Universidad Tecnológica Nacional

FRRo

Seminario: Data Collection

Año 2021 - Comisión 304

Alumno	Legajo
Biscaldi Ivan	**45997**
Santolari Matias	**45790**
Romero Joan	**45862**

Docentes:
Mario Castagnino
Juan Ignacio Torres
Ezequiel Castaño

Mati-Ivan-Joan

Introducción

Data Collection es un proceso sistemático de recopilación de observaciones y mediciones. Esto permite responder preguntas relevantes y evaluar resultados. Para esto se usan técnicas estandarizadas.
El objetivo principal de Data Collection es garantizar una recopilación de datos confiables y ricos en información, para análisis y toma de decisiones.

Si bien en cada campo de investigación hay un enfoque diferente para recolectar datos, todo proceso de recolección de datos debe tener:

- Objetivo de investigación
- Tipo de información a recolectar
- Métodos y procedimientos para colectar, almacenar y procesar datos

Sin embargo, establecer ese proceso puede ser complicado. Implica hacer un balance de sus objetivos, identificar sus requisitos de datos, decidir un método de recopilación de datos y, finalmente, organizar un plan de recopilación de datos que sintetice los aspectos más importantes de su programa.

Técnicas

Existen formas tradicionales de colectar datos:

- Encuesta
- Entrevista
- Prueba
- Evaluaciones Fisiológicas
- Revisión de Registros
- Muestras Biológicas

Pero es este documento vamos a hablar de formas más modernas de recolección de datos, como son el web scraping y el uso de APIs.

Web Scraping

El web Scraping(del inglés *scraping* = arañar/raspar) es un proceso en el que se usan bots que extraerán contenido y datos de sitios web. Estos extraen código HTML y almacenan los datos, bajo la suposición de que todo contenido web se puede duplicar.

A los bots que se programan para extraer informacion se los puede personalizar para:

- Reconocer estructuras únicas de sitios HTML
- Extraer y transformar contenido
- Almacenar datos
- Extraer datos de APIs

El web scraping tiene casos de uso legítimos cómo lo son clasificación de sitios y estudio de mercado, pero también tiene usos ilegales cómo raspado de precios y robo de contenido con derechos de autor.

Bots legítimos y maliciosos

Los bots legítimos se identifican con una organización/empresa mientras que los maliciosos se hacen pasar por usuarios HTTP. Además un bot legítimo respeta los archivos robot.txt pero los maliciosos rastrean los sitios sin importar que está permitido o no.

Un operador de bots legítimos tendrá servidores para procesar la información y un operador de bots maliciosos usa una red de bots dispersos en computadoras infectadas por malware, que son controladas desde una ubicación central.

Protección contra el web scraping

- Prevenir ataques: se pueden identificar posibles direcciones IP y filtrarlas a través del firewall para evitar que realicen solicitudes

- Usar tokens de falsificación de solicitud(CSRF): el uso de tokens CSRF evita que invitados hagan solicitudes arbitrarias a las URL

- Usar archivo .htaccess para evitar raspado: este es un archivo de configuración de servidor que se puede modificar para evitar el acceso a datos de ciertos usuarios.

- Prevenir hotlinking: Los atacantes suelen copiar todo el contenido de un sitio y alojarlo en un sitio diferente. El proceso de mostrar un recurso en un servidor que está alojado en otro servidor se conoce cómo hotlinking. Además cada vez que alguien visite el sitio web del atacante consumirá datos del sitio original. Para evitar esto se alojan en los servidores archivos .htaccess a los cuales se les puede especificar qué recursos enviar o no cada vez que se los solicite, en base al URL que lo está solicitando.

- Listas negras de direcciones IP: a través de .htacces se pueden bloquear direcciones IP de raspado identificadas y también patrones de direcciones IP.

- Limitar solicitudes de una dirección IP: también se puede limitar la cantidad de solicitudes de una dirección IP, siendo esto no tan efectivo ya que por lo general un atacante usa más de una dirección. Ante solicitudes anormales sería más efectivo usar captcha.

- Crear “honeypots”: los honeypots son enlaces a contenido falso contenido en código html pero que un usuario no puede ver. Estos pueden detectar raspadores y enviarlos a enlaces vacíos haciéndolos desperdiciar recursos.

- Cambiar la estructura HTML: cómo la mayoría de los raspadores usan el codigo HTML para acceder a datos, es conveniente modificar la estructura de este con frecuencia, lo cual obligará al atacante a tener que analizar nuevamente el código y dificultará el acceso a datos

- Proporcionar APIs: también podría permitirse de forma selectiva y bajo ciertas reglas el acceso a datos. Esto se hace con APIs basadas en suscripción, ya que estas permiten monitorear y restringir el uso de servicios ofrecidos.

