SÃO PAULO
GOVERNO DO ESTADO
SÃO PAULO SÃO TODOS

DESENVOLVIMENTO DE SOFTWARE MULTIPLATAFORMA

Disciplina: ISW-039 - Mineração de Dados

Aula 07: Web Scraping.

Data 19/10/2024

Prof. Me. Anderson Silva Vanin

WEB SCRAPING
RASPAGEM DE DADOS

1. WEB SCRAPING

A aula sobre "**Introdução à Raspagem de Dados em Sites da Internet**" apresenta um conjunto de técnicas para extrair e processar dados de páginas web de maneira automatizada. Aqui, introduzimos o conceito de *web scraping*, discutimos as ferramentas principais que serão usadas, como *requests*, *BeautifulSoup* e *pandas*, e definimos o objetivo do encontro: capacitar os alunos a extrair dados estruturados da web e manipulá-los para análise.

2. O Que é Raspagem de Dados?

Tecnicamente a raspagem de dados pode ser definido como o **processo de realizar requisições automatizadas a páginas web para extrair informações estruturadas**. Explicamos como a raspagem permite converter informações não estruturadas (em HTML) em dados prontos para análise. Exibimos alguns cenários de uso típicos, como o monitoramento dinâmico de preços, coleta de dados para aprendizado de máquina e análise de tendências em redes sociais, destacando o valor de utilizar dados extraídos para insights de mercado e tomadas de decisão automatizadas.

3. Como a Raspagem Funciona?

Fluxo de trabalho da raspagem de dados em quatro etapas técnicas:

1. **Requisição HTTP**: Usamos a biblioteca *requests* para realizar uma requisição *GET* e obter o HTML completo da página de destino.
2. **Parsing HTML**: Com o *BeautifulSoup*, fazemos o *parsing* (análise sintática) do HTML para navegar pela árvore de elementos e selecionar apenas os dados desejados.
3. **Extração de Dados**: Usamos métodos como *.find()* e *.find_all()* para localizar e extrair informações específicas, como textos ou atributos.
4. **Armazenamento e Manipulação**: Estruturamos os dados em um formato legível, como um *DataFrame* do pandas, que facilita análises estatísticas e visualizações.

4. Cuidados com Raspagem

A raspagem de dados levanta questões éticas e legais, que são abordadas aqui com uma visão técnica sobre limitações e práticas recomendadas:

- **Respeito ao arquivo robots.txt**: Este arquivo orienta *bots* e *crawlers* sobre áreas permitidas e restritas para acesso. Ensina-se como verificar *robots.txt* de um site e interpretá-lo para respeitar suas diretrizes.
- **Compliance com Termos de Serviço**: Explicamos que raspagem em alguns sites pode violar os Termos de Serviço e que é importante verificar esses documentos para garantir conformidade legal.
- **Práticas de Raspagem Ética**: Para evitar o bloqueio do servidor e possíveis restrições, recomendamos introduzir atrasos entre as requisições usando bibliotecas como *time.sleep()*, e configurar cabeçalhos HTTP adequados, como *User-Agent*, para simular acessos reais.

5. Ferramentas e Bibliotecas para Raspagem em Python

Bibliotecas essenciais para raspagem em Python:

- **requests**: Envia requisições HTTP simples e confiáveis, oferecendo métodos como *GET, POST, PUT e DELETE*. Vamos detalhar o uso de *requests.get()* para obter HTML de uma URL.
- **BeautifulSoup**: Explicamos que o *BeautifulSoup* faz *parsing* do HTML em uma árvore de objetos, permitindo navegar pela hierarquia com métodos como *.find()* e *.select()*.
- **Pandas**: Exibimos como essa biblioteca converte dados em tabelas organizadas, chamadas *DataFrames*, facilitando operações de filtragem, agrupamento e visualização de dados.

6. Estrutura HTML Básica

Para entender o HTML é necessário compreender como localizar dados dentro de uma estrutura hierárquica. Neste slide, introduzimos elementos HTML com foco técnico:

- **Tags HTML**: Apresentamos tags HTML como <*div*>, <*h2*>, e <*p*>, que definem diferentes tipos de conteúdo.
- **Classes e IDs**: Explicamos como classes *(class)* e IDs *(id)* ajudam a identificar e extrair dados específicos. Cada elemento pode ser encontrado através de seu atributo exclusivo para uma extração precisa de dados.
- **Análise de Estrutura HTML**: Mostramos um exemplo em HTML que simula a estrutura de produtos com nome, preço e descrição, ensinando a reconhecer as informações desejadas dentro da árvore HTML para futura extração.

7. Exemplo Prático de Raspagem

Este slide mostra um exemplo prático para ilustrar a aplicação das técnicas de raspagem:

- **Passo 1**: Com *requests.get()*, fazemos uma requisição *GET* à URL de interesse para obter o HTML da página.
- **Passo 2**: Utilizando o *BeautifulSoup*, realizamos o *parsing* do HTML e buscamos elementos usando *.find_all()* para identificar o conteúdo específico dos produtos.
- **Passo 3**: Iteramos por esses elementos para extrair os valores textuais e atributos relevantes, armazenando cada item em listas para organização futura.

8. Estruturando Dados em um DataFrame

Após a raspagem, o próximo passo é estruturar os dados de forma organizada para análise:

- **Criação do DataFrame**: Usamos listas para armazenar os dados raspados (nome, preço e descrição) e criamos um *DataFrame* com *pandas.DataFrame()*.
- **Tipos de Dados e Limpeza**: Mostramos como converter dados raspados em tipos adequados, como *strings* e *floats*, e discutimos a importância de limpar dados, por exemplo, removendo símbolos de moeda antes de conversões numéricas.
- **Manipulação e Análise**: Introduzimos funções básicas de manipulação no pandas, como *head()*, *info()*, e *describe()*, para explorar o *DataFrame* e garantir que os dados estejam prontos para análise.

9. Demonstração Completa

Este slide une todos os passos anteriores em um exemplo completo e integrado:

1. **Extração dos Dados**: Aplica o código para extrair informações de uma página e transformar em listas organizadas.
2. **Conversão para DataFrame**: Estrutura os dados em um *DataFrame* para organização e análise.
3. **Manipulação e Visualização**: Realiza operações como filtragem, agrupamento e, opcionalmente, visualização gráfica com *MatplotLib*. Um exemplo é a criação de um histograma de preços, útil para identificar tendências de mercado.

10. Cuidados e Considerações Finais

Considerações éticas e práticas:

- **Validação e Manutenção de Código**: É necessário validar dados para garantir precisão e adaptar o código caso o site atualize sua estrutura HTML.
- **Alternativas à Raspagem**: Em alguns casos, a ***API*** oficial de um site oferece uma alternativa segura e confiável para obtenção de dados, evitando desafios técnicos e riscos legais.
- **Práticas para Raspagem Escalável**: Para grandes volumes de dados, recomendam-se ferramentas robustas como *Scrapy*, que possui um motor de *scraping* assíncrono e é capaz de manusear volumes maiores de requisições.

PARTE PRÁTICA

1 - Criando o Site de Exemplo

O site será simples, contendo uma estrutura em HTML com informações básicas de produtos, como:

- **Nome**
- **Preço**
- **Descrição**

1 - Criando o Site de Exemplo

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Loja de Exemplo</title>
</head>
<body>
    <h1>Produtos Disponíveis</h1>
    <div class="produto">
        <h2>Produto 1</h2>
        <p>Preço: R$20,00</p>
        <p>Descrição: Um produto de alta qualidade.</p>
    </div>
    <div class="produto">
        <h2>Produto 2</h2>
        <p>Preço: R$45,00</p>
        <p>Descrição: Perfeito para o uso diário.</p>
    </div>
</body>
</html>
```

1 - Criando o Site de Exemplo

Este arquivo HTML pode ser hospedado em um repositório no *GitHub Pages* para facilitar o acesso. As instruções para configurar o *GitHub Pages* são as seguintes:

1. Crie um repositório no *GitHub* com o nome, por exemplo, ***site-exemplo-web-scraping***.
2. Envie o arquivo ***index.html*** para o repositório.
3. Acesse as configurações do repositório e habilite o *GitHub Pages*.
4. Assim que configurado, você terá uma URL (por exemplo, ***https://seu-usuario.github.io/site-exemplo-web-scraping/***) para o site de exemplo.

2 - Raspagem de Dados no Google Colab

```
!pip install requests
!pip install beautifulsoup4
```

```
import requests
from bs4 import BeautifulSoup

# URL do site de exemplo hospedado no GitHub Pages
url = 'https://seu-usuario.github.io/site-exemplo-web-scraping/'

# Fazendo a requisição HTTP para obter o conteúdo da página
response = requests.get(url)
```

```
# Verificando se a requisição foi bem-sucedida
if response.status_code == 200:
    print("Conexão bem-sucedida!")
    # Criação do objeto BeautifulSoup para parsing do HTML
    soup = BeautifulSoup(response.text, 'html.parser')

    # Encontrando todos os produtos na página
    produtos = soup.find_all('div', class_='produto')

    # Extraindo e exibindo as informações dos produtos
    for produto in produtos:
        nome = produto.find('h2').text
        preco = produto.find('p').text
        descricao = produto.find_all('p')[1].text

        print("Nome:", nome)
        print("Preço:", preco)
        print("Descrição:", descricao)
        print("-" * 20)
else:
    print("Erro ao acessar o site.")
```

3 - Explicação do Código

- ***requests.get(url)***: faz a requisição à URL especificada e retorna o conteúdo da página.
- ***BeautifulSoup(response.text,'html.parser')***: interpreta o HTML usando o *BeautifulSoup*.
- ***soup.find_all('div', class_='produto')***: localiza todas as *divs* com a classe produto, onde cada uma representa um produto.
- Itera sobre os produtos e extrai os dados.

4 – TESTANDO A RASPAGEM

- Execute o código no Google Colab e confirme se os dados são exibidos corretamente.
- Modificar o HTML ou ajustar o código para raspar outras informações, como preços ou descrições.

Alguns sites para web scraping e suas limitações

Aqui estão cinco sites em português onde é possível praticar *web scraping*. Eles oferecem dados variados, como cotações de moedas, condições climáticas e informações de livros. Também detalho as restrições de cada site conforme especificado em seus arquivos ***robots.txt***:

Alguns sites para web scraping e suas limitações

1. Banco Central do Brasil

- **Descrição**: Oferece dados sobre cotações de moedas estrangeiras e outros indicadores econômicos.
- **URL**: https://www.bcb.gov.br/
- **Restrição de *robots.txt***:

➢ O Banco Central bloqueia a raspagem de muitas seções. O arquivo *robots.txt* impede acesso a várias *URLs* internas específicas, embora algumas áreas públicas, como as páginas de cotações, estejam disponíveis para leitura.

➢ **Aconselhamento**: Para acessar dados de câmbio e indicadores, é preferível usar a API do Banco Central (link para API), que foi desenvolvida exatamente para esse tipo de uso.

Alguns sites para web scraping e suas limitações

2. Climatempo

- **Descrição**: Provedor de informações climáticas para cidades do Brasil, como previsão de temperatura, umidade e condições do tempo.
- **URL**: https://www.climatempo.com.br/
- **Restrição de robots.txt**:

➢ O Climatempo limita o acesso a várias páginas internas e seções privadas. O *robots.txt* permite o acesso às previsões climáticas básicas, mas é restrito para áreas de login, APIs internas e conteúdo exclusivo.

➢ **Aconselhamento**: É recomendável usar o site com parcimônia para scraping educativo e sempre inserir delays entre requisições para evitar sobrecarga. Verifique o site para evitar bloqueios e excesso de acessos, pois eles possuem uma API paga para previsões climáticas.

Alguns sites para web scraping e suas limitações

3. IBGE (Instituto Brasileiro de Geografia e Estatística)

- **Descrição**: Oferece uma variedade de dados estatísticos sobre o Brasil, como censos, índices econômicos, dados demográficos, entre outros.
- **URL**: https://www.ibge.gov.br/
- **Restrição de robots.txt**:

➢ O arquivo robots.txt do IBGE permite acesso a várias páginas públicas de dados, mas restringe o acesso a certas áreas internas e de administração.

➢ **Aconselhamento**: É recomendável utilizar a API do IBGE (link para API) para acessar dados de forma estruturada e evitar a sobrecarga do site com raspagem direta.

Alguns sites para web scraping e suas limitações

4. Mercado Livre Brasil

- **Descrição**: Marketplace com uma variedade de produtos, permitindo a prática de raspagem de dados de produtos para estudo de e-commerce.
- **URL**: https://www.mercadolivre.com.br/
- **Restrição de robots.txt**:

➢ O Mercado Livre possui um robots.txt bastante restritivo, que bloqueia muitas das áreas do site e várias URLs de produtos e categorias.

➢ **Aconselhamento**: **A prática de scraping direto no Mercado Livre pode resultar em bloqueio.** O ideal é utilizar a API oficial do Mercado Livre (link para API), que fornece acesso controlado e legal aos dados do marketplace.

Alguns sites para web scraping e suas limitações

5. Skoob

- **Descrição**: Rede social de livros em português, com informações de títulos, autores, resenhas e avaliações, ideal para prática de raspagem de dados de produtos culturais.
- **URL**: https://www.skoob.com.br/
- **Restrição de robots.txt**:

➢ O robots.txt do Skoob permite a indexação de várias páginas públicas de livros e autores, mas bloqueia áreas privadas, como perfis de usuários e seções de login.

➢ **Aconselhamento**: A prática de scraping no Skoob deve ser realizada apenas nas páginas públicas, e com um limite de requisições para evitar bloqueios. Atualmente, o Skoob não oferece uma API, então qualquer raspagem precisa ser feita de forma controlada e respeitosa.

Esses sites são úteis para aprender scraping, mas é fundamental respeitar as restrições do robots.txt de cada um e usar APIs sempre que disponíveis para evitar bloqueios e problemas com os Termos de Serviço. A prática ética garante que a raspagem seja feita de forma responsável e dentro dos limites permitidos.

SITES PARA WEB SCRAPING LIVRES

Alguns sites foram criados especificamente para prática de web scraping, sem limitações e com conteúdo fictício. Eles são ideais para fins de aprendizado e possuem uma estrutura adequada para simular raspagem em ambientes reais. Aqui estão alguns dos mais utilizados:

SITE PARA WEB SCRAPING LIVRE

1. Books to Scrape

- **Descrição:** Um site fictício com uma coleção de livros organizados por categorias, cada um com título, preço, disponibilidade e uma imagem de capa. Foi criado para simular uma loja de livros e permite a raspagem sem restrições.
- **URL:** http://books.toscrape.com/
- **Conteúdo:** Ideal para aprendizado de raspagem de e-commerce e manipulação de dados tabulares.
- **Nota:** O site é seguro para raspagem e permite práticas variadas sem limite de requisições.

Exercício

Utilize o site http://books.toscrape.com/ , realize um web scraping neste site a fim de responder as seguintes questões:

1. **Preço médio dos livros**: Mostra o preço médio de todos os livros listados.
2. **Contagem de livros disponíveis**: Informa quantos livros estão disponíveis no estoque.
3. **Livros mais caros**: Exibe os 3 livros com os preços mais altos na lista.

Referências

- Documentação Oficial do BeautifulSoup:

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

- Documentação do Requests:

https://docs.python-requests.org/en/master/

- Documentação do Pandas:

https://pandas.pydata.org/docs/

- Google Colab Help Center:

https://research.google.com/colaboratory/faq.html

- Real Python - Web Scraping com BeautifulSoup:

https://realpython.com/beautiful-soup-web-scraper-python/

- Kaggle:

https://www.kaggle.com/

- Stack Overflow:

https://stackoverflow.com/