

# Processamento de Linguagem Natural (PLN) – aula II

por: Rafael Stoffalette João

Data: 10/10/2020

UNiversidade Paulista (UNIP) – Araçatuba  
Pós-graduação em Data Science e Machine Learning

# Agenda da disciplina

**Módulo 01 - Introdução ao Processamento de Linguagem Natural**

**Módulo 02 - Análise Semântica e Morfológica**

**Módulo 03 - Processo de Mineração de Texto**

**Módulo 04 - Modelagem Estatística da Linguagem**

**Módulo 05 - Word Embeddings**

**Módulo 06 - Classificação de Texto**

**Módulo 07 - Extração de Informação**

**Módulo 08 - Geração de Resumo**

**Módulo 09 - Análise de Sentimentos**

**Módulo 10 - Deep Learning aplicado ao Processamento de Linguagem Natural**

## No encontro de hoje...

- Aquisição da informação;
- Crawler;
- Spider;
- Chatbot.

# Aquisição de informação na web

Existem várias formas de recuperar informação da web.

Uma consulta deve ser parametrizada por uma sentença de busca/um termo.

Um token extraído de uma sentença pode ser verificado quanto à sua importância e ser um termo de busca.

# Aquisição de informação na web

Aprendemos como recuperar tweets a partir da API do Twitter.

```
#https://www.earthdatascience.org/courses/use-data-open-source-python/intro-to-apis/twitter-data-in-python/
import os
import tweepy as tw
import pandas as pd

APIkey= '2xmZzKzI5XSydWu7bF0nAuF0n'
APIkeysecret= 'drzoukxLeVRnXGcqbZehepnWClVS1gHWkg4jvBmG7l3Z5ZSp2'

access_token= '1307181393479380993-bbejpWItu84p6KLSRdRTqT2Nmv4L49'
access_token_secret= 'qxstwRWL7hF9vYKLND1r0aCIHDylz8extTN7LK60wP6Ms'

auth = tw.OAuthHandler(APIkey, APIkeysecret)
auth.set_access_token(access_token, access_token_secret)

api = tw.API(auth, wait_on_rate_limit=True)

termoDeBusca = "robôs" #é a classe do 'sentimento'
a_partir_de = "2020-09-19"

tweets = tw.Cursor(api.search, #vou fazer uma busca no api do Twitter
                    q=termoDeBusca, #com o termo que está na variável termoDeBusca
                    lang="pt", #somente em portugues do Brasil
                    since=a_partir_de).items(10) #e a partir desta data na variavel a_partir_de

localizacao = [[tweet.user.screen_name, tweet.user.location, tweet.created_at, tweet.text, termoDeBusca] for tweet in
pdTweet = pd.DataFrame(data=localizacao, columns=['Nome', "Localização", 'Data', 'Tweet', 'classe'])

pdTweet
```

# Aquisição de informação na web

Mas e quando não temos uma API ?

Podemos realizar uma ‘consulta geral’ na web.

Uma ótima ferramenta é a BeautifulSoup do python

```
from bs4 import BeautifulSoup
```

```
# !pip3 install bs4
```

```
from bs4 import BeautifulSoup
```

```
pagina = BeautifulSoup(page.content,'html')
```

```
print(pagina.title)
```

<http://localhost:8888/notebooks/PLN/aula03/bs4.ipynb>

```
from bs4 import BeautifulSoup
```

**Note que...**

```
driver.get(url)
```

**inicialmente carrega a página toda como no seu browser, não só o html.**



# Sabe o que acabamos de fazer?

Um varredor de internet....

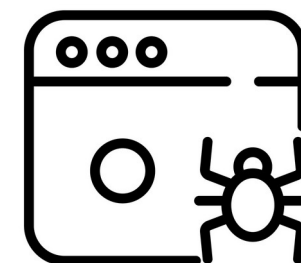
Também chamado web Crwaler...

# Aquisição de informação na web

**O Web crawler (Crawler, Spider ou Bot) nada mais é que um algoritmo utilizado para adquirir e analisar o código da web**

Entendido como um robô que varre a Internet em busca de informações de um tema

- captura informações das páginas e cadastra os links identificados para facilitar encontrar outras páginas semelhantes
- mantém sempre sua base de dados atualizada.



# Aquisição de informação na web

## O web crawler do Google se chama Googlebot.

Rastrear as páginas públicas de toda a internet.

Os resultados são adicionados ao índice do Google, o processo também conhecido como **indexação**.

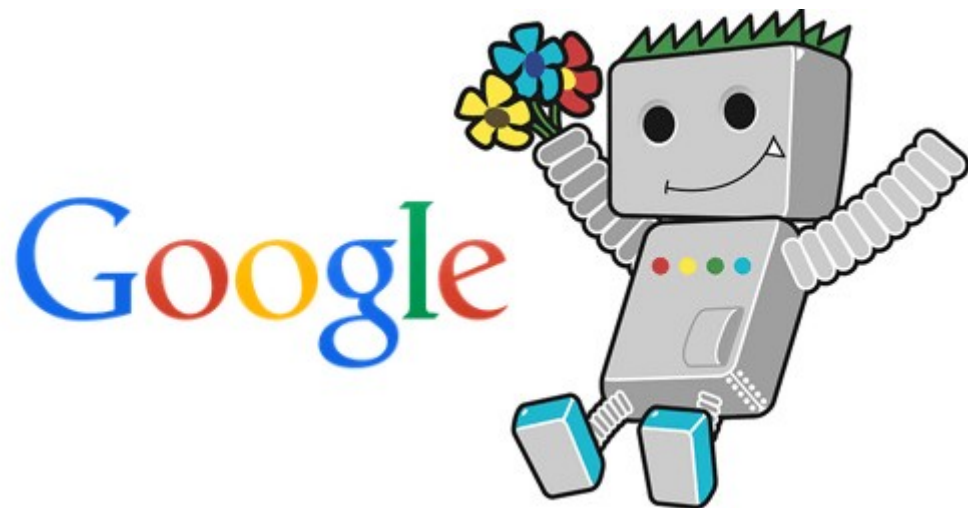
O índice funciona simplesmente como um banco de dados do mecanismo de busca.

É justamente lá que ficam armazenadas as informações que o Googlebot conseguiu examinar nos canais pelos quais passou, como a URL, título e o conteúdo textual.

```
<meta name="robots" content="noindex" />
```

```
<meta name="googlebot" content="noindex" />
```

```
<meta name="googlebot" content="all" />
```



# Aquisição de informação na web

Por meio da tag meta robots você consegue indicar ao Google como e SE você quer que seu site seja apresentado nos resultados das buscas no Google

```
<!DOCTYPE html>
<html><head>
<meta name="robots" content="noindex" />
(...)
</head>
<body>(...)</body>
</html>
```

O name (robots) especifica que a diretiva é aplicada a todos os rastreadores.

Para um rastreador específico, deve indicar o nome do rastreador (user agents)

Ex: do Google tem o nome de user agent Googlebot

<https://support.google.com/webmasters/answer/1061943?hl=pt-br>

```
<meta name="googlebot" content="noindex" />
```

# Aquisição de informação na web - Diretivas

## **all**

Não há restrições para a indexação nem a exibição. Essa diretiva é o valor padrão e não terá efeito se for listada explicitamente.

## **noindex**

Não mostrar esta página nos resultados da pesquisa.

## **nofollow**

Não seguir os links nesta página.

## **none**

É equivalente a noindex, nofollow.

## **noarchive**

Não exibir um link em cache nos resultados da pesquisa.

## **nosnippet**

Não exibir um snippet de texto ou uma visualização de vídeo nos resultados da pesquisa para esta página. Uma miniatura de imagem estática (se disponível) continuará visível, se essa opção resultar em uma melhor experiência do usuário. Isso será aplicado a todas as formas de resultados da pesquisa (na Pesquisa Google na Web, no Imagens do Google e no Discover).

# Aquisição de informação na web

**Um webcrawler como o googlebot pode ser construído a partir de:**

- A entrada do usuário é tokenizada;
- Os tokens são normalizados;
- As entidades reconhecidas;
- Os termos são stemmizados/lemmatizados (opcional)(expressões/gírias?)
- para cada token/entidade com tf-idf alto, uma varredura é feita na web.

Essa varredura pode ser prévia e a consulta apenas em um banco de dados.

# Aquisição de informação na web – web crawler

Tudo começa com uma lista de URLs (seeds) a serem visitadas pelo web crawler.

Em toda visita a cada um desses sites, o robô identifica os links das páginas e os inclui em listas específicas para uma nova varredura.

O robô respeita as tags meta para definição da vontade do dono do site. (não os nossos! =D )

Então, volta a elas recursivamente, conforme as regras estabelecidas.

E se encontra conteúdo novo, ele é indexado. Caso localize conteúdo atualizado em uma página preexistente, a sua classificação nos buscadores é que pode ser alterada.

# Aquisição de informação na web – web crawler

Crawler:

<https://www.google.com/search?q=lego>

Já reparou na classe `yuRUBf` ???