- Usar archivo Robots.txt: Este archivo indica a los rastreadores a que URLs pueden acceder. Se usa para evitar que un sitio se sobrecargue de solicitudes y limitar las páginas que puede indexar un motor de búsqueda. EL archivo debe colocarse en la raíz de un sitio(por ejemplo para un sitio www.ejemplo.com se lo ublica en www.ejemplo.com/robots.txt)

Aplicaciones

El web scraping se utiliza para una gran variedad de tareas, por ejemplo, para recopilar datos de contacto o información especial con gran rapidez. En el ámbito profesional, el scraping se utiliza a menudo para obtener ventajas respecto a la competencia. De esta forma, por medio de la colecta de datos, una empresa puede examinar todos los productos de un competidor y compararlos con los propios. El web scraping también resulta valioso en relación con los datos financieros: es posible leer datos desde un sitio web externo, organizarlos en forma de tabla y después analizarlos y procesarlos.
Google es un buen ejemplo de web scraping. El buscador utiliza esta tecnología para mostrar información meteorológica o comparaciones de precios de hoteles y vuelos.
Muchos de los portales actuales de comparación de precios también utilizan el scraping para representar información de diferentes sitios web y proveedores.
Los dos casos de uso más comunes son el raspado de precios y el robo de contenido.

➢ Raspado de precios

El raspado de precios es una tecnica generalmente maliciosa de web scraping en la que generalmente un atacante utiliza una red de bots(arañas) para inspeccionar las bases de datos de la competencia. El objetivo es acceder a la información de precios, ganar a los rivales e impulsar las ventas. Para los atacantes, un raspado de precios exitoso puede hacer que sus ofertas sean destacadas en sitios web de comparación.

Los ataques ocurren con frecuencia en industrias donde el precio de los productos son fácilmente comparables. Porque el precio juega un papel importante en las decisiones de compra. Las víctimas del raspado de precios pueden ser agencias de viajes, vendedores de electrónica en línea, etc.

Ya que los clientes siempre suelen optar por la oferta más económica. Para obtener una ventaja, un proveedor puede usar un bot para raspar continuamente los sitios web de sus competidores y actualizar casi instantáneamente sus propios precios en consecuencia.

➢ Raspado de contenido

El raspado de contenido implica el robo de contenido a gran escala de un sitio determinado. Los objetivos típicos incluyen catálogos de productos en línea y sitios web que se basan en contenido digital para impulsar el negocio. Para estas empresas, un ataque de raspado de contenido puede ser devastador.

Por ejemplo, los directorios de negocios en línea invierten cantidades significativas de tiempo, dinero y energía en la construcción de su base de datos. El raspado puede hacer que todo se vaya al traste. Se usa en campañas de envío de correo no deseado. O se revende a los competidores. Es probable que cualquiera de estos hechos afecte a los resultados de una empresa y a sus operaciones diarias.

Existen muchas más tecnicas:

- Búsqueda random de información.

- Recopilación de datos para análisis de Big Data, Machine Learning e Inteligencia Artificial.
- Rastreo de mercados bursátiles y financieros.
- Obtención de datos para estudios de investigación y periodísticos.
- Rastreo de ofertas laborales.Rastreo de mercados de segunda mano.
- Extracción de información de publicaciones en pdf como, por ejemplo de boletines oficiales.

Aplicaciones para el marketing digital

En cualquier departamento de marketing, la mayoría de acciones son para ayer. Contar con herramientas que te ayuden a gestionar mejor los deadlines es de agradecer. Enseña a tu equipo qué es el web scraping y sus utilidades:

- **Crear una base de datos con lo que quieras: emails, teléfonos, direcciones, empresas, datos estadísticos, etc**
- **Obtener datos de la competencia**, para hacer benchmarking.
- Monitorizar a la competencia, controlar, rastrear y generar alertas para saber cuándo los competidores actualizan sus catálogos de producto o servicio, renuevan su sitio web, escriben sobre un tema concreto, mencionan nuestros productos…
- **Conocer la reputación online de la competencia.**
- **Conocer tu propia reputación online** y controlar la presencia del nombre de tu marca en determinados foros.
- **Caza de tendencias**: De qué marcas, productos o personas se va a hablar durante los próximos meses.
- Optimizar precios de tiendas online, a través del análisis histórico de la competencia.
- Conocer los resultados de búsqueda en Google de diversas palabras clave. Identificar las posiciones en dichos resultados, tipo de contenidos, y mucho más.
- Marketing de contenidos. Obtener datos concretos de webs para generar tu propio contenido. O conseguir contenido relevante en otros idiomas que al traducirlo se convierte en contenido original.
- **Ganar visibilidad en redes sociales: Puedes utilizar los datos para interactuar a través de un robot con usuarios en redes sociales**
- Generar datos de las etiquetas de imágenes y de sitios web para crear modelos de clasificación de imágenes.
- Extraer comentarios de usuarios y de sitios de comercio electrónico como Amazon.
- **Detectar influencers: Sería una información muy útil para planificar tu campaña de marketing digital.**
- Optimización ecommerce. Elegir qué imagen mostrar como destacada, qué categorización de productos funciona mejor, qué nicho está libre en un mercado concreto, etc.
- Optimización del Copy: Podrás saber qué estructuras gramaticales llaman la atención de los lectores, analizando, por ejemplo, los títulos de los vídeos con más visualizaciones.
- Eventos: extraer información sobre los eventos de un determinado vertical en un área geográfica y crear una lista. Compilar información de viviendas inmobiliarias.

Beautiful Soup

Es una librería de Python que permite extraer datos desde archivos HTML y XML. Beautiful Soup crea árboles con los elementos del documento y permite extraer información a partir de estos.
Se puede utilizar con diferentes analizadores incluido el analizador de HTML que viene con Python.

Parser/analizador	Llamada/Definición	Ventajas	Desventajas
html.parser	`BeautifulSoup(markup, "html.parser")`	• Incluido en Python • Velocidad decente • Tolerante a los errores*(indulgente)	• No tan rápido cómo lxml y menos indulgente que html5lib
lxml	`BeautifulSoup(markup, "lxml")`	• Muy rápido • Indulgente	• Dependencias de C externas
lxml-xml	`BeautifulSoup(markup, "lxml-xml")`	• Muy rápido • Unico parser de xml soportado	• Dependencias de C externas
html5lib	`BeautifulSoup(markup, "html5lib")`	• Extremadamente indulgente • Analiza las páginas de la misma forma que un buscador lo hace • Crea HTML5 valido	• Muy lento • Dependencias externas de Python

El paquete de Beautiful Soup funciona con Python 2 y Python 3.
Con linux/debian se puede instalar usando pip o easy-install y también con el comando apt-get desde la terminal

Autor(y mantenimiento): Leonard Richardson(experto en diseño de restful Api)
Última Versión: Beautifulsoup4 4.9.3- fue lanzada el 3 de Octubre de 2020
Está version se puede usar con cualquier version de Python 3, y versiones de Python 2 superiores a 2.7

Beautifulsoup4 a diferencia de la versión anterior, Beautifulsoup3, soporta Python3.

Debido a que Beautifulsoup está desarrollado en Python 2, hay muchas características de Python 3 que no pueden aprovecharse. Por esto a partir del 31/12/2020 se desarrollará exclusivamente en Python 3. Sin embargo las versiones anteriores seguiran estando disponibles.
La única dependencia de Beautifulsoup4 es soupsieve.

Scrapy

Scrapy es un framework de rastreo web y raspado web rápido de alto nivel, que se utiliza para rastrear sitios web y extraer datos estructurados de sus páginas. Se puede utilizar para una amplia gama de propósitos, desde minería de datos hasta monitoreo y pruebas automatizadas.
Scrapy está escrito en Python puro y depende de algunos paquetes clave de Python(entre otros):

- lxml , un analizador XML y HTML eficiente
- parsel , una libreria de extracción de datos HTML / XML escrita sobre lxml,
- w3lib , un ayudante multipropósito para tratar con URL y codificaciones de páginas web
- twisted , un framework de red asincrónico
  Es una herramienta que permite crear un servidor internet y no solamente un servidor web. En otras palabras, no solo administra el protocolo HTTP, sino que puede hacer WebDav, FTP o implementar sus propios protocolos.
- cryptography and pyOpenSSL, para hacer frente a diversas necesidades de seguridad a nivel de red.

Las versiones mínimas con las que se prueba Scrapy son:

- Torcido 14.0
- lxml 3.4
- pyOpenSSL 0,14

Scrapy puede funcionar con versiones anteriores de estos paquetes, pero no se garantiza que seguirá funcionando porque no se está probando con ellos.
Algunos de estos paquetes dependen de paquetes que no son de Python que pueden requerir pasos de instalación adicionales según su plataforma.
Scrapy es mantenido por Zyte (anteriormente Scrapinghub) y muchos otros colaboradores .

➢ Requisitos

- Python 3.6+
- Funciona en Linux, Windows, macOS, BSD

➢ Instalación de Scrapy

Si está usando Anaconda o Miniconda , puede instalar el paquete desde el canal conda-forge , que tiene paquetes actualizados para Linux, Windows y macOS.

Para instalar Scrapy usando conda, ejecute:
conda install -c conda-forge scrapy

Alternativamente, si ya está familiarizado con la instalación de paquetes de Python, puede instalar Scrapy y sus dependencias desde PyPI con: **pip install Scrapy**

Recomendamos encarecidamente que instale Scrapy en un virtualenv dedicado , para evitar conflictos con los paquetes de su sistema.

➢ Una manera de iniciar con Scrapy es escribir en la terminal:

```
scrapy startproject [nombre_proy] #crea proyecto sin araña
scrapy startproject [nombre_proy] [url]  #crea proyecto con araña
```

De esta manera se genera un proyecto de scrapy donde vas a poder trabajar de manera mucho más ordenada con una sección para alojar las arañas, otra los items, otra sección de pipelines, etc.

- Spider(araña): Son clases que definen cómo se raspará un sitio/grupo de sitios, incluido cómo realizar el rastreo (es decir, seguir enlaces) y cómo extraer datos estructurados de sus páginas (es decir, raspar elementos).
- Item: Las arañas pueden devolver los datos extraídos como items , objetos de Python que definen pares clave-valor
  (items son el lugar donde las arañas devuelven los datos extraídos).
- Item Pipeline: Es una clase de Python que implementa un método simple. Reciben un Item y realizan una acción sobre él, y también deciden si el Item debe continuar o debe descartarse y dejar de procesarse.
  Los usos típicos de los items pipelines son:
  - limpieza de datos HTML
  - validando datos raspados (verificando que los elementos contienen ciertos campos)
  - buscando duplicados (y soltándolos)
  - almacenar el elemento raspado en una base de datos
- El middleware: framework de ganchos en el mecanismo de procesamiento de araña de Scrapy donde puede conectar una funcionalidad personalizada para procesar las respuestas que se envían a las arañas para su procesamiento y para procesar las solicitudes y elementos que se generan a partir de arañas.
- Scrapy utiliza objetos de solicitud(request) y respuesta(response) para rastrear sitios web. Por lo general, los objetos de solicitud se generan en las arañas y atraviesan el sistema hasta que llegan al descargador, que ejecuta la solicitud y devuelve un objeto de respuesta que regresa a la araña que emitió la solicitud.

➢ Respecto a las Versiones de Scrapy

La última versión estable es la 2.5.0.
y algunas de sus novedades respecto de la versión anterior 2.4.1 son:

Nuevas características

- Soporte experimental de HTTP / 2 a través de un nuevo controlador de descarga que se puede asignar al httpsprotocolo en la configuración DOWNLOAD_HANDLERS.
- **La nueva función** scrapy.downloadermiddlewares.retry.get_retry_request() se puede usar desde devoluciones de llamada de araña o middlewares para manejar el reintento de una solicitud más allá de los escenarios que admite RetryMiddleware.

- **La nueva señal** headers_received brinda acceso temprano a los encabezados de respuesta y permite detener las descargas .
- El nuevo atributo Response.protocol da acceso a la cadena que identifica el protocolo utilizado para descargar una respuesta.
- **Las estadísticas** ahora incluyen las siguientes entradas que indican la cantidad de éxitos y fracasos en el almacenamiento de feeds
  feedexport/success_count/<storage type>
  feedexport/failed_count/<storage type>
- **El middleware** de araña UrlLengthMiddleware ahora registra las URL ignoradas con el nivel de registro INFO en lugar de DEBUG, y ahora incluye la siguiente entrada en las estadísticas para realizar un seguimiento del número de URL ignoradas:
  urllength/request_ignored_count
- El middleware HttpCompressionMiddleware del descargador ahora registra el número de respuestas descomprimidas y el recuento total de bytes resultantes:
  httpcompression/response_bytes
  httpcompression/response_count

Corrección de errores

- Se corrigió la instalación en PyPy instalando PyDispatcher además de PyPyDispatcher, lo que podría evitar que Scrapy funcione según el paquete que se haya importado.

- Al inspeccionar una devolución de llamada para comprobar si es un generador que también devuelve un valor, ya no se genera una excepción si la devolución de llamada tiene una cadena de documentación con una sangría más baja que el código siguiente.

- El encabezado Content-Length ya no se omite en las respuestas cuando se usa el controlador de descarga HTTP / 1.1 predeterminado (ver DOWNLOAD_HANDLERS).

- Establecer la meta clave de solicitud handle_httpstatus_all en False ahora tiene el mismo efecto que no establecerla en absoluto, en lugar de tener el mismo efecto que establecerla en True.

Diferencia entre Scrapy y Beautiful Soup

Scrapy y BeautifulSoup utilizan los dos la misma técnica, un analizador sintáctico o *parser* HTML para extraer datos del texto fuente (en HTML) de la web siguiendo este esquema:

URL → Solicitud HTTP → HTML → Scrapy/BeautifulSoup

El concepto clave del Scrapy son los llamados *spiders* (*arañas*), están programados para *scrapear* una web concreta y va *desplegándose* de página a página. La programación usada está orientada a objetos: cada *spider* es una clase de Python propia.

Su arquitectura está orientada a las necesidades de proyectos profesionales. La apertura de las páginas en Scrapy se produce de forma asíncrona, es decir, con la posibilidad de descargar varias páginas simultáneamente. Por ello, Scrapy es una buena opción para proyectos de *scraping* que hayan de procesar grandes volúmenes de páginas.

BeautifulSoup es la más antigua y al igual que en el caso de Scrapy, utilizan la misma técnica. Sin embargo, a diferencia de Scrapy, en BeautifulSoup el desarrollo del *scraper* no requiere una programación orientada a objetos, sino que el *scraper* se redacta como una sencilla secuencia de comandos o *script*. Con ello, BeautifulSoup ofrece el método más fácil para *pescar* información de la *sopa de tags* a la que hace honor su nombre. Solo recupera el contenido de la URL que usted da y luego se detiene. No se arrastra a menos que lo coloques manualmente dentro de un bucle infinito con ciertos criterios.

Beautiful Soup es una biblioteca, mientras que Scrapy es un marco completo.

Raspado con APIs

A pesar de su efectividad, el *web scraping* no es el mejor método para obtener datos de páginas web. De hecho, a menudo existe una alternativa mejor: muchos operadores web publican los datos de forma estructurada y en un formato legible para las máquinas. Para acceder a este tipo de datos se usan interfaces de programación especiales llamadas *Application Programming Interfaces* (interfaces de programación de aplicaciones, API por sus siglas en inglés).

El uso de una API ofrece importantes ventajas:
El propietario de la web crea la API precisamente para permitir el acceso a los datos. De esta forma se reduce el riesgo de infracciones y el operador web puede regular mejor el acceso a los datos. Una manera de hacerlo es, por ejemplo, solicitando una clave API para acceder a ellos. Este método también permite al operador regular de forma más precisa las limitaciones del rendimiento.
La API presenta los datos directamente en un formato legible para las máquinas. Con ello, ya no es necesaria la laboriosa tarea de extraer los datos del texto fuente. Además, la estructura de los datos se separa de su representación visual, por lo que se mantiene sin importar si cambia el diseño de la web.

Hay varias opciones dentro del mundo python, alguna de ellas son:

- Yahoo Finance
- PyOWM
- APIs de Redes Sociales
- Telegram
- Mercado Libre

APIs

Estas son un conjunto de definiciones y protocolos para facilitar el desarrollo e integración de aplicaciones.
En las Apis, una parte envía una solicitud remota con cierta escritura, y esta determina la respuesta del software del otro lado.

Las Apis permiten la comunicación de productos y servicios sin necesidad de conocer cómo están implementados. Flexibilizan el desarrollo. Simplifican diseño y administración y uso de aplicaciones

Una Api debe admitir rápida implementación y desarrollo de servicios para garantizar competitividad, ya que los mercados digitales cambian constantemente.

Una Api hace posible hacer cambios en los sistemas internos sin necesidad de afectar a los clientes, siempre y cuando no se modifique el comportamiento de está.

Las Apis no solo son un medio simplificado para conectar infraestructura, una Api pública permite compartir datos con clientes y usuarios externos, lo que da la posibilidad de hacer rentables los datos.

Una API permite habilitar el acceso a recursos, y al mismo tiempo mantener seguridad y control.

Existen 3 enfoques respecto a las política de Apis:

- Apis privadas: solo se pueden usar internamente, lo que da control a la empresa sobre el uso de la Api.
- Apis de partners: la Api es compartida con empresas específicas, lo cual puede generar flujos de ingreso adicionales pero no afecta la calidad.
- Apis públicas: la Api es de libre acceso, por lo que cualquiera podría hasta desarrollar una aplicacion que se comunica con está. De está forma la Api puede convertirse en un recurso de innovación.

Las Apis son previas a las computadoras personales. Funcionaban cómo bibliotecas para los sistemas operativos. Estaban habilitadas localmente en los sistemas que la operaban y a veces pasaban mensajes entre las computadoras centrales.

Luego surgieron las Apis remotas, que estaban diseñadas para interactuar a través de una red de comunicaciones. Y cómo la red de comunicaciones más utilizada es internet, la mayoría de las Apis están diseñadas de acuerdo a estándares web.

3 tipos básicos de Api-(en presentacion)

Api Rest y Api Soap

Con la difusión de las Apis se estandarizó una especificación de protocolo para intercambio de información, llamado 'Protocolo de Acceso a Objetos Simples'. Una API diseñada con SOAP usa XML para sus mensajes, y recibe solicitudes HTTP o SMTP. Este protocolo facilita compartir información entre aplicaciones escritas en diferentes lenguajes o de diferentes entornos.

Otra especificación se la conoce cómo Transferencia de Estado Representacional (REST). Esto no es un protocolo sino un estilo de arquitectura. No hay un estandard para Apis restful. Las Api restful solo deben cumplir con las 6 limitaciones de un sistema restful:

- Arquitectura cliente-servidor: arquitectura compuesta por clientes, servidores y recursos, y administra solicitudes con HTTP
- Sin estado: la información sobre el estado de la sesión no se almacena en el servidor, queda en posesión del cliente
- Capacidad de caché: el almacenamiento caché eliminara algunas interacciones cliente-servidor
- Sistema en capas: la interacción cliente servidor puede tener capas adicionales que ofrecen funciones cómo equilibrio de carga, cachés compartidas o seguridad.
- Interfaz uniforme: limitación fundamental, tiene 4 características.
    - Identificación de recursos en las solicitudes: los recursos se identifican en las solicitudes y se separan de las representaciones que se devuelven al cliente
    - Administración de los recursos mediante representaciones: el cliente recibe archivos que son representaciones y deben contener la suficiente info cómo para ser modificadas o eliminadas
    - Mensajes autodescriptivos: cada mensaje devuelto al cliente debe contener la info necesaria para describir cómo procesarla
    - Hipermedios cómo motor de estados de la aplicación: luego de acceder a un recurso, el cliente debe ser capaz de descubrir mediante hipervínculos todas las acciones disponibles
- Código de demanda(opcional): los servidores pueden extender las funciones de un cliente transfiriendo código ejecutable.

Las Apis que funcionan con está limitación de arquitectura se las conoce cómo Api restful. Aunque parece tener muchas especificaciones es mucho más sencilla que una Api SOAP.

SOA y arquitectura de microservicios

Existen 2 enfoques de arquitectura para Apis remotas:

Uno es la arquitectura orientada a servicios(SOA), que implica el uso de varias aplicaciones que ofrecen diferentes funciones y que no tienen conexión directa. Está arquitectura es más sencilla que una monolítica pero tiene un riesgo de cambios en cascada.

Otro enfoque es la arquitectura de microservicios, que es parecida a la SOA respecto a servicios especializados y conexión no directa, pero además las arquitecturas tradicionales se descomponen en partes más pequeñas y sus servicios usan un marco de mensajería común cómo Apis restful, evitando una necesidad de operaciones de conversión de datos compleja. Además el uso de restful apis fomenta una distribución más rápida de funciones nuevas y actualizaciones. Cada servicio es independiente y se puede reemplazar modificar o abandonar.

Formatos

- **CSV:** Los archivos CSV (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal como en Chile, Perú, Argentina, España, Brasil, entre otros) y las filas por saltos de línea.
- **JSON:** JavaScript Object Notation (JSON) es un formato basado en texto estándar para representar datos estructurados en la sintaxis de objetos de JavaScript. Es comúnmente utilizado para transmitir datos en aplicaciones web (por ejemplo: enviar algunos datos desde el servidor al cliente, así estos datos pueden ser mostrados en páginas web, o vice versa)
- **Binario:** se denomina archivos binarios a todos aquellos que deben ser interpretados por un programa o procesador de hardware que entiende cómo están formateados. Es decir, el archivo no tiene un formato identificable externamente por lo que el programa que lo lea debe saber cómo se distribuyen los datos. Un archivo binario puede tratarse tanto de un programa ejecutable cómo de datos a utilizar en un programa. Ejemplo de archivo binario pueden ser los .bin de un programa ejecutable, los .dat , los .264 que tienen codec de video de alta compresión(se usa en cámaras), los .uhs que son similares a archivos zip y existen muchas otras extensiones...
    - **Modulo Pickle:** este módulo nos permite almacenar fácilmente colecciones y objetos en ficheros binarios abstrayendo todo la parte de escritura y lectura binaria.
- **HDF:** El formato de datos jerárquico es un **conjunto de formatos de archivo** (HDF4 , HDF5) diseñado para **almacenar y organizar grandes cantidades de datos**.
- **Feather:** Frecuentemente necesitamos usar en R un data frame generado en Python por Pandas, o al revés, tengo datos en un data frame de R que quiero usar en Python.
  Una manera común para compartir data frames entre Python y R es exportarlas como archivos CSV. Este método tiene el inconveniente de que perdemos los metadatos de nuestro data frame.
  Si tenemos datos de fecha con husos horarios, factores de R, texto con codificación específica o con cualquier otro metadato, este se pierde al exportar un data frame a CSV. Es posible que los datos que creímos exportar en Python no sean los mismos que importamos en R.
  Para compartir data frames entre Python y R de manera que no perdamos información, podemos usar el formato de archivo Feather. Si guardamos un data frame como un archivo feather, será legible por Python y por R, sin pérdida de contenido. Para usar este formato, tenemos que instalar el paquete Feather en Python y R.

- **Apache:**
    - **ORC:** es un formato que almacena colecciones de filas en un archivo y dentro de la colección en la que están almacenados los datos de fila en un formato de columnas. Esto permite el proceso paralelo de colecciones de filas en un clúster.

      La atención se centró en permitir el procesamiento de alta velocidad y reducir el tamaño de los archivos.
      ORC es un formato de archivo en columnas autodescriptivo. Está optimizado para grandes lecturas de transmisión, pero con soporte integrado para encontrar rápidamente las filas requeridas. El almacenamiento de datos en formato de columnas **permite al lector leer**, **descomprimir y procesar** solo **los valores necesarios para la consulta actual**. Debido a que los archivos ORC reconocen los tipos, el escritor elige la codificación más adecuada para el tipo y crea un índice interno a medida que se escribe el archivo.
      Facebook usa ORC para ahorrar decenas de petabytes en su almacén de datos y demostró que **ORC es significativamente más rápido que** RC File o **Parquet**. Yahoo utiliza ORC para almacenar sus datos de producción y ha publicado algunos de sus resultados de referencia .

    - **Parquet**: Es un formato de almacenamiento en columnas que puede procesarse en un entorno de Hadoop y utiliza un algoritmo de destrucción y ensamblado de registros.
    - **Avro**: es un sistema de serialización de datos en formato binario o en otros formatos de datos; los datos de Avro están en un formato que puede no ser directamente legible para el usuario.

Git scraping

Es una forma extremadamente eficaz de convertir todo tipo de fuentes de datos interesantes en un registro de cambios a lo largo del tiempo.
En síntesis es una técnica de scraping que tiene como objetivo recopilar conjuntos de datos que cambian a medida que pasa el tiempo(modificando datos que ya pertenecen a dicho conjunto de datos, agregando nuevos o eliminandolos).

En algunos proyectos hay manipulación y transformación constante de datos llegando a ser necesario tener un control de los cambios de estos a lo largo del proyecto. La idea es lograr algo similar a lo que hace el sistema de control de versiones de git con los archivos: registrar los cambios, quién los hizo y cuándo. Esto sería muy útil para casos cómo los de machine learning ya que tener las diferentes versiones del conjunto de datos permitiria experimentar con los modelos, ver que modelo funcionaba con determinado conjunto de datos, reproducir experimentos anteriores y hasta restaurar versiones anteriores.
De esta manera podemos raspar un repositorio cada cierto tiempo y de esa forma poder visualizar qué cambios ha tenido dicho repositorio a medida que transcurría el tiempo y los colaboradores de dicho proyecto agregaron versiones nuevas.

Algunos ejemplos que podríamos mencionar donde se haya aplicado git scraping son:

- Registros de cambios para ayudar a comprender los incendios en North Bay
  En este caso Simon Willison lo que hizo fue (en 2017) recopilar y rastrear algunos datos que podrían ser útiles para cualquiera que intente comprender lo que está sucediendo allí. Con la esperanza de que puedan resultar útiles.
  Simon raspo una serie de fuentes relevantes para la crisis y colocándolos en un repositorio en github. ya que como es un repositorio en git, los cambios en esas fuentes se rastrean automáticamente.
  El valor aquí yace en el historial de los datos.

  Las fuentes se están rastreando en este caso son:

  1. La información de emergencia del cuerpo de bomberos de Santa Rosa.
     Tiene detalles clave como la ubicación y disponibilidad de los refugios y es útil saber qué se cambió y cuándo.
     Historial de cambios en esa página .
  2. Cortes de energía de PG&E .
     El historial de confirmación de estos muestra exactamente cuándo se informan nuevas interrupciones y cuántos clientes se vieron afectados.
  3. Condiciones de las carreteras en el condado de Sonoma . Si desea comprender qué tan lejos se ha extendido el fuego, esta es una fuente de datos útil, ya que muestra qué carreteras se han cerrado debido a un incendio u otras razones.
     Historia de cambios .
  4. Incidentes de la Patrulla de Caminos de California.
     Dado que estos cubren todo el estado de California, hay muchas cosas aquí que no son directamente relevantes para North Bay, pero los incidentes que mencionan incendios aún ayudan a contar la historia de lo que ha estado sucediendo. Historia de cambios .

  El código de los raspadores se puede encontrar en north_bay.py .

- Generación de un registro de confirmación para la lista oficial de árboles de San Francisco
  San Francisco tiene un portal de datos abiertos ordenado. Entre ellos, una lista de todos los árboles de la ciudad mantenida por el Departamento de Obras Públicas.

  El archivo se actualiza regularmente.

  Aquí se aplicó git scraping y el resultado fue sf-tree-history , un repositorio de git dedicado a registrar el historial de cambios realizados en la lista oficial de árboles de San Francisco.

Data Version Control - DVC

DVC es cómo un Git pero para datos. Es una herramienta gratuita y de código abierto de gran utilidad para ciencia de datos y Machine Learning. DVC tiene versionado de datos para archivos grandes, versionado de modelos Machine Learning, de conjuntos de datos, flujo de trabajo, ademas facilita la colaboración y la reproducibilidad. DVC hace que los proyectos sean reproducibles y compartibles, los que nos posibilita saber cómo se construyeron y probaron los modelos y cómo han sido transformados los datos originales. DVC está basado en [Git] aunque puede funcionar de manera autónoma (pero sin capacidad de versionado).

FastDS es una herramienta adicional que complementa git y dvc. Cómo git y dvc trabajan de forma independiente los pasos a realizar se terminan duplicando y hasta pueden ocurrir errores humanos. FastDs intenta minimizar esto, sus comandos commit/ push/ add y demas aplican tanto para los archivos que se almacenan en git cómo para los que usan DVC.

Ofertas Laborales

Las ofertas de trabajo de web scraping abarcan campos como Analizador y Desarrollador de Web Scraping/Datos, Data Science, Data Engineer y más. Dentro del mercado laboral argentino hay pocas ofertas, pero muchas oportunidades de trabajo remoto de empresas del exterior. Estas ofertas de este tipo de trabajo se caracterizan por tener una gran remuneración.

Links útiles:

Web Scraping:

Resumen general:
https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/web-scraping-con-python/
Lista de reproducción:
https://www.youtube.com/playlist?list=PLuaGRMrO-j-8B_RT_2kGE6NW-ZxzHaU17

Beautifulsoup:

Guia simple de Beautifulsoup con lxml:
https://youtu.be/ng2o98k983k

Scrapy:

Guia para usar Scrapy(lista de reproducción)
https://youtube.com/playlist?list=PLRzwgpycm-Fjvdf7RpmxnPMyJ80RecJjv

DVC:

Tutorial con dvc:
https://youtu.be/kLKBcPonMYw

Fuentes:

- https://www.scribbr.com/methodology/data-collection/
- https://www.questionpro.com/blog/data-collection/
- https://www.antevenio.com/blog/2019/03/que-es-el-web-scraping-y-para-que-sirve/
- https://en.wikipedia.org/wiki/Data_collection
- https://www.dimagi.com/data-collection/
- https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/que-es-el-web-scraping/
- http://www.runsoftware.net/utilidades/que-es-el-hotlinking-y-como-evitarlo
- https://www.octoparse.es/blog/web-scraping-api-para-extraccion-de-datos
- https://whatis.techtarget.com/definition/binary-file
- https://beautiful-soup-4.readthedocs.io/en/latest/
- https://pypi.org/project/beautifulsoup4/
- https://libraries.io/pypi/beautifulsoup4
- https://www.quora.com/What-are-some-interesting-web-scraping-projects-you-have-done
- https://coobis.com/es/cooblog/que-es-el-web-scraping/
- https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces
- https://evaluandocloud.com/la-economia-las-interfases-api/
- https://opensciencelabs.org/2021/03/22/que-es-el-data-version-control-y-por-que-es-necesario-que-tu-equipo-sepa-como-utilizarlo/
- https://dvc.org/
- https://github.com/DAGsHub/fds