



Instituto Tecnológico Autónomo de México

Diplomado: Ciencia de Datos y Machine Learning Aplicado a Finanzas

Módulo 3: Ciencia de Datos

Tema

Web Scraping

Nombre: Denzel Arik Martinez Maldonado



Hoy en día hay muchísima información en Internet: redes sociales, sitios web, portales, plataformas... y esa información puede servir para tomar decisiones, medir desempeño, encontrar patrones y hasta descubrir cosas que no sabíamos que existían. Justo por eso, el artículo plantea que en un mundo guiado por datos cada vez es más importante saber cómo extraer datos útiles de la web, porque eso alimenta directamente a la analítica de big data y a modelos de ciencia de datos.

A esta extracción se le suele llamar de varias formas: web scraping, web extraction, web harvesting, web crawling, etc. La idea es la misma: automatizar el proceso de entrar a páginas, encontrar la información importante y guardarla en un formato que sí podamos analizar por ejemplo tablas, CSV, JSON, bases de datos.

¿Para qué se usa el web scraping?

El artículo menciona usos muy comunes como comparar precios, monitorear clima, detectar cambios en páginas, juntar datos de distintas fuentes, extraer ofertas, scrapear vacantes, monitorear marcas y hacer análisis de mercado. Además, muestra ejemplos por dominio:

- **Salud**

Hay mucha información clínica pero recolectarla manualmente es pesado; entonces se propone automatizar la recolección, por ejemplo, en escenarios tipo SARS-CoV2 y también extraer documentos como folletos de medicamentos con crawlers.

- **Redes sociales**

Se usa mucho para marketing como saber el sentimiento del cliente, engagement, reputación, etc. Un ejemplo del artículo es un “descargador” de datos de cuentas de Instagram hecho con scraping para evitar restricciones del API, y luego exportar a Excel/JSON/CSV.

- **Finanzas**

Usa textos en línea de empresas para construir indicadores, por ejemplo, detectar empresas innovadoras y luego usar modelos para clasificar y comparar contra otras métricas como patentes o indicadores regionales.

- **Marketing**

Se usa para recolectar huellas digitales del consumidor o para alimentar modelos que predicen intención de compra cuando los datos están en formato no estructurado en la web y por eso conviene extraerlos primero.

Para este tipo de técnicas es importante fijar un proceso para no perderse, primero preguntas quieres contestar, qué datos necesitas, luego eliges las fuentes, después defines cómo extraer lo que necesitas y finalmente guardas y analizas.

Métodos para scrapear

Aquí lo importante es entender que no hay un solo scraping: depende de qué tan simple o compleja sea la página.

- **Copy & Paste**

Es lo más básico: copiar a mano. Obvio no escala, y si hay barreras o muchísimas páginas, se vuelve inviable.

- **HTML Parsing (Parseo del HTML)**

Muchas páginas se generan con plantillas. Entonces se puede reconocer el patrón y extraer bloques específicos del HTML usando “wrappers” o lenguajes de consulta para contenido semi-estructurado.

- **DOM parsing / HTML DOM**

La página se convierte a un árbol (DOM) y extraes secciones específicas. Esto da mucha flexibilidad porque si algo está en la página, en teoría lo puedes ubicar y extraer, sin esperar a que alguien te lo “publique” por otra vía.

- **Regex (Expresiones Regulares)**

Sirve cuando quieres encontrar patrones en texto (por ejemplo, fechas, correos, códigos). Es potente pero también fácil de romper si cambia la estructura.

- **XPath**

Es como dar direcciones dentro de un documento (HTML/XML): “ve al nodo tal”. Es muy útil cuando la estructura está clara y estable.

- **Plataformas de agregación vertical**

Son plataformas que construyen y monitorean bots en un sector específico, sin tanta intervención manual, y buscan escalar para muchos sitios.

- **Anotación semántica**

Si la página trae metadatos/etiquetas semánticas, el scraper puede apoyarse en eso para ubicar datos más fácil, como si la página te estuviera diciendo “esto es precio”, “esto es autor”, etc.

- **Computer vision para páginas**

En lugar de leer el HTML, analiza la página como imagen renderizada como la vería un humano y busca la estructura semántica visual.

Al comparar los métodos, Regex vs HTML DOM vs XPath, se encontro que Regex usa menos RAM, pero HTML DOM tarda menos y consume menos datos que Regex y XPath.

Crawlers

Un crawler es el bot que **visita páginas y saca enlaces** para seguir explorando. Empieza con pocas ligas, detecta más links y los mete a una lista tipo “frontera” para seguir. También tiene que saber manejar URLs relativas/absolutas y dar vueltas en lo mismo.

Tipos de crawlers:

- **Focused crawler:** busca páginas relacionadas a un tema específico.
- **Incremental crawler:** regresa seguido a ver cambios y mantener frescura.
- **Distributed crawler:** reparte el trabajo entre varios nodos coordinados.
- **Parallel crawler:** varios procesos a la vez para acelerar.
- **Hidden crawler:** intenta acceder a contenido “oculto” o no indexado.

Parsers y Políticas

Parsers: después de bajar contenido, se necesita transformarlo a algo usable como un HTML, PDF, CSV, JSON y muchos navegadores/librerías ya traen parsers integrados.

Políticas de crawling: para que un crawler funcione bien, el artículo menciona 4 políticas clave:

- **Selection:** elegir qué links sí valen la pena primero para no perder tiempo.
- **Re-visit:** volver a páginas si cambian, que son dinámicas.
- **Politeness:** no saturar servidores para no hacer miles de requests como loco.
- **Parallelization:** paralelizar con cuidado para hacerlo más eficiente.

Herramientas

Existen varias herramientas que nos ayudan a extraer la información, parsearla y convertirla en algo más usable y por tanto obtener un mejor análisis:

- **BeautifulSoup:** Útil para proyectos más chicos o de menor complejidad, pero puede ser más lento.
- **Scrapy:** lo pintan como la mejor opción en proyectos grandes o complejos, y una razón clave es que hace requests de forma asíncrona, lo que acelera mucho. Además se integra fácil con proxies/VPNs.
- **Selenium:** cuando el sitio depende mucho de JavaScript. Sirve para navegar y extraer partes específicas como si fueras un usuario.

Recomendación

El artículo menciona que muchos scrapers son similares y sirven para cosas generales, pero **Scrapy** da mejores resultados porque es rápido, extensible y potente. Al manejar requests asíncronos scrapea rápido, su arquitectura está basada en crawler y puedes extraer con selectores CSS/XPath. También facilita integrar proxies/VPNs.