PYTHON ACADEMY

PYTHON E LANGCHAIN: WEB SCRAPING COM IA

Nesse ebook, você vai desvendar como usar Langchain e Inteligência Artificial para extrair conteúdos de páginas web

PYTHONACADEMY.COM.BR

Salve salve Pythonista 👋

No mundo digital de hoje, o **Web Scraping** é uma técnica essencial para extrair informações de sites.

Com a evolução da inteligência artificial, surge a possibilidade de incorporar agentes inteligentes através de Frameworks LLM, como o **Langchain**, e automação de navegadores, como o **Playwright**, para enriquecer esse processo.

Neste artigo, exploraremos como combinar essas ferramentas para realizar web scraping de maneira eficiente e inteligente.

Vamos entender o que são o Langchain e o Playwright, como configurá-los e implementá-los em um projeto de web scraping.

Além disso, utilizaremos outras bibliotecas úteis e o ChatGPT para analisar os dados extraídos.

Vamos lá!

O que é Langchain?

O **Langchain** é uma biblioteca que simplifica a criação de pipelines de processamento de linguagem natural sofisticados.

Ela permite que desenvolvedores combinem diferentes modelos e técnicas de forma modular.

Utilizando o Langchain, podemos facilmente misturar e combinar modelos de IA para obter uma análise aprimorada do texto extraído durante o web scraping.

Se quiser saber mais sobre o LangChain, clique aqui para ler um artigo completo com mais detalhes sobre essa formidável ferramenta.

O que é Playwright?

Playwright é uma biblioteca para automação de navegadores, semelhante ao Selenium, mas com desempenho melhorado e suporte a multiplataforma.

Ele permite controlar navegadores para extrair dados dinâmicos de sites de forma eficiente, essencial quando lidamos com páginas que carregam dados via JavaScript.

O langchain utiliza o Playwright para carregar páginas dinâmicas e extrair o conteúdo HTML para análise.

Bibliotecas Necessárias

Para implementar nosso projeto de Web Scraping com IA, precisamos instalar as seguintes bibliotecas:

```
pip install playwright \
  beautifulsoup4 \
  python-decouple \
  langchain \
  langchain-community \
  langchain-openai
```

Explicação das Bibliotecas

- **playwright**: Automação de navegador para carregamento de páginas dinâmicas.
- **beautifulsoup4**: Extração de dados do HTML.
- **python-decouple**: Carregar variáveis de ambiente de arquivos `.env`.

- **langchain**: Construção de pipelines de linguagem natural.
- **langchain-community**: Funcionalidades adicionais para o Langchain.
- **langchain-openai**: Integração com a API da OpenAI.

Implementação do Web Scraping com IA

Vamos implementar o processo de web scraping com as etapas detalhadas a seguir.

Configuração Inicial

Primeiro, vamos configurar nossa chave de API e o modelo de linguagem natural que usaremos:

Crie um arquivo `.env` na raiz do projeto com a chave da API da OpenAI:

```
from decouple import config
import os

# Carrega a chave da API da OpenAI do arquivo .env usando decouple
os.environ['OPENAI_API_KEY'] = config('OPENAI_API_KEY')
```

Definição do Modelo

Definimos o modelo para estruturar os dados do artigo, utilizando o Pydantic.

Estruturamos o modelo do artigo com campos como título, descrição, data, autor e link:

```python
from pydantic import Field, BaseModel

class Article(BaseModel):
    """
    Representa um artigo com informações estruturadas.
    """
    headline: str = Field(description="O título do artigo")
    description: str = Field(description="Uma breve descrição do artigo")
    date: str = Field(description="A data em que o artigo foi publicado")
    author: str = Field(description="O autor do artigo")
    link: str = Field(description="O link para o artigo")
```

Carregamento de Dados

Usamos o **AsyncChromiumLoader** do Langchain para carregar o conteúdo HTML das URLs. Ele carrega o conteúdo HTML das URLs especificadas:

```python
from langchain_community.document_loaders import AsyncChromiumLoader

# Lista de URLs que serão carregadas
urls = ["https://pythonacademy.com.br/blog/"]

loader = AsyncChromiumLoader(urls)
docs = loader.load()
```

Extração com BeautifulSoup

Com o **BeautifulSoupTransformer**, extraímos as tags desejadas:

Dentro da função `transform_documents`, passamos a lista de documentos carregados e as tags que queremos extrair (como “article”) para obter as tags você precisa inspecionar o HTML da página e identificar as tags que contêm o conteúdo desejado.:

```
from langchain_community.document_transformers import BeautifulSoup-
        Transformer

# Utiliza BeautifulSoupTransformer para extrair apenas as tags deseja-
        das ("article")
bs_transformed = BeautifulSoupTransformer()
docs_transformed = bs_transformed.transform_documents(
  documents=docs,
  tags_to_extract=["article"]
)
```

Tokenização de Texto

Usamos o **RecursiveCharacterTextSplitter** para tokenizar o texto entregue para análise:

```
from langchain_text_splitters import RecursiveCharacterTextSplitter

# Divide o conteúdo extraído em pedaços usando tokenização
splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
  chunk_size=2000,
  chunk_overlap=0
)
splits = splitter.split_documents(documents=docs_transformed)
```

Análise de Conteúdo

Finalmente, invocamos o modelo estruturado de linguagem natural para estruturar os dados:

```python
from langchain_openai import ChatOpenAI

# Cria uma instância do modelo ChatOpenAI configurado com GPT-4o-mini
llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)

# Configura o modelo para retornar respostas estruturadas no formato da
        classe Article
structured_llm = llm.with_structured_output(Article)

# Extrai e estrutura as informações usando o modelo configurado para
        cada pedaço
extracted_content = [structured_llm.invoke(split.page_content) for
        split in splits]

# Converte o conteúdo estruturado em uma lista de dicionários
extracted_content_dict = [article.model_dump() for article in extrac-
        ted_content]

# Exibe o resultado
print(extracted_content_dict)
```

Resultado da Extração de Dados

O código acima extrai e estrutura dados das páginas especificadas retornando uma lista de artigos:

```
[
  {'headline': 'Padrões de Projeto em Python (Design Patterns)',
  'description': 'Neste artigo, você aprenderá sobre Padrões de Proje-
to
  em Python (Design Patterns) e sua importância.',
  'date': '26/02/2025',
  'author': 'Vinícius Ramos',
  'link': 'https://pythonacademy.com.br/blog/padroes-de-projeto-em-py-
thon'},

  {'headline': 'O que é o Django Rest Framework (DRF)?',
  'description': 'Entenda o que é o Django Rest Framework (DRF) e como
ele
  pode facilitar o desenvolvimento de APIs REST em Django.',
  'date': '27/08/2024',
  'author': 'Vinícius Ramos',
  'link': 'https://pythonacademy.com.br/blog/o-que-e-o-django-rest-fra-
mework'}
]
```

Este resultado revela como o uso de IA pode facilitar e estruturar processos de extração de dados.

Código Completo do Web Scraping com IA

Aqui está o código completo para referência:

```python
from decouple import config
from langchain_community.document_loaders import AsyncChromiumLoader
from langchain_community.document_transformers import BeautifulSoup-
    Transformer
from langchain_openai import ChatOpenAI
from langchain_text_splitters import RecursiveCharacterTextSplitter
from pydantic import Field, BaseModel
import os

os.environ['OPENAI_API_KEY'] = config('OPENAI_API_KEY')

llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)

class Article(BaseModel):
    headline: str = Field(description="O título do artigo")
    description: str = Field(description="Uma breve descrição do arti-
        go")
    date: str = Field(description="A data em que o artigo foi publica-
        do")
    author: str = Field(description="O autor do artigo")
    link: str = Field(description="O link para o artigo")

structured_llm = llm.with_structured_output(Article)

urls = ["https://pythonacademy.com.br/blog/"]
tags_to_extract = ["article"]

loader = AsyncChromiumLoader(urls)
docs = loader.load()

bs_transformed = BeautifulSoupTransformer()
docs_transformed = bs_transformed.transform_documents(
  documents=docs,
  tags_to_extract=tags_to_extract
)

splitter = RecursiveCharacterTextSplitter.from_tiktoken_encoder(
  chunk_size=2000,
  chunk_overlap=0
)

splits = splitter.split_documents(documents=docs_transformed)
```

```
extracted_content = [structured_llm.invoke(split.page_content) for
        split in splits]

extracted_content_dict = [article.model_dump() for article in extrac-
        ted_content]

print(extracted_content_dict)
```

> ***Falando em LangChain e IA:*** *Estou usando exatamente essas tecnologias para construir o* ***DevBook*** *— uma plataforma que gera ebooks técnicos automaticamente. O mesmo poder do LangChain que você está aprendendo aqui, aplicado para criar conteúdo didático de alta qualidade. Confira!*

Conclusão

Neste artigo, mergulhamos na combinação poderosa do **Langchain** e do **Playwright** para realizar web scraping avançado com IA.

Estudamos como configurar e implementar estas ferramentas, além de outras bibliotecas essenciais, para extrair dados de sites de forma eficiente.

Com a capacidade de estruturar informações de maneira precisa, esse método transforma a maneira como abordamos a extração de dados na web.

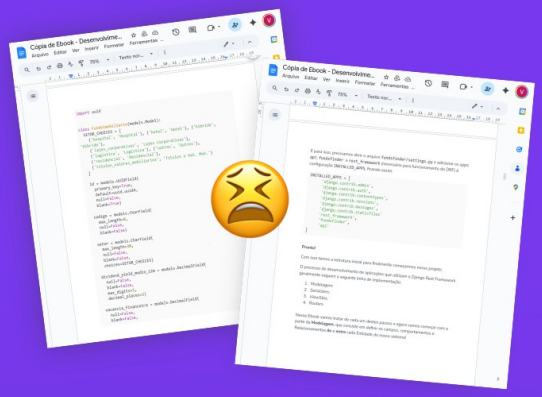
Experimente estas técnicas no seu próximo projeto e veja os benefícios dessa abordagem inovadora.

Não se esqueça de conferir!

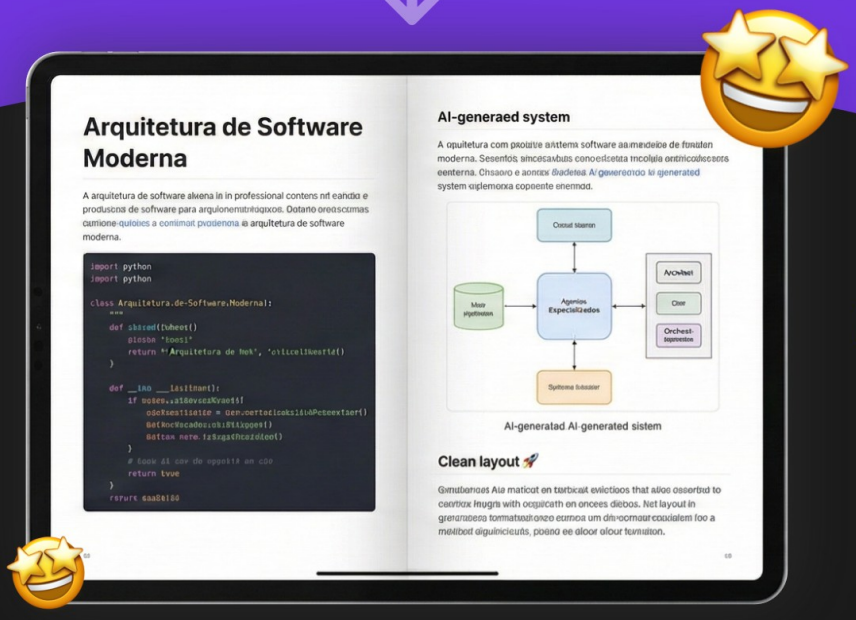
DevBook

Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!



Chega de formatar código no Google Docs



Syntax Highlight

Infográficos feitos por IA

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS