Concepción, Santander 15 de septiembre de 2021.

Señores
Luis Alejandro Pico,
Camilo Andrés Suarez,
Yershon Estiven Caro,
David Alvis.
Tripulantes MISIONTIC
UIS - S38 – G5

Asunto: Entrega de Informe de investigación y análisis del método WEB-SCRAPING para extraer información de sitios web.

Debido a la necesidad de dar una información real de los vuelos y rutas de las aerolíneas nos vimos en la tarea de buscar alternativas para la extracción de datos de dichas aerolíneas, en la investigación realizada no encontramos con la metodología WEB-SCRAPING, a continuación, me permito relacionar la investigación realizada a dicha metodología:

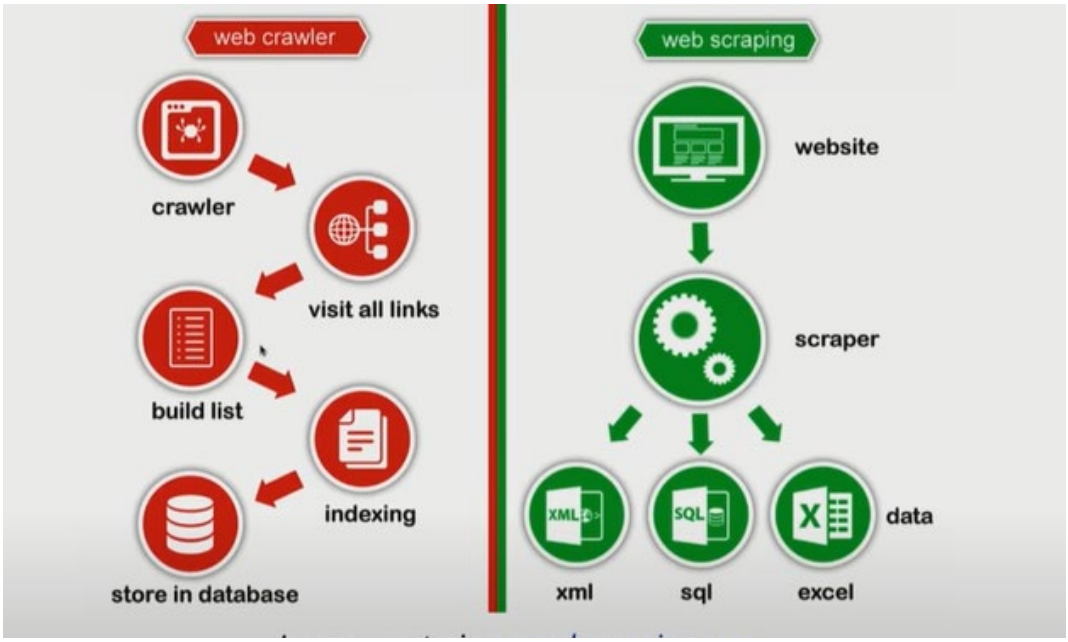
1. DOCUMENTACIÓN.

WEB-SCRAPING:

La World Wide Web está compuesta por muchos millones de documentos enlazados entre sí, conocidos también como páginas web. El texto fuente de las páginas web está escrito en el lenguaje Hypertext Markup Language (HTML). Los códigos fuente en HTML son una mezcla de informaciones legibles para los humanos y códigos legibles para las máquinas, llamados tags o etiquetas. El navegador, como puede ser Chrome, Firefox, Safari o Edge, procesa el texto fuente, interpreta las etiquetas y presenta al usuario la información que contienen.

Web scraping o raspado web, es una técnica utilizada mediante programas de software para extraer información de sitios web. Se trata de los programas llamados web scrapers, crawlers, spiders o, simplemente, bots, que examinan el texto fuente de las páginas en busca de patrones concretos. Usualmente, estos programas simulan la navegación de un humano en la World Wide Web ya sea utilizando el protocolo HTTP manualmente, o incrustando un navegador en una aplicación. El web scraping está muy relacionado con la indexación de la web, la cual indexa la información de la web utilizando un robot y es una técnica universal adoptada por la mayoría de los motores de búsqueda. Sin embargo, el web scraping se enfoca más en la transformación de datos sin estructura en la web (como el formato HTML) en datos estructurados que pueden ser almacenados y analizados en una base de datos central, en una hoja de cálculo o en alguna otra fuente de almacenamiento. Alguno de los usos del web scraping son la comparación de precios en tiendas, la monitorización de datos relacionados

con el clima de cierta región, la detección de cambios en sitios webs y la integración de datos en sitios webs.



2. PYTHON PARA WEB SCRAPING.

Python, el popular lenguaje de programación se presta especialmente bien para la creación de programas de web scraping. Puesto que las páginas web han de ser constantemente modificadas y actualizadas, sus contenidos cambian con el tiempo. Puede que cambie su diseño, por ejemplo, o que se les añadan nuevos elementos. Los webs scrapers se desarrollan teniendo en cuenta la estructura específica de una página web, de forma que, si dicha estructura cambia, el scraper también debe modificarse. Este proceso resulta especialmente sencillo con Python.

Asimismo, Python tiene como puntos fuertes el procesamiento de texto y la apertura de recursos web, dos de las bases técnicas del web scraping. Python es, además, un estándar consolidado en materia de análisis y procesamiento de datos. Por si esto fuera poco, Python ofrece un amplísimo ecosistema de programación, que cuenta con bibliotecas, proyectos de código abierto, documentación y referencias explicativas del lenguaje, así como entradas de foros, informes de error y artículos de blog.

Más concretamente, existen varias herramientas consolidadas diseñadas para hacer web scraping con Python. Las más conocidas: Scrapy, Selenium y BeautifulSoup.

3. TÉCNICAS.

Web scraping es el proceso de recopilar información de forma automática de la Web. Es un campo con desarrollos activos, compartiendo un propósito en común con la visión de la Web semántica. Utiliza soluciones prácticas basadas en tecnologías existentes que son comúnmente *ad hoc*. Existen distintos niveles de automatización que las existentes tecnologías de Web Scraping pueden brindar:

- «Copiar y pegar» humano: algunas veces incluso las mejores técnicas de *web scraping* no pueden reemplazar el examen manual de un humano, y a veces esta puede ser la única vía de solución cuando el sitio que tenemos en mente pone ciertas barreras para prevenir que se creen softwares para realizar tareas automáticas en este.
- Uso de expresiones regulares: una posible vía para extraer información de páginas webs pueden ser las expresiones regulares, aunque comúnmente no se recomienda utilizarlas para parsear el formato HTML.
- Protocolo HTTP: páginas webs estáticas y dinámicas pueden ser obtenidas haciendo peticiones HTTP al servidor remoto utilizando sockets, etc.
- Algoritmos de minería de datos: muchos sitios webs tienen grandes colecciones de páginas generadas dinámicamente a partir de una base de datos. Datos de la misma categoría aparecen usualmente en páginas similares mediante un *script* o una plantilla. En la minería de datos, un programa detecta estas plantillas en un contexto específico y extrae su contenido.
- Parsers de HTML: Algunos lenguajes, como XQuery y HTQL pueden ser utilizados para parsear documentos, recuperar y transformar el contenido de documentos HTML.
- Aplicaciones para *web scraping*: existen muchas aplicaciones disponibles que pueden ser utilizadas para personalizar soluciones de Web Scraping. Estas aplicaciones podrían reconocer automáticamente la estructura de cierta página o brindar una interfaz al usuario donde este pudiera seleccionar los campos que son de interés dentro del documento. De esta forma no es necesario escribir manualmente código para realizar estas tareas.
- Reconocimiento de información semántica: las páginas que son analizadas podrían incluir metadatos o cierta información semántica como anotaciones o comentarios, los cuales pueden ser usados comúnmente. Si estas anotaciones están en las mismas páginas, como sucede con los micro formatos, estas podrían ser de utilidad cuando parseamos el DOM del documento. En otro caso, las anotaciones, organizadas en una capa semántica, son almacenadas y manejadas de forma separada desde otras páginas, por lo que los scrapers pueden recuperar estos esquemas y las instrucciones desde esta capa antes de analizar los documentos.

4. VENTAJAS.

Con Web Scraping los procesos de encontrar y recabar información se automatizan y con ello conseguimos:

- Disminuir carga de trabajo.
- Menos costoso: una vez implementado, el coste general de extracción de datos se reduce significativamente, especialmente si se compara con el trabajo manual.
- Acceso "ilimitado". Todo lo que puedas ver lo puedes extraer.
- Aumentar la velocidad de los procesos.
- Eliminar el error humano.
- Conseguir los datos en formatos procesables.
- Manejar grandes cantidades de datos.
- Organizado: el experto en Scraping puede organizarse para raspar datos de manera regular o en momentos puntuales, por ejemplo, cuando hay nuevos datos disponibles. De esa manera, la empresa se asegura de tener siempre los datos más recientes.
- Mantenimiento básico: El Data Scraping generalmente no requiere mucho mantenimiento.

5. DESVENTAJAS.

Si bien el Web Scraping puede proporcionar a una empresa enormes beneficios, también hay algunas desventajas y suposiciones en las que se basa:

- Webs menos complejas: cuanto más compleja sea la web que se desea raspar, más difícil será el raspado. Las razones son porque configurar el raspador se vuelve más difícil, y los costes de mantenimiento pueden aumentar, porque es más probable que el experto tenga errores y problemas.
- Página de inicio estable: el Web Scraping automatizado solo tiene sentido si la página de inicio de destino no cambia su estructura con frecuencia. Cada cambio de estructura implica costes adicionales, porque el Scraping necesitará ser ajustado.
- Datos estructurados: el raspado web no funcionará si se quiere raspar datos de 1000 webs diferentes y cada web tiene una estructura completamente diferente. Será necesario que exista alguna estructura básica que difiera solo en ciertas situaciones.
- Protección baja: si los datos en la web están protegidos, el raspado también puede convertirse en un desafío y aumentar los costes.
- Los accesos que no correspondan por acciones humanas pueden provocar el bloqueo de la IP.
- Si es utilizado de mala forma, el Scraping viola los derechos de autor, se considera spam.

6. PROCESO.

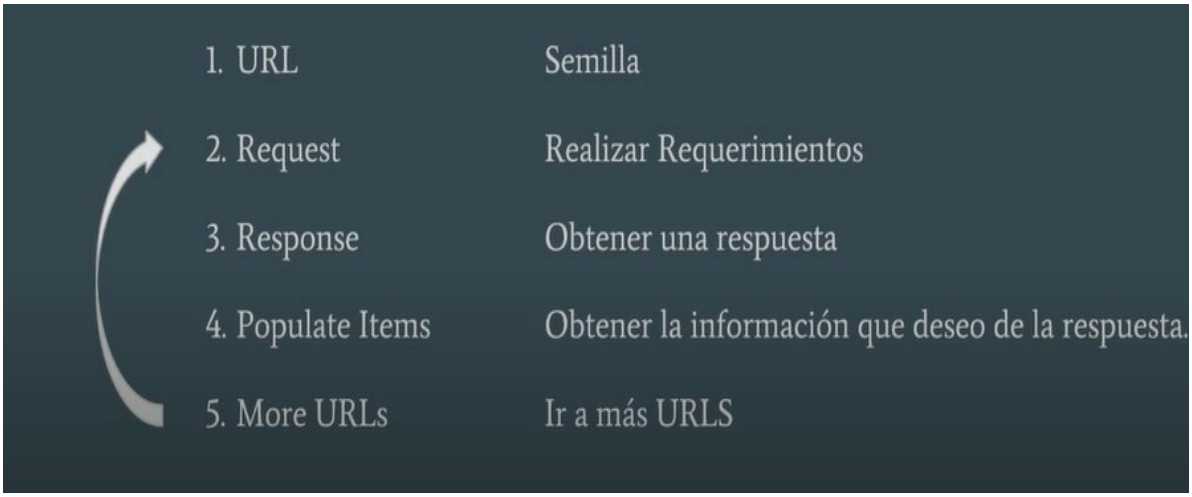
El esquema básico del web scraping es sencillo de explicar. En primer lugar, el desarrollador del scraper analiza el texto fuente en HTML de la página web en cuestión. Por lo general, encontrará patrones claros que permitirán extraer la información deseada. El scraper será entonces programado para identificar dichos patrones y realizará el resto del trabajo automáticamente:

1. Abrir la página web a través del URL.
2. Extraer automáticamente los datos estructurados a partir de los patrones.
3. Resumir, almacenar, evaluar o combinar los datos extraídos, entre otras acciones.

Casos de aplicación del web scraping:

El web scraping puede tener aplicaciones muy diversas. Además de para la indexación de buscadores, el web scraping también puede usarse con los siguientes fines, entre muchos otros:

- Crear bases de datos de contactos.
- Controlar y comparar ofertas online.
- Reunir datos de diversas fuentes online.
- Observar la evolución de la presencia y la reputación online.
- Reunir datos financieros, meteorológicos o de otro tipo.
- Observar cambios en el contenido de páginas web.
- Reunir datos con fines de investigación.
- Realizar exploraciones de datos o data mining.



7. TIPOS.

1. Una sola página web:
    - scrapy.spider.Spider

2. Varias páginas web:
    - scrapy.spider.CrawlSpider
    - Crawlin vertical.
    - Crawlin Horizontal.

8. LIBRERIAS.

La mayoría de los scrapers están escritos en python para facilitar procesamiento posterior a la recolección de datos extraídos de la web. Algunos de los scrapers se realizan usando marcos y librerías para el rastreo web:

- Request: Esta librería permite realizar peticiones a los sitios web desde python. Se suele utilizar junto con Beautiful Soup ya que ésta sólo permite buscar en el HTML las etiquetas que necesitamos.
- Beautiful Soup: Es una librería python que permite extraer información de contenido en formato HTML o XML. Esta librería suele funcionar correctamente en la mayoría de páginas, aunque suele tener problemas en páginas que contengan AJAX o Javascript.
- Scrapy: Es un framework para scraping, además de las herramientas de scraping, se puede exportar la data recopilada en varios formatos, como JSON o CSV, y almacenar los datos en un backend de su elección. También, tiene una serie de extensiones integradas para tareas como el manejo de cookies, suplantación de user-agent, restricción de la profundidad del rastreo, entre otras, así como un API para ampliar fácilmente las funcionalidades.
- Selenium: La librería Selenium nos permite automatizar navegadores web. Es una herramienta bastante potente que nos permite utilizar un navegador web como si fuese un humano y, posteriormente, se puede utilizar desde python para extraer la información.

8. CONCLUCIONES.

El Web Scraping permite obtener gran cantidad de información de una manera automática de diferentes sitios web; gracias a esto, podemos crear bases de datos aprovechables, las cuales tienen una infinidad de maneras de ser aplicadas.

Gracias al Big Data y el Scraping al ser combinados con una base de datos tiene un gran potencial, el Web Scraping puede mejorar procesos de manera muy satisfactoria, ya que todo lo que haga un humano en una página web puede programarse para que lo haga un robot.

Hoy la información es sumamente valiosa para optimizar los diferentes procesos de tu organización y para tomar decisiones acertadas.

Atentamente,

JAES ANDERSON RAMIREZ SIERRA
Tripulante MISIONTIC
UIS - S38 – G5