ISSN 1870-4069

Uso de técnicas de Web Scraping para obtención automática de bases de datos en la Web

Rogelio Mijangos-Espinosa, Alicia Martínez-Rebollar,
Hugo Estrada-Esquivel, Yasmín Hernández-Pérez

[1]Tecnológico Nacional de México
Departamento de Ciencias Computacionales,
México

{m20ce066, alicia.mr, hugo.ee, yasmin.hp}@cenidet.tecnm.mx

Resumen. Actualmente, la Web es el canal de comunicación más utilizado a nivel mundial. Millones de usuarios utilizan esta red para transmitir sus datos a diferentes personas, empresas, dependencias gubernamentales, instituciones educativas y de investigación. La Web permite la distribución de información de manera continua e ilimitada, lo cual implica que ésta requiera ser localizada, recolectada, revisada para poder ser utilizada con fines de investigación. Sin embargo, este enorme flujo de datos que se produce cada día en la Web hace muy lenta y complicada la captura, oportuna y sistemática, de la información que se produce en forma periódica, así como la identificación de nueva información colocada en la Web. En este sentido se requiere de sistemas a los cuales se pueda delegar las tareas de búsqueda y recolección de información. En este artículo se presenta una herramienta software que implementa técnicas de *Web Scraping* para la búsqueda, selección y descarga de bases de datos alojadas en la web, creando un compendio de datos que puede ser utilizado por técnicas de análisis de información. El sistema propuesto se ha evaluado con los procesos de búsqueda, descarga y almacenamiento de información sobre el COVID-19 en México.

Palabras clave: Web scraping, extracción de información, big data, COVID-19.

Use of Web Scraping Techniques for Automatic Capturing of Databases Located in the Web

Abstract. Currently, the Web is the most widely used communication channel worldwide. Millions of people use this network to transmit their data to different people, companies, government agencies, educational and, and also research institutions. The Web enables the unlimited and continuous distribution of

information, which implies that it needs to be located, collected, reviewed in order to be used for research purposes. However, this enormous flow of data that is produced every day on the Web makes it very difficult the timely and systematical capturing of information that is produced periodically, as well as the identification of new information published on the Web. In this context, software systems are needed to which the tasks of searching and collecting information can be delegated. This paper presents a software tool that implements Web Scraping techniques for searching, selecting and downloading databases located on the Web, which enables to create a data repositorium that can be used by information analysis techniques. The proposed system has been evaluated with the processes of searching, downloading and storing of information about the COVID-19 in Mexico.

Keywords. Web scraping, information extraction, big data, COVID-19.

1. Introducción

En la actualidad la Web representa la fuente de información más utilizada en el ámbito académico, científico e industrial. La cantidad de información contenida en la Web es impresionante, y comprende desde documentación formal y estructurada de muy diversos temas, hasta contenidos informales generados directamente por usuarios a través de medios sociales. De esta forma la cantidad de información almacenada en la Web crece día con día a medida que más usuarios cuentan con mecanismos de acceso a internet.

De esta manera, la Web se ha convertido en la principal fuente de información para las personas que realizan investigaciones que requieren información actualizada. La recolección de información de forma oportuna, correcta y completa requiere de algoritmos inteligentes que ayuden en el proceso automático de captura y almacenamiento de información.

Uno de los ejemplos más notables de la necesidad de contar con mecanismos de ayuda para el proceso de captura de información es la actual pandemia de COVID-19, la cual da origen a una enorme cantidad de información difundida y actualizada contantemente por dependencias de gobierno, instituciones públicas e instituciones de salud de todo el mundo [1]. La recuperación de información en la Web no es algo sencillo, especialmente cuando se tiene información con características Big data, es decir, con grandes cúmulos de información de muy diferentes fuentes de datos.

Las cantidades de datos masivos que se generan en la Web, el constante crecimiento y la actualización (muchas veces en tiempo real) vuelve complicada la tarea de obtener buena información estructurada en los tiempos para realizar análisis con oportunidad. Esta situación obliga la revisión constante de los contenidos de sitios Web para mantener la información actualizada.

Sin embargo, la revisión de múltiples sitios Web con técnicas convencionales merma la cantidad de información que se puede recuperar manualmente [2]. Este artículo presenta un sistema de recuperación de información que automatiza la recolección de

información de archivos localizados en la Web. El sistema propuesto utiliza técnicas de Scraping para rastrear y almacenar información contenida en bases de datos disponibles en sitios Web de forma pública.

Las descargas de información tienen como punto de inicio un grupo de direcciones URL y rutas XPath que permite rastrear la información dentro de la página Web. Una vez localizada la información requerida, ésta es recuperada y almacenada en una base de datos local, para estar disponible para el usuario.

Este artículo se encuentra organizado de la siguiente manera, la Sección 2 presenta el marco teórico y los trabajos relacionados. La Sección 3, muestra una descripción de la propuesta del sistema de recuperación. La Sección 4, presenta el algoritmo de recuperación de información. La Sección 5, describe las pruebas realizadas al algoritmo de recuperación de información. La Sección 6 presenta la discusión. Finalmente, la Sección 7 presenta las conclusiones y trabajos futuros.

2. Trabajos relacionados

Las técnicas de Web Scraping se utilizan para extraer información de sitios Web de manera automática mediante el análisis y manipulación de la estructura HTML. El Web Scraping rastrea, recupera y estructura información incrustada en etiquetas HTML de las páginas Web. Esto es una tarea compleja debido a los diferentes tipos de datos que se encuentran en los sitios Web [3].

La recuperación y almacenamiento unificado de los contenidos de diferentes sitios Web sobre un mismo dominio permite al usuario un rápido acceso a la información requerida, dando una mayor ventaja a la obtención de conocimiento encontrado en la Web. [4]. La búsqueda de trabajos relacionados permitió analizar los siguientes trabajos de investigación: El trabajo presentado por Zhao [5] muestra la descripción básica de un sistema de *Web Scraping,* mostrando el flujo de los procesos, la utilidad de las tareas y las áreas de aplicación de un *Scraping*.

El documento se divide en dos partes, la primera muestra la forma de realizar conexiones con diferentes fuentes de información. La segunda parte describe la forma de extraer información de estas fuentes. En el trabajo de Zhao se utilizaron herramientas de apoyo como Urllib2 y Selenium para realizar conexiones estables. Los datos encontrados pueden ser visualizados de forma previa a la extracción y la extracción y análisis de los datos se realiza de forma local.

El artículo puntualiza los problemas legales, derechos de autor, condiciones de servicio como restricciones y limitaciones en un Web Scraping. Rodríguez [6] realizó un análisis de 18 sistemas que ocupaban técnicas Web Scraping y que se encontraban en el mercado en el año 2019. Los sistemas fueron comparados de acuerdo con los servicios que cada uno ofrecía y los clasificaron en tres tipos: recursivos, buscadores, e-commerce. Rodríguez aplicó librerías de Java para scrapers recursivos.

Los formatos JSON se utilizaron para la transferencia de archivos, realizando pruebas con estas características se obtuvieron aproximadamente 100 mil resultados. El autor aplicó patrones de análisis al tipo e-commerce permitiéndoles extraer los precios de productos y servicios. El trabajo de investigación realizado por Uzun [7] analiza la

forma tradicional de hacer *Web Scraping* presentando diferentes aspectos como tiempo, metodología, accesos, extracciones, entre otros aspectos.

Uzun describe el diseño e implementación de un nuevo enfoque para *Web Scraping* denominado UzunExt, en el cual los métodos en cadena son la base para la extracción de datos mejorando los tiempos de respuesta. El enfoque propuesto es explicado en dos partes: En la primera parte se presenta las acciones de rastreo y extracción de datos. En la segunda parte, el autor aplica un análisis y procesamiento a los datos antes de ser almacenados. Uzun realizó experimentos de funcionalidad a una muestra de 100 sitios Web que contenían datos en diferentes idiomas y formatos.

Los resultados fueron favorables al utilizar el enfoque propuesto obteniendo 300 páginas y 4 patrones. El trabajo de investigación de Ram Sharan [8] presenta un análisis de aplicaciones tradicionales y aplicaciones basadas en la nube. El artículo establece que algunos de los problemas a los cuales se enfrentan los sistemas de *Web Scraping* son: restricciones de acceso, captcha, volúmenes enormes en los archivos, distribución heterogénea y la fiabilidad de los datos recuperados. El autor propuso la utilización de servicios Web proporcionados por Amazon, Elastic compute cloud y DynamoDB.

También se utilizaron los servicios Web que ofrece Selenio para el control del sistema de *Web Scraping*. Las pruebas las realizaron en 50 sitios Web pertenecientes a Amazon, Google shopping y otros sitios de comercio, desplegados en instancias de 20 y 30 máquinas utilizando un nodo o un nodo con varios hilos. Los sistemas basados en la nube arrojaron mejores resultados mostrando datos confiables, una reducción de costos significativa y mantenimientos más económicos.

El trabajo presentado por Glez-Peña [9] aborda la revisión de las fortalezas y limitaciones de las herramientas *Web Scraping* encontrando que los problemas de conexión con las bases de datos, la denegación de servicios Web y la demanda de datos masivos son algunas de las problemáticas comunes de estos sistemas. Glez-Peña presenta tres aspectos importantes para el análisis de las herramientas: el tiempo, la capacidad y la comunