Extração e Tratamento de Dados com Python: Uma Aplicação com Web Scraping e Análise de Livros

Allyson de Almeida Sirvano

19 de maio de 2025

1 Introdução

O presente trabalho tem como objetivo demonstrar um fluxo completo de extração, tratamento, análise e exportação de dados utilizando a linguagem Python. Os dados analisados referem-se a livros extraídos do site `books.toscrape.com`. O processo envolve bibliotecas modernas como `Playwright`, `Pandas`, `Seaborn`, `Matplotlib` e `OpenPyXL`.

2 Bibliotecas Utilizadas

2.1 Playwright

A biblioteca `Playwright` é uma ferramenta moderna para automação de navegadores. Seu principal diferencial é o suporte a múltiplos navegadores e a execução assíncrona. No contexto deste projeto, foi utilizada para navegar em páginas web e extrair informações dos livros.

Comandos essenciais utilizados:

- `playwright.chromium.launch()`: inicia o navegador Chromium.
- `page.goto(url)`: navega até a URL desejada.
- `page.query_selector()`: seleciona elementos HTML.
- `page.eval_on_selector_all()`: extrai listas de elementos via JavaScript.

2.2 Pandas

A biblioteca `pandas` foi empregada para estruturar os dados em um `DataFrame`, o que facilita operações como ordenação, filtragem e estatísticas descritivas.

Comandos principais:

- `pd.DataFrame()`: cria o objeto com os dados extraídos.
- `df.to_excel()`: exporta o DataFrame para uma planilha.
- `df['coluna'].value_counts()`: análise da distribuição de valores.

2.3 Matplotlib e Seaborn

Estas bibliotecas foram utilizadas para a criação de gráficos de barras e gráficos de pizza com o objetivo de visualizar a distribuição das avaliações dos livros.

Exemplos utilizados:

- `plt.subplots()`: estrutura a área de plotagem.
- `sns.barplot()`: gráfico de barras com contagem de avaliações.
- `plt.pie()`: gráfico de setores com percentual.

2.4 OpenPyXL

A biblioteca `openpyxl` foi empregada para manipular planilhas Excel diretamente, adicionando fórmulas, indicadores e gráficos aos arquivos gerados.

Funções aplicadas:

- `load_workbook()`: carrega o arquivo Excel.
- `sheet.cell()`: manipula células.
- `BarChart, PieChart, Reference`: criação de gráficos dinâmicos.

3 Fluxo do Projeto

3.1 Extração de Dados

A função `extrair_dados_livros()` utiliza a biblioteca `Playwright` para iterar sobre os livros da página e coletar título, preço, estoque e avaliação.

```
page.goto('https://books.toscrape.com/')
livros_links = page.eval_on_selector_all('.product_pod h3 a', '''
    (elements) => elements.map(element => ({
        title: element.getAttribute('title'),
        href: element.href
    }))
''')
```

Listing 1: Extração de dados com Playwright

3.2 Indicadores de Performance

Os principais indicadores calculados a partir dos dados foram:

- **Percentual de livros bem avaliados**: livros com avaliação 4 ou 5.
- **Percentual de estoque crítico**: livros com estoque menor ou igual a 5.
- **Preço médio dos livros bem avaliados**.

3.3 Visualização de Dados

Gráficos foram gerados tanto no terminal (com `Seaborn` e `Matplotlib`) quanto na planilha Excel (com `OpenPyXL`).

```
sns.barplot(
    x=distribuicao.index,
    y=distribuicao.values,
    palette="muted",
    ax=axes[0]
)
```

Listing 2: Gráfico de barras com Seaborn

```
axes[1].pie(
    percentual.values,
    labels=[f'{p} estrelas ({v:.1f}%)' for p, v in zip(percentual
        .index, percentual.values)],
    startangle=140,
    colors=sns.color_palette("pastel", len(percentual))
)
```

Listing 3: Gráfico de pizza com Matplotlib

3.4 Exportação e Formatação em Excel

Utilizou-se a função `tratar_dados_excel()` para adicionar indicadores e gráficos diretamente na planilha, facilitando a visualização e interpretação dos dados por outros usuários.

```
chart_bar = BarChart()
data = Reference(sheet, min_col=2, min_row=5, max_row=10)
labels = Reference(sheet, min_col=1, min_row=5, max_row=10)
chart_bar.add_data(data, titles_from_data=True)
chart_bar.set_categories(labels)
sheet.add_chart(chart_bar, "E5")
```

Listing 4: Criação de gráfico no Excel com openpyxl

4 Conclusão

O desenvolvimento deste código tem como objetivo, além da extração automatizada de dados por meio de uma biblioteca de Web Scraping, a aplicação de bibliotecas especializadas em análise e visualização de dados. Após a coleta das informações do site, torna-se essencial representá-las de forma estruturada e visualmente clara, a fim de viabilizar interpretações precisas e subsidiar a tomada de decisões com base nos dados obtidos.