-9_]</code>
<code>\W</code>	any non-alphanumeric	<code>[^a-zA-Z0-9_]</code>
<code>\s</code>	whitespace (space, tab)	<code>[\r\t\n\f]</code>
<code>\S</code>	non-whitespace	<code>[^\r\t\n\f]</code>

`[^a-zA-Z][dD]e[^A-Za-z]`



regular expressions 101

@regex101 donate contact bug reports & feedback

REGULAR EXPRESSION 11 matches, 1474 steps (~5ms)

`ir" [^a-zA-Z][dD]e[^A-Za-z] " g`

TEST STRING SWITCH TO UNIT TESTS ▶

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.

EXPLANATION

- ▼ " `[^a-zA-Z][dD]e[^A-Za-z]` " g
  - ▼ Match a single character not present in the list below
    - `[^a-zA-Z]` a single character in the range between **a** (index 97) and **z** (index 122) (case sensitive)
    - `A-Z` a single character in the range between **A** (index 65) and **Z** (index 90) (case sensitive)
  - ▼ Match a single character present in the list below
    - `[dD]` matches a single character in the list **dD** (case sensitive)

MATCH INFORMATION

Match 1  
Full match 69-73 ` de `

Match 2  
Full match 242-246 ` de `

Match 3  
Full match 457-461 ` de `

Match 4

QUICK REFERENCE



**REGULAR EXPRESSION** 11 matches, 1635 steps (~12ms)

`:r" \b[dD]e\b` " g

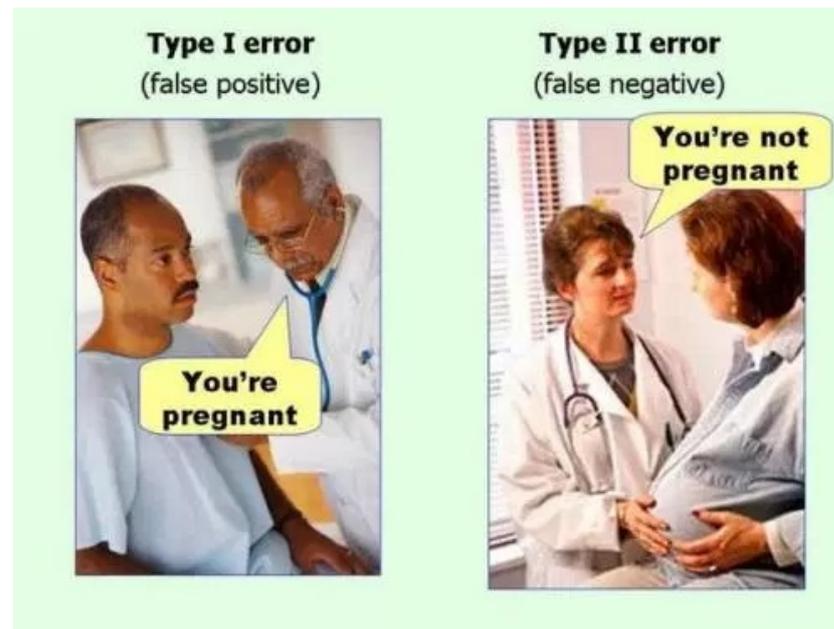
**TEST STRING** SWITCH TO UNIT TESTS ▶

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.

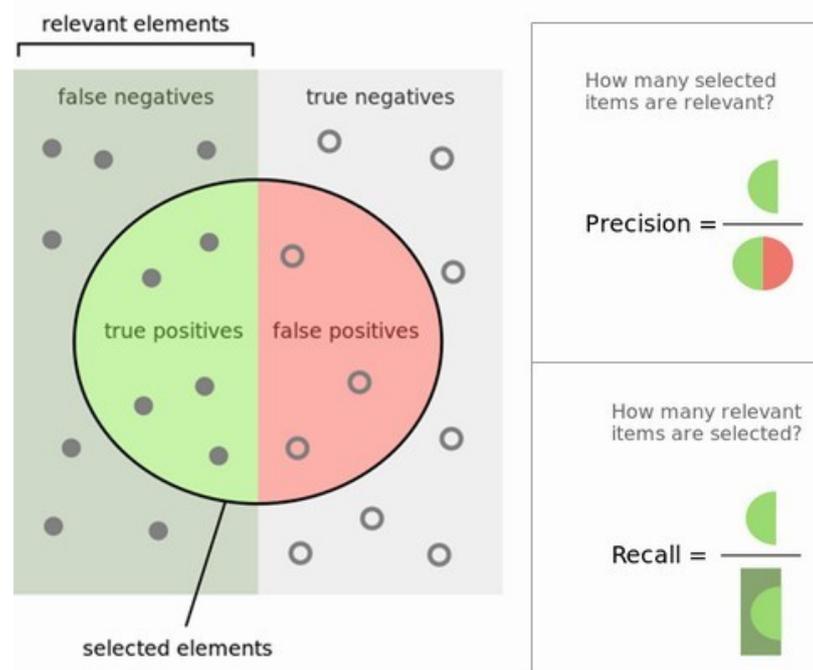
# Dois tipos de erros

- **Falso positivo:**  
Identificar cadeias que não deveriam ser identificadas (*humanidade, idade, cadeira*)
- **Falso negativo:**  
Não identificar cadeias que deveriam ser identificadas (*De*)



# Dois tipos de erros

- Em PLN, e qualquer outro problema de reconhecimento de padrões, devemos de **lidar** com esses 2 tipos de erros.
- Minimizar os Falsos Positivos: **Aumentando a precisão.**
- Minimizar os Falsos Negativos: **Aumentando a cobertura.**



# Expressões regulares

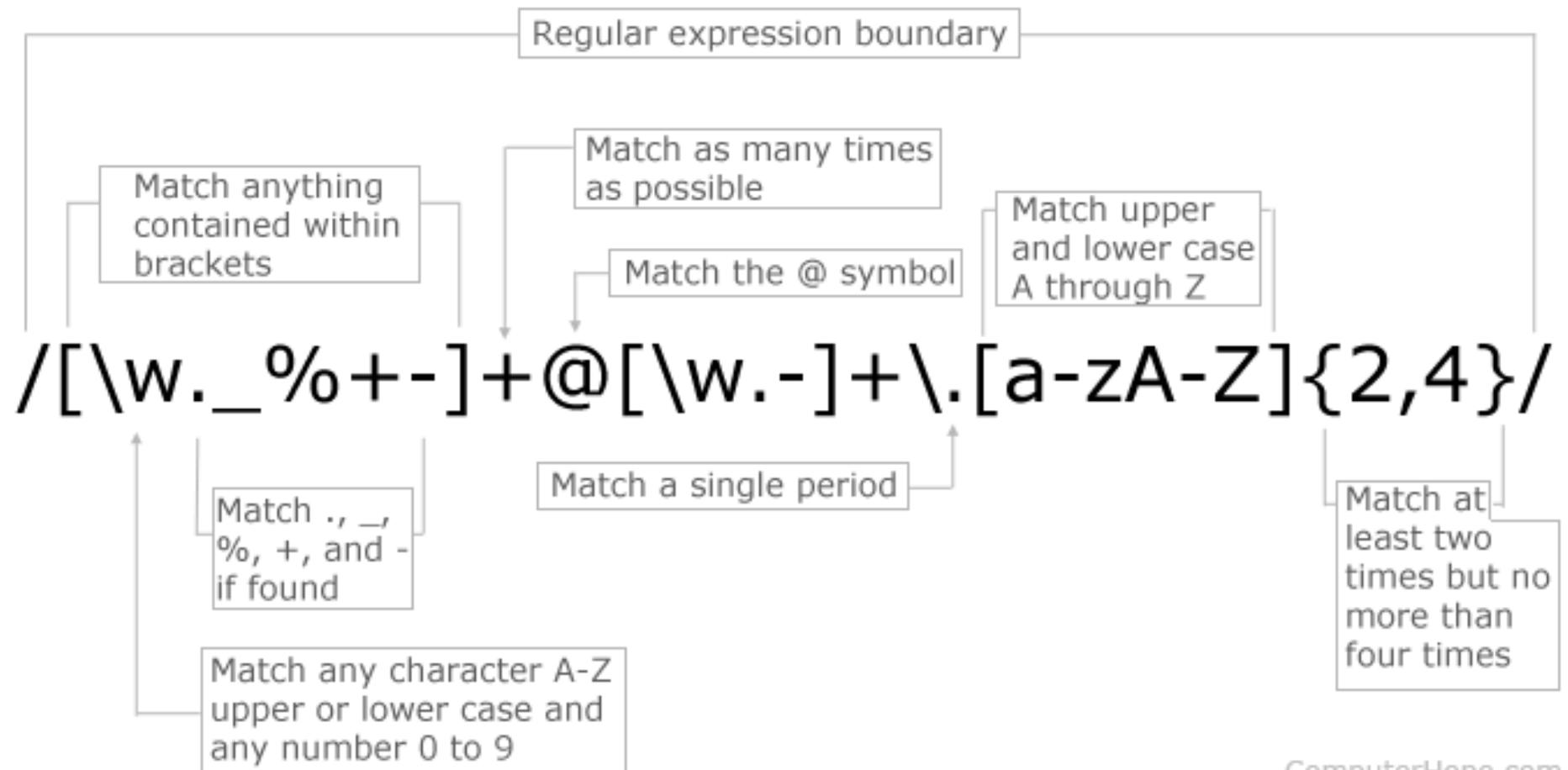


- São utilizadas, frequentemente, como primeiro passo para qualquer processamento de texto.
- O resultado das ERs é utilizado em classificadores de técnicas de aprendizado de máquina.

## **Desafio:**

Crie uma ER para identificar emails

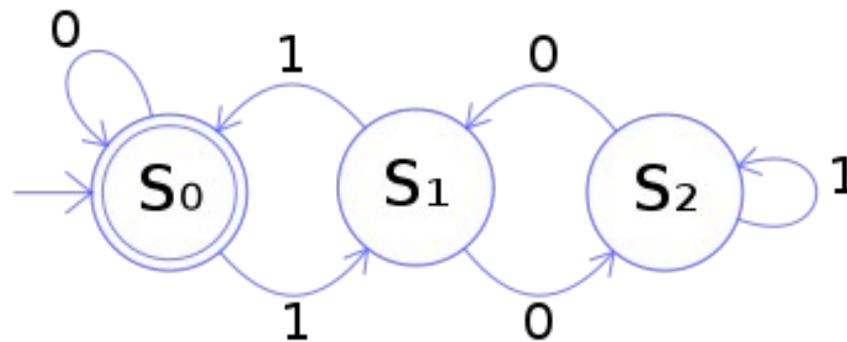
# Regular Expression E-mail Matching Example



# Máquina de estados finita

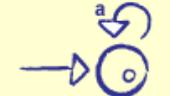
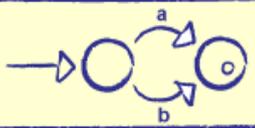


- Uma ER pode ser representado/descrito por uma **MEF** ou **Autômato finito determinístico**.
- Uma MEF é uma máquina abstrata que está em um número finito de estados.



# Máquina de estados finita

- Para verificar se uma ER está bem construída pode **descrever/representar** a instrução usando um autômato finito determinista.

regular expression	finite state machine
$a$	
$a^*$	
$a^+$	
$a/b$	
$(a/b)^*c(d/e)$	

*~f 2006*



# **Teste de avaliação em aula**

# Operadores e Precedência



*Maior*

Parenthesis	()
Counters	* + ? {}
Sequences and anchors	the ^my end\$
Disjunction	

*Menor*

RE	Expansion	Match	First Matches
\d	[0-9]	any digit	Party_of_5
\D	[^0-9]	any non-digit	Blue_moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in_Concord

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	n occurrences of the previous char or expression
{n,m}	from n to m occurrences of the previous char or expression
{n, }	at least n occurrences of the previous char or expression

# Operadores e Precedência



<b>RE</b>	<b>Match</b>	<b>First Patterns Matched</b>
<code>\*</code>	an asterisk “*”	“K_A*P*L*A*N”
<code>\.</code>	a period “.”	“Dr. Livingston, I presume”
<code>\?</code>	a question mark	“Why don’t they come and lend a hand?”
<code>\n</code>	a newline	
<code>\t</code>	a tab	

# Teste de avaliação em aula



1. Selecionar todas as cadeias alfabéticas

`[a-zA-Z]+`

# Teste de avaliação em aula



2. Selecionar todas as cadeias alfabéticas (em minúscula) que terminam com a letra 'm'.

[a-z]\*m      ← Errado  
[a-z]\*m\b

# Teste de avaliação em aula



3. Selecionar todas as cadeias que começam no início da linha com um número inteiro e que terminam no final da linha com uma palavra.

Exemplo. No seguinte fragmento, apenas as linhas sublinhadas devem ser identificadas:

1223. Universidade  
1 Universidade Federal.  
A Universidade Federal do  
1 Universidade Federal.  
1223. Universidade Federal do ABC  
1223. Universidade Federal do ABC!

```
^[0-9]+\b.*\b[a-zA-Z]+$
```

```
^\d+\b.*\b\w+$
```

# Teste de avaliação em aula



4. Selecionar todas as cadeias alfabéticas que tenham as palavras 'gato' ou 'cachorro'.	<code>(gato)   (cachorro)</code>
5. Selecionar todas as cadeias alfabéticas que tenham as palavras 'gato gato' ou 'cachorro'.	<code>(gato\s){2}   (cachorro)</code>

# Teste de avaliação em aula



```
!r" R\$\d+(,\d{2})?\b
```

## TEST STRING

```
texto 1, texto 2, texto 3  
texto A, texto B, texto C  
texto A.1, texto B.1, texto C.1  
Valor R$14,34 valor R$18  
valor R$20,46 varor R$56,450
```

```
!r" [pP]rogramação\s(\w+)
```

## TEST STRING

```
Processamento da Informação  
Sistemas operacionais  
Técnicas Avançadas de Programação A  
Técnicas Avançadas de Programação B  
Programação Orientada a Objetos  
Programação Avançada para Dispositivos Móveis  
Programação Estruturada
```

```
!r" [pP]rogramação\s|w+
```

## TEST STRING

```
Processamento da Informação  
Sistemas operacionais  
Técnicas Avançadas de Programação A  
Técnicas Avançadas de Programação B  
Programação Orientada a Objetos  
Programação Avançada para Dispositivos Móveis  
Programação Estruturada
```

# Teste de avaliação em aula



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
```

# Greedy Vs Non-greedy

- Nas ER, por padrão são utilizadas estratégias gulosas (gananciosas, *greedy*).

REGULAR EXPRESSION 2 matches, 74 steps (~2ms)

`ir" pod.*\b` " gm

TEST STRING SWITCH TO UNIT TESTS ▶

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.**



# Greedy Vs Non-greedy

- O operador '?' é utilizado para modificar o \* (ou o +).

REGULAR EXPRESSION 2 matches, 76 steps (~1ms)

`ir" pod.*?\b` " gm

TEST STRING SWITCH TO UNIT TESTS ▶

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.



## **Um exemplo prático: Eliza**

# Sobre Eliza (chatbot)

- Foi um dos primeiros sistemas de PLN / MIT:1964-1966

*Weizenbaum, Joseph. (1966). **ELIZA – A computer program for the study of natural language communication between man and machine.** Communications of the ACM, 9(1), 36–45.*

- Programado para manter conversa bem '*limitada*' (superficial) com um usuário, **imitando** respostas.
- Basicamente são **procurados padrões textuais** na conversa.
- A resposta de Eliza seria uma **substituição** desses padrões, com a finalidade de criar uma **ilusão de entendimento** da conversa.

# Sobre Eliza (chatbot)

- No sistema são procuradas frases como por exemplo:

“Você está X”

- Sendo substituídas

“O que faz pensar que eu estou X?”

- Simulação de um terapeuta:

- As respostas são um espelho do pensamento do paciente.
- Na tentativa de os mesmos pacientes descobriam os motivos pelos quais eles gostariam de mudar.

# Sobre Eliza (chatbot)

- **Pessoa:** Os pais são todos iguais!!!
- **Eliza:** DE QUE FORMA?
  
- **Pessoa:** Eles estão sempre nos incomodando sobre algo
- **Eliza:** VOCÊ PODE PENSAR EM UM EXEMPLO ESPECÍFICO?
  
- **Pessoa:** Olha só, meus pais me fizeram vir aqui
- **Eliza:** SEUS PAIS FIZERAM VIR AQUI
  
- **Pessoa:** Eles falam que estou deprimido há muito tempo
- **Eliza:** LAMENTO OUVIR QUE VOCÊ ESTÁ DEPRIMIDO

# Proposta 'simples' em Python

```
1 import re
2
3 patterns = [
4     (r"\b(to|estou|eu estou)\b", "EU ESTOU"),
5     (r"\b(sou|eu sou)\b", "EU SOU"),
6     (r"\b(olha so,?) ", ""),
7     (r".*EU ESTOU (deprimido|triste|cansado).*", r"LAMENTO OUVIR QUE VOCE ESTA \1"),
8     (r".*EU SOU (infeliz).*", r"POR QUE VOCE ESTA \1?"),
9     (r".* todos .*", "DE QUE FORMA?"),
10    (r".* sempre .*", "VOCE PODE PENSAR EM UM EXEMPLO ESPECIFICO?"),
11    ]
12
13 while True:
14     comment = input()
15     response = comment.lower()
16     for pat, sub in patterns:
17         response = re.sub(pat, sub, response)
18     print (response.upper())
```

# Chat bot comparative table

<https://docs.google.com/spreadsheets/d/1RgG-dRS42EHIG7QdJOTg2ZO587KutTTPeUfyxVKoIn8/edit#gid=0>

Bot Name	Platform	Features	Programming languages / Apps / Integration	Technical details	License	Languages	Project Link	Channels	Clients/Fields	More information
IBM Watson Conversation Service		Built on a neural network (one billion Wikipedia words).  Has three main components: Intents, Entities, Dialog	Node SDK Java SDK Python SDK iOS SDK Unity SDK		Free Standard Premium  <a href="https://www.ibm.com/watson/developercloud/conversation.html#pricing-block">https://www.ibm.com/watson/developercloud/conversation.html#pricing-block</a>	English, Japanese	<a href="https://www.ibm.com/watson/developercloud/conversation.html">https://www.ibm.com/watson/developercloud/conversation.html</a>	speech image text	Healthcare, Finance, Legal, Retail, Fantasy Football	<a href="https://www.youtube.com/watch?v=1rT1WEbg5U">https://www.youtube.com/watch?v=1rT1WEbg5U</a>
AgentBot	Aivo's own natural language processing technology.	Understands natural language. Memory to maintain coherence during long conversations. Gathers customer information to deliver customized solutions. Continuous evolution. Clarifies intent.	Use our REST API to integrate with your CRM and other platforms.	Integrates with any CRM, internal system, human chat and third party application.	<a href="http://agentbot.net/en/request-a-demo/">http://agentbot.net/en/request-a-demo/</a>	English, Spanish, Portuguese	<a href="http://agentbot.net/en/">http://agentbot.net/en/</a>	voice or messenger channel	Telecommunications and Cable Operators E-Commerce and Online Services Banks and Financial Services Government	<a href="https://www.youtube.com/watch?v=KEHMP6TkbSU">https://www.youtube.com/watch?v=KEHMP6TkbSU</a>
Twyla	A proprietary AI platform.	Learns from agent/customer live chats.  Blends machine learning and rule-based methods.  Answers questions, deflects tickets.	Analyses data either via the API of your helpdesk or chat solution, or a secure file upload.	Is integrated with most major cloud Helpdesk and Live Chat solutions, like Zendesk, Salesforce and Liveperson. So no new processes or software.	Twyla is Software as a Service (AI-as-a-Service) and so comes at a monthly subscription cost, with no up-front setup or installation fees.	English	<a href="https://www.twylahelps.com/">https://www.twylahelps.com/</a>	Web Facebook Telegram  Through messenger apps and live chat	Automation and self-service customer support.	<a href="https://www.youtube.com/watch?v=An4UmvgAx0Q">https://www.youtube.com/watch?v=An4UmvgAx0Q</a>
Pypestream	Pypestream's Smart Messaging Platform.	Pypestream uses a patented framework of 'Pypes' and 'Streams'.  Natural Language Processing and keyword parsing.	The Smart Messaging Framework Pypeconnect SDK Pypemanager The Pypestream mobile app API plug-ins and integrations	An open and flexible API platform allows for custom integrations and development of 3rd party connectors, plugins and extensions.	Contact Pypestream to obtain current pricing.	English	<a href="https://www.pypestream.com/">https://www.pypestream.com/</a>	Pypestream mobile app  Brand apps/SMS  Web chat  Messenger  IoT  website Pype	By April 2016, the company had 500 businesses signed up and using the messaging platform, including Washington Gas and Billboard.	<a href="https://www.youtube.com/watch?v=nPpXUxx7I2w">https://www.youtube.com/watch?v=nPpXUxx7I2w</a>

- Na atualidade existem diferentes sistemas de chat desenvolvidos com técnicas sofisticadas para atendimento, por exemplo, para clientes de um negócio.
- Certamente Eliza foi um sistema pioneiro (baseado em busca de padrões)



# **Pesquisa científica com ERs?**

# Identificando pessoas importantes?

This is a preprint of an article accepted for publication in *Scientometrics*.  
DOI 10.1007/s11192-013-1091-8

## Extracting and quantifying eponyms in full-text articles

Guillaume Cabanac

Received: May 22, 2013 / Accepted: July 9, 2013

**Abstract** Eponyms are known to praise leading scientists for their contributions to science. Some are so widespread that they are even known by laypeople (e.g., Alzheimer's disease, Darwinism). However, there is no systematic way to discover the distributions of eponyms in scientific domains. Prior work has tackled this issue but has failed to address it completely. Early attempts involved the manual labelling of all eponyms found in a few textbooks of given domains, such as chemistry. Others relied on search engines to probe bibliographic records seeking a single eponym at a time, such as Nash Equilibrium. Nonetheless, we failed to find any attempt of eponym quantification in a large volume of full-text publications. This article introduces a semi-automatic text mining approach to extracting eponyms and quantifying their use in such datasets. Candidate eponyms are matched programmatically by regular expressions, and then validated manually. As a case study, the processing of 821 recent *Scientometrics* articles reveals a mixture of established and emerging eponyms. The results stress the value of text mining for the rapid extraction and quantification of eponyms that may have substantial implications for research evaluation.

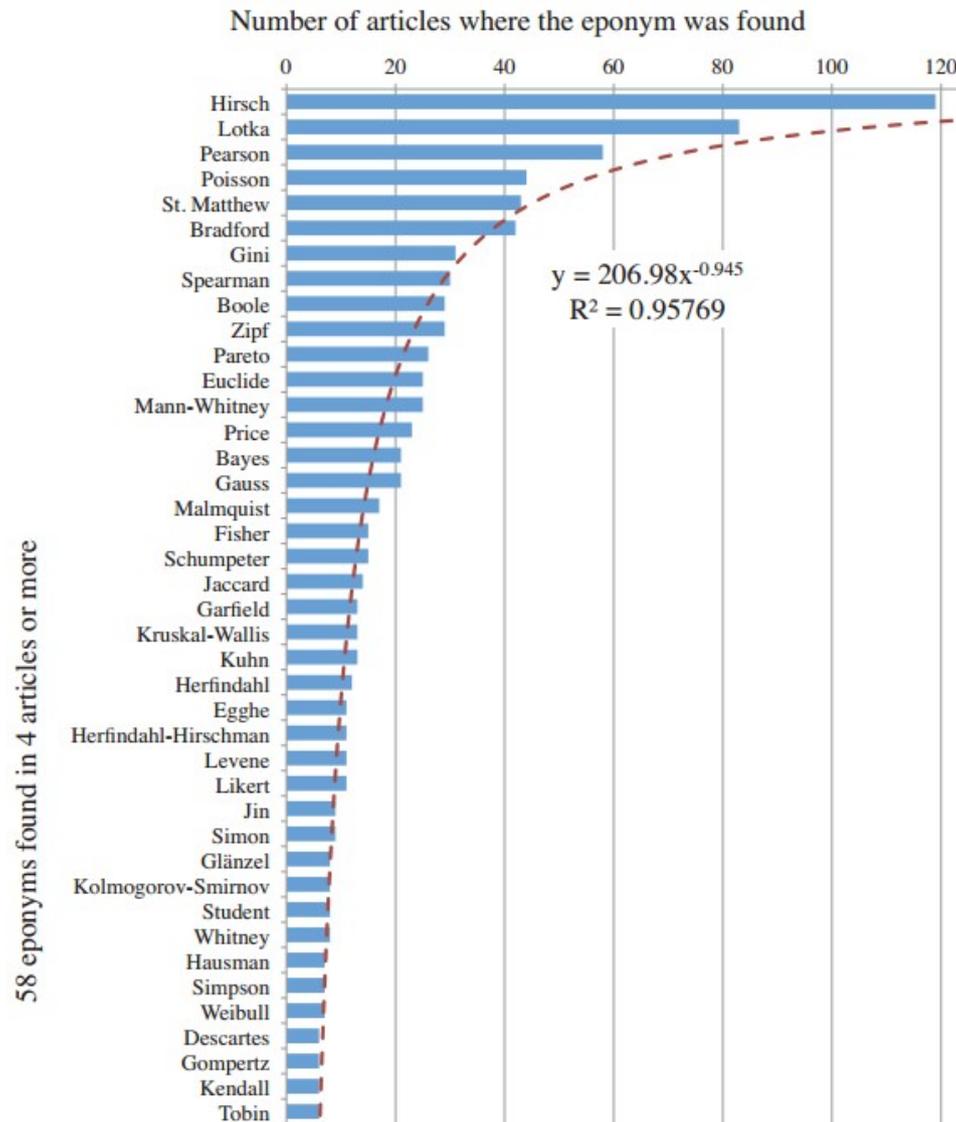
**Keywords** Eponymy · Text Mining · Regular Expressions · Academic Publications

*I have long worshiped the eponym as one of the last vestiges of humanism remaining in an increasingly numeralized and computerized society.*  
(Robertson 1972)

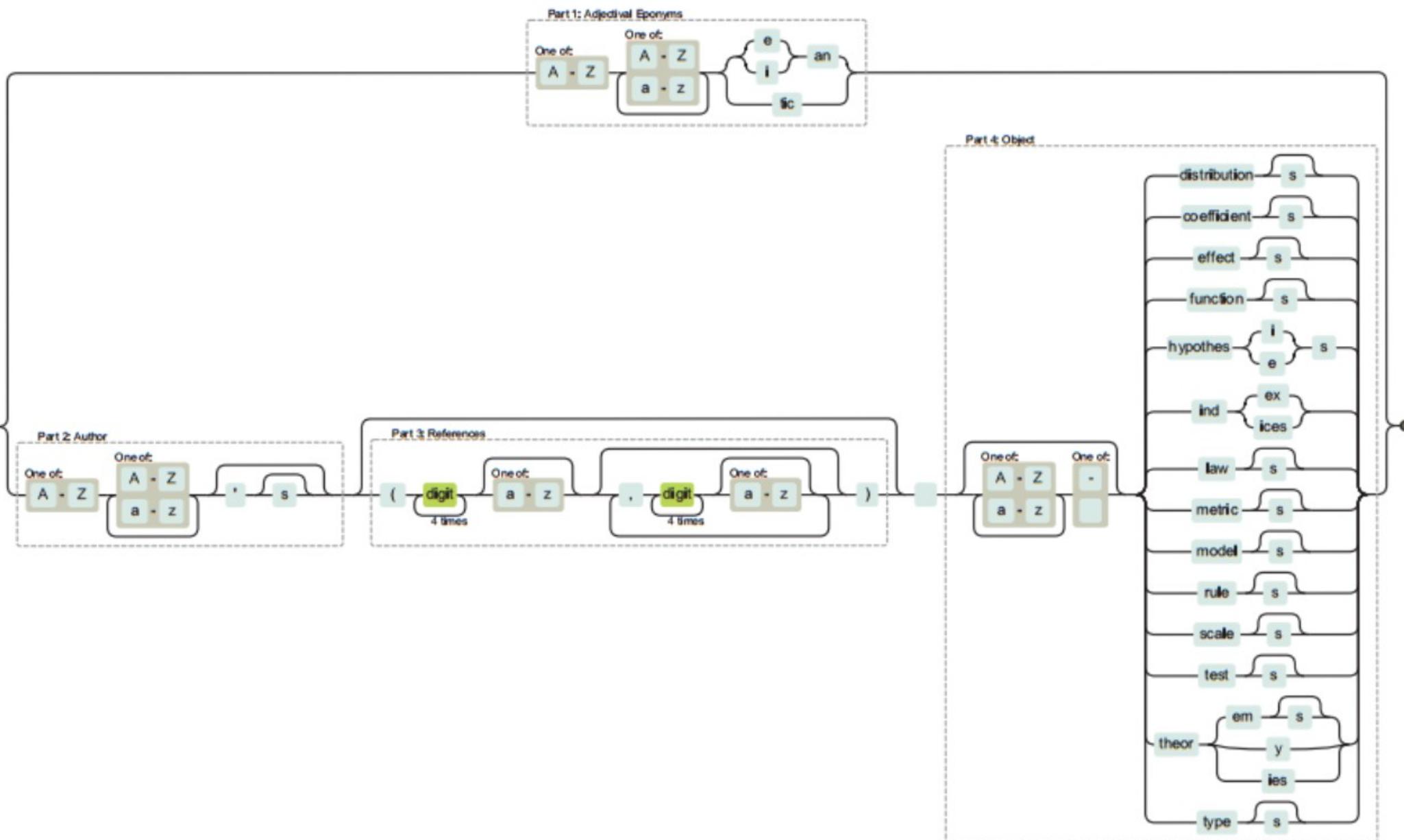


**Fig. 4** Hall of Fame for the eponymised persons extracted from 821 *Scientometrics* articles published in 2010–2013 (see Figure 3). This word cloud was produced by [www.wordle.net](http://www.wordle.net).

# Identificando pessoas importantes?



# Identificando pessoas importantes?



# Identificando pessoas importantes?

## Appendix: Computer program for eponym extraction and quantification

Listing 1 Bash script processing a corpus of full-text articles to extract and quantify eponyms

```
#!/bin/bash
# Extracts Eponymic Expressions from Textual Documents (e.g., Hirsch's h index)
# Requires GNU Coreutils software (/opt/local/libexec/gnubin in the path)
# License: Creative Commons Attribution-ShareAlike 3.0 Unported License.
# (see http://creativecommons.org/licenses/by-sa/3.0)
# @author Guillaume Cabanac (guillaume.cabanac@univ-tlse3.fr)
# @version 14-MAY-2013

# Parts of the regular expression matching eponyms ($RE_EPONYMS)
RE_ADJECTIVE="[A-Z][A-Za-z]+(?:(:e|i)an|tic)"
RE_AUTHOR="[A-Z][A-Za-z]+(?:s)?"
RE_REFERENCE="(?:_\d{4}[a-z]?(?:,_\d{4}[a-z]?+\\))?"
RE_OBJECT="(?:[A-Za-z]+[_])?(?:i:distributions?|coefficients?|effects?|functions?| >
  hypotheses|i:es|ind(?:ex|ices)|laws?|metrics?|models?|rules?|scales?|tests?| >
  theor(?:ems?|y|ies)|types?)"
RE_EPONYMS="$RE_ADJECTIVE|$RE_AUTHOR$RE_REFERENCE_$RE_OBJECT"

# Matches non-eponyms
RE_NO_EPONYMS="(?:All|An|Her|His|In|Its|The|These|This|Thus|Two|Our|We|When|While)\\ >
  s+"

# Create a temp file where eponymic expressions found in articles are concatenated
EXTRACTED_EPONYMS=$(mktemp)

# Extract eponymic expressions from each article. Concatenate in $EXTRACTED_EPONYMS
for ARTICLE in $(ls ScimPapers/s11192-*.txt) ; do
  sed -r "s/[[:blank:]]+/_/g" $ARTICLE | # Drop multiple whitespaces
  sed "s/[^A-z0-9(),_]/g" | # Drop accents (e.g., Glänzel -> Glnzl)
  grep -Pho "$RE_EPONYMS" | # Search for eponyms
  grep -Pv "$RE_NO_EPONYMS" | # Skip non-eponyms
  grep -vif nonEponymicAdjectives.txt | # Skip non-eponymic adj. (e.g., Belgian)
  sort | # Sort eponyms (mandatory before uniq)
  uniq -i >> $EXTRACTED_EPONYMS # Drop duplicated entries & append to file
done

# Rank eponymic expressions wrt their number of occurrences in distinct articles
cat $EXTRACTED_EPONYMS | # Output all eponymic expressions
sed -r "s/[^A-z]+/_/g;s/_/_/g" | # Only keep letters (Pearson's = Pearson)
sort | # Sort (mandatory before uniq)
uniq -ic | # Drop duplicates & count number of occurrences
sort -rk 1 | # Rank the list of eponymic expressions found

rm $EXTRACTED_EPONYMS # Remove temporary file
```



# **Sobre o resumo da aula de hoje**

# Resumo 2 - Tidia

- **Uma breve descrição da aula:**
  - Expressões regulares.
  - Sobre o teste de avaliação.
- **Deadline (daqui a 48h)**
  - Aprox. 250-500 palavras. Apenas texto.
  - Dia: 08/jun, até às 23h50.
  - Resumo submetido no prazo ~= resumo aprovado.
  - Todos os resumos serão publicados na página da disciplina seguindo um “ranking”.