...E se a gente testasse nosso bs4 com resultados do Google?

<http://localhost:8888/notebooks/PLN/aula03/bs4.ipynb>



# Ops!!!

Páginas que são construídas dinamicamente (javascript) não são formadas por tags html.

E aí?

Como recuperar uma informação que não existe?

CALMA, nem tudo está perdido....

# Pacote Selenium do Python

O Selenium é um biblioteca utilizada para testes automatizados de navegadores.

O webdriver do Selenium é usado para criar uma instância de um navegador.

Portanto ele cria um site html a partir dos resultados dinâmicos gerados pelo site original...

# Selenium

1) !pip3 install selenium

2) Baixar o driver selenium para controlar o navegador

<https://selenium-python.readthedocs.io/installation.html#drivers>

3) encontrar a pasta PATH do navegador com o comando no jupyter

```
import os
```

```
import sys
```

```
os.path.dirname(sys.executable)
```

4) Colar o driver baixado na pasta PATH

# Selenium

```
!pip3 install selenium
```

```
Requirement already satisfied: selenium in /home/rafaelstojoao/.local/lib/python3.7/site-packages (3.141.0)  
Requirement already satisfied: urllib3 in /usr/lib/python3/dist-packages (from selenium) (1.24.1)
```

```
|  
from selenium import webdriver  
browser = webdriver.Chrome()  
browser.get('http://www.unip.br')  
browser.close()
```

Webdriver é o simulador do navegador que será utilizado.  
É possível utilizar outros navegadores, como o Firefox()

# Selenium

```
!pip3 install selenium
```

```
Requirement already satisfied: selenium in /home/rafaelstojoao/.local/lib/python3.7/site-packages (3.141.0)  
Requirement already satisfied: urllib3 in /usr/lib/python3/dist-packages (from selenium) (1.24.1)
```

```
|  
from selenium import webdriver  
browser = webdriver.Chrome()  
browser.get('http://www.unip.br')  
browser.close()
```

`find_elements_by_id('nome_id')`

<http://localhost:8888/notebooks/PLN/aula03/Selenium1.ipynb>

# Crawler com selenium

Agora sim...

... Sabemos como simular uma página gerada de forma dinâmica para uma estrutura HTML básicas;

Conseguimos, então, capturar elementos e valores...

O que nos impede de fazer um crawler?

<http://localhost:8888/notebooks/PLN/aula03/Selenium.ipynb>

# Um pouco mais de rastejamento

[http://localhost:8888/notebooks/PLN/aula03/spacy%20token\\_sentence.ipynb](http://localhost:8888/notebooks/PLN/aula03/spacy%20token_sentence.ipynb)

# Pacote Scrapy do python

**O scraping é um processo em dois passos:**

Você encontra e faz o download de páginas web sistematicamente.

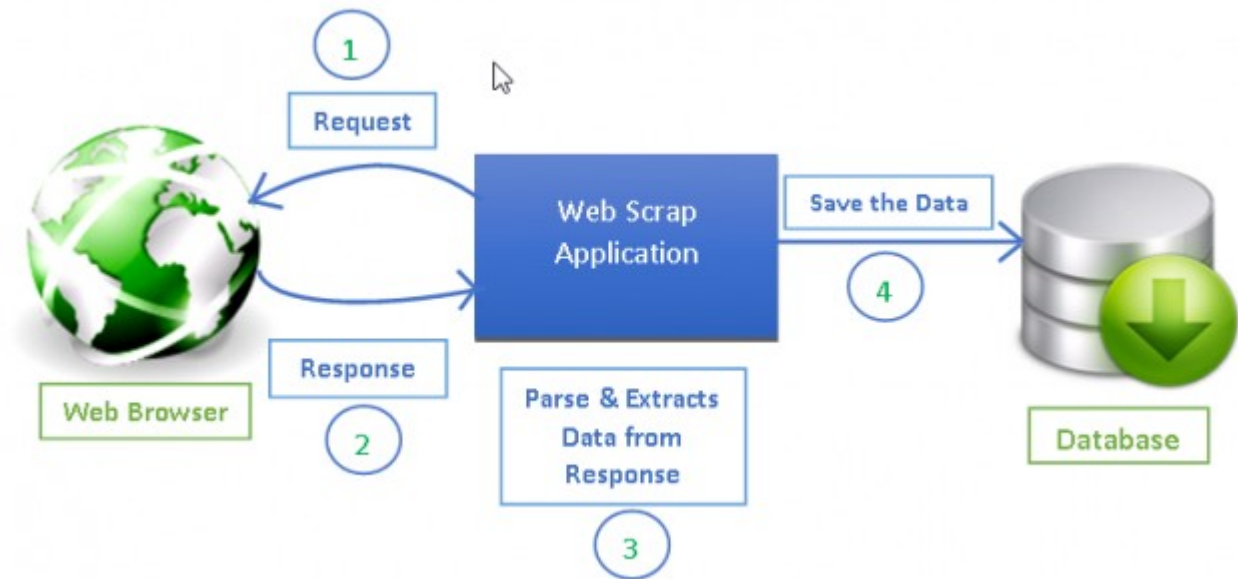
Você pega essas páginas web e extrai informações delas.

**O Scrapy é uma das bibliotecas de scraping mais populares e poderosas do Python; ele usa uma abordagem de “pilhas incluídas” para scraping, o que significa que ele lida com muitas das funcionalidades comuns que todos os scrapers precisam para que os desenvolvedores não tenham que reinventar a roda a cada vez. Isso torna o scraping um processo rápido e divertido!**



# Pacote Scrapy

`!pip3 install scrapy`



# Pacote Scrapy

Scrapy usa um spider que é autocontido na biblioteca para buscar e manipular informações.

# Pacote Scrapy

**No terminal, digite:**

```
scrapy shell
```

**Um ambiente de execução do scrapy será criado.**

**O comando fetch é quem busca todas as informações na web.**

```
fetch('http://www.unip.br')
```

# Pacote Scrapy

**O comando fetch é quem busca todas as informações na web.**

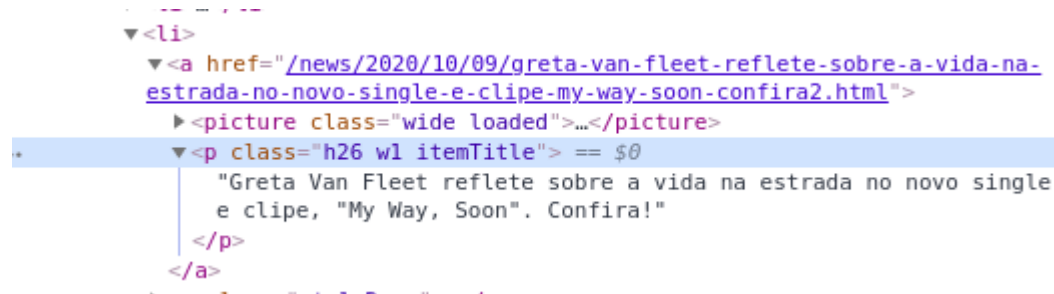
```
fetch('http://www.vagalume.com.br')
```

Com o comando `view(response)` é possível ver a página que foi garimpada.

Com o comando `print(response.text)` é possível ver toda a página em texto.

# Pacote Scrapy

Observando o código, é possível ver que os títulos seguem a organização de aparecer dentro da classe itemTitle.

A screenshot of a web browser's developer tools showing the HTML structure of a list item. The code is as follows:

```
<li>  
  <a href="/news/2020/10/09/greta-van-fleet-reflete-sobre-a-vida-na-  
estrada-no-novo-single-e-clipe-my-way-soon-confira2.html">  
    <picture class="wide loaded">...</picture>  
    <p class="h26 w1 itemTitle"> == $0  
      "Greta Van Fleet reflete sobre a vida na estrada no novo single  
e clipe, "My Way, Soon". Confira!"  
    </p>  
  </a>
```

The line containing the `<p class="h26 w1 itemTitle">` tag is highlighted in blue. The text inside the paragraph tag is visible as "Greta Van Fleet reflete sobre a vida na estrada no novo single e clipe, "My Way, Soon". Confira!".

Com o comando:

```
response.css(".itemTitle::text").extract_first()
```

É possível identificar o primeiro título listado

```
response.css(".itemTitle::text").extract()
```

Mostra TODOS os títulos

# Pacote Scrapy

**/ indica filho direto da tag. Se for interessante ver TODOS os itens de lista li da página é só rodar o comando...**

```
response.xpath('/html//li').extract()
```

/

Todos os li, mesmo que fora de html:

```
response.xpath('//li').extract()
```

## O que faz?

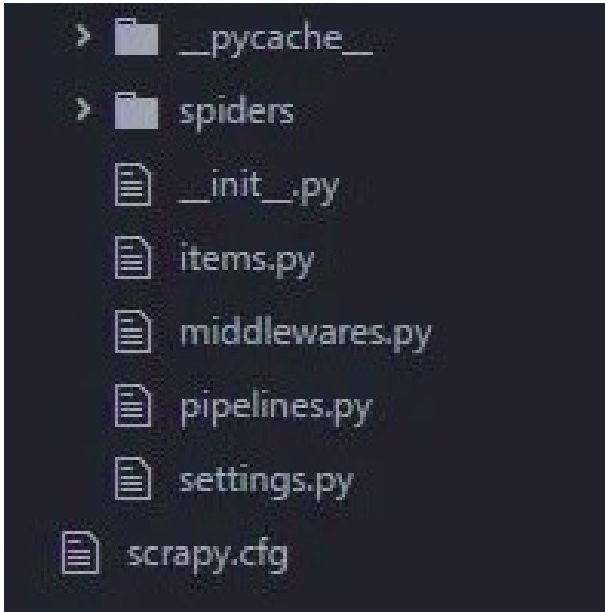
```
response.xpath("//li/a/p[@class='h26 w1 itemTitle']/text()").extract()
```

# Pacote Scrapy

Iniciando um projeto scrapy

`scrapy startproject unipSpider`

Uma ramificação de diretórios será criada



```
> __pycache__  
> spiders  
__init__.py  
items.py  
middlewares.py  
pipelines.py  
settings.py  
scrapy.cfg
```

# Pacote Scrapy

Com o comando:

Scrapy genspider aranha\_unip <http://www.vagalume.com.br>

Um arquivo Spider é construído para buscar informações no site indicado.

O nome aranha\_unip é uma sugestão

```
rafaelstojoao@stojoao: /mnt/DADOS/DOCS/Unip/codes/PLN/aula03/scrapyUnip$ scrapy genspider aranha_unip http://www.vagalume.com.br
/home/rafaelstojoao/.local/lib/python2.7/site-packages/OpenSSL/crypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported by the Py
thon core team. Support for it is now deprecated in cryptography, and will be removed in a future release.
  from cryptography import x509
Created spider 'aranha_unip' using template 'basic'
```



# Pacote Scrapy

O arquivo `aranha_unip.py` é criado e pode ser alterado para:

```
# -*- coding: utf-8 -*-
import scrapy

class AranhaUnipSpider(scrapy.Spider):
    name = 'aranha_unip'
    allowed_domains = ['http://www.vagalume.com.br']
    start_urls = ['http://http://www.vagalume.com.br/']

    def parse(self, response):

        titulos = response.css(".itemTitle::text").extract()
        yield titulos
```

Para salvar os resultados buscados, edite o arquivo `settings.py` para incluir as linhas:

```
FEED_FORMAT="csv"
```

```
FEED_URI="musicas.csv"
```

# Pacote Scrapy

Ao executar o comando:

```
scrapy crawl
```

As informações são buscadas e salvas no arquivo musicas.csv

Agora podem ser processadas da melhor forma.

```
aranha_unip.py  aranha_unip.pyc  __init__.py  __init__.pyc  musicas.csv  
rafaelstojao@stojaoa:/mnt/DADOS/DOCS/Unip/codes/PLN/aula03/scrapyUnip/unipSpider/unipSpider/spiders$
```

# Chatbot

Conceito que define a interação entre humano e computador em formato de texto, onde o humano e o computador aprendem...



# Chatbot

O chatbot se embasa no conceito de casamento de padrões.

Quando muitas sentenças do tipo: “o livro da anne frank é bom”

“ eu gosto do livro Narnia”, “nos tempos livres eu gosto de ler livros”,...

Os padrões (repetições) vão sendo observados e armazenados em bancos de dados de conhecimento...

Um chatbot age por meio de uma base de conhecimento, na qual padrões são identificados para cada resposta esperada.

# Chatbot

<https://iaexpert.academy/2016/10/18/historico-da-ia/>

<https://www.masswerk.at/elizabot/>

# Chatbot com Chatterbot

**Biblioteca pythin para construção simples de um chatbot.**

<https://github.com/anikethsukhtankar/ChatterBot/blob/master/readme.pt.md>

<http://localhost:8888/notebooks/PLN/aula03/chatbot.ipynb>

<http://localhost:8888/notebooks/PLN/aula03/botUnip.ipynb>



# Chatterbot

Ao conversar com o botinhoUnip, uma sentença inserida por um usuário/visitante pode conter uma entidade.

Quando o botinho não reconhece a entidade, ele pode buscar informações a respeito dela e indicar ao usuário uma lista indexada de links nos quais existem informações a respeito dessa entidade que foi informada.

## Por exemplo:

Usuário: Você conhece o Obama?

BotinhoUnip: hmmm, não sei muito sobre ele, mas achei alguns links que podem nos ajudar.  
\$link1, \$link2,....