Relatório sobre Web Scraping e o Projeto Feeling AI – FATEC Rio Claro

Nisso aqui. O tema de hoje é web scrapping, sabe? Aquela técnica, a arte, talvez, de coletar dados da web de forma automática. E olha, a gente tem um material bem rico para analisar. Hoje tem uns relatórios super detalhados de um projeto de IA lá da Fatec Rio Claro, o projeto Feeling AI, do grupo Soul Care, né? Bem interessante esse projeto. Exato. E também uma visão geral da Wikipedia sobre o assunto para dar aquele contexto e até uns trechos de código em Python mesmo, mostrando como a coisa funciona na prática com BeautifulSoup, Selenium, ferramentas bem conhecidas na área e uma aplicaçãozinha Flask também, né, para fechar o ciclo. Isso. Então, a nossa missão aqui hoje é tentar desvendar o que é exatamente web scrapping, entender como as diferentes técnicas funcionam, desde o básico até as com Python, e discutir porque isso é tão relevante hoje em dia, especialmente para quem está aprendendo para projetos como esse da Fatec. E claro, não dá para fugir: encarar a parte mais delicada, as barreiras, a ética, a parte legal. Com certeza, a ideia é dar uma visão clara, prática. Exato. Direto ao ponto. É um campo fascinante porque mistura programação, análise de dados e, como você disse, até questões legais bem complexas. A gente vai explorar tanto a parte técnica, o como fazer, mas também o porquê isso importa tanto hoje.

Perfeito. Então vamos começar pelo começo. O que é afinal web scrapping? A Wikipedia define como a extração automatizada de dados de websites, usando programas, bots, crawlers. Não é simplesmente copiar e colar manualmente. Isso não escala. É usar software para fazer o trabalho pesado. Exato. E os usos são muitos. A Wikipedia lista várias aplicações: indexação para buscadores, como o Google começou, para mapear a web, comparação de preços online, monitoramento de notícias, de clima, coleta de contatos, pesquisa de mercado e, o que interessa muito para nossa conversa, mineração de dados para treinar inteligência artificial. Ah, aqui a coisa fica realmente interessante, porque o web scrapping muitas vezes é o primeiro passo fundamental para áreas como aprendizado de máquina e processamento de linguagem natural. Os relatórios da Fatec até citam fontes importantes para contextualizar isso. Para ensinar uma IA a entender, por exemplo, o sentimento num texto, que era o objetivo do Feeling AI, você precisa de quê? De dados. Muitos dados. Exemplos. Exatamente. Muitos exemplos de texto. E o web scrapping é a ferramenta que permite buscar essa matéria-prima em grande quantidade na web.

Faz todo sentido. É como ir “lapidar” os dados brutos. Mas tecnicamente, como isso funciona? A ideia básica tem duas etapas principais: primeiro buscar a página, basicamente baixar o código HTML dela, e depois extrair a informação que você quer daquele monte de código — o chamado parsing. E tem jeitos mais simples de fazer isso. A Wikipedia menciona copiar manualmente — às vezes não tem jeito — ou usar expressões regulares (regex) para achar padrões no texto do HTML. Funciona para coisas simples, mas o foco do material que a gente tem aqui está mais nas ferramentas de programação, principalmente Python.

Python domina essa área, tem bibliotecas excelentes, como a combinação Requests com BeautifulSoup. Requests pega a URL e baixa o HTML da página. É como tirar uma foto da página naquele momento. Gostei da analogia. E o BeautifulSoup pega essa foto, esse HTML bruto, e organiza. Transforma numa estrutura que o Python consegue navegar, entender e encontrar as partes que interessam. Tipo no exemplo: find_all('p') para achar todos os parágrafos ou find_all('h5') para os autores. Exato. E depois o .get_text() para pegar só o texto mesmo, limpo. No final o script salva tudo num arquivo CSV organizadinho.

Parece bem direto — e é, para páginas estáticas. Para essas, a dupla Requests e BeautifulSoup é ótima, rápida, eficiente. Mas a web hoje é super dinâmica. Tem muito site que carrega conteúdo depois com JavaScript, tipo feeds infinitos ou dashboards que atualizam sozinhos. A foto do Requests não pega isso. Não mesmo. Requests só vê o que veio na primeira resposta do servidor. Tudo que o JavaScript carrega depois, ele não vê. E aí entra o Selenium.

O pessoal da Fatec detalhou isso bem no relatório. O Selenium não tira só a foto — ele automatiza um navegador de verdade, Chrome, Firefox, o que você quiser. Controla o navegador. Pensa assim: não é alguém tirando a foto da vitrine, é um robozinho que entra na loja, aperta botões, espera as luzes acenderem, os produtos aparecerem, e só então anota o que viu. Muito mais poderoso. Ele consegue clicar, preencher formulários, rolar a página, esperar elementos aparecerem — tudo que um usuário faria. Mas precisa de um tal WebDriver, certo? Sim. O WebDriver é a ponte entre o código Python e o navegador. Cada navegador tem o seu.

E as “esperas” (waits) são importantes. Pensa bem: páginas dinâmicas não carregam instantaneamente. Se o script tentar pegar um dado que ainda não apareceu, dá erro. As esperas fazem o script aguardar até que o elemento desejado esteja disponível, tipo “espere até que tal botão esteja clicável”. Isso evita erros, mas deixa o script mais sensível a mudanças no site. Esse é o calcanhar de Aquiles do Selenium. Ele exige manutenção, porque se o site muda, o script quebra.

A Wikipedia ainda cita outras abordagens: agregação vertical, anotações semânticas e análise por visão computacional. Agregação vertical é quando existem plataformas que já fazem o scraping de nichos, tipo passagens aéreas. Anotações semânticas é quando o próprio site usa padrões como Schema.org para marcar dados — isso facilita muito a extração. E análise por visão computacional é usar IA para “enxergar” a página como um humano, útil para sites muito visuais.

Toda essa técnica é interessante, mas ganha mais vida quando a gente vê aplicada. E aí entra o projeto Feeling AI da Fatec Rio Claro, um ótimo exemplo acadêmico. O objetivo era usar o web scraping como ferramenta dentro de um projeto maior: a análise de sentimentos.

Os alunos do grupo SoulCare foram atrás mesmo: viram tutoriais online, leram documentação, usaram IAs como ChatGPT e Gemini para tirar dúvidas, e colaboraram

entre si. Mostraram autonomia e capacidade de aprender a aprender, essencial hoje em dia. Mesmo que Selenium não estivesse no currículo, eles buscaram porque o projeto pedia. Aprenderam fazendo — o web scraping não era o fim, mas o meio para atingir o objetivo: entender PLN na prática.

Mas há o lado ético e legal. A Wikipedia e o relatório da Fatec mencionam barreiras que os sites impõem — bloqueio de IP, CAPTCHAs e o arquivo robots.txt, que define o que os robôs podem ou não acessar. É uma etiqueta digital. A regra de ouro é a responsabilidade: respeitar o robots.txt, ler os termos de uso e agir dentro da lei.

A parte legal é complexa. Nos EUA há casos famosos como LinkedIn vs. hiQ Labs. Na Europa há o GDPR. E no Brasil, a LGPD. Mesmo dados públicos, se forem pessoais, exigem base legal e finalidade clara.

Por isso, o grupo SoulCare deixou claro no relatório o uso apenas educacional e ético das técnicas. Não é só saber fazer — é saber quando e por que fazer, e quais os limites.

Assim, o web scraping é uma ferramenta poderosa, mas que exige cautela. Usar para aprender, como a Fatec fez, está dentro dos limites éticos e legais. Usar de forma predatória, sem autorização, pode gerar riscos legais e de reputação.

Coletar os dados é só o começo — o valor aparece quando se usa esses dados. E aí entra o Flask, mostrado no código app.txt. Esse exemplo cria uma aplicação web simples: um formulário de reclamação que salva os dados num arquivo JSON e os exibe depois. Ele mostra o passo seguinte — transformar dados coletados em informação útil.

Imagina que as reclamações vieram de um site tipo Reclame Aqui, coletadas via scraping (respeitando as regras). O Flask organiza, armazena e exibe, podendo até gerar uma API que devolve os dados em JSON.

Assim, o ciclo se completa: Web Scraping $\rightarrow$ Armazenamento $\rightarrow$ Análise $\rightarrow$ Apresentação.

Em resumo: - Web Scraping coleta dados automaticamente da web. - Requests e BeautifulSoup servem para páginas estáticas. - Selenium é usado para páginas dinâmicas, mas exige manutenção. - O projeto Feeling AI é um exemplo de aplicação prática e ética. - A LGPD e o uso responsável dos dados são fundamentais.

No fim, o verdadeiro desafio não é só como fazer scraping, mas quando e por que. Usar essas ferramentas com inteligência e responsabilidade é o que faz a diferença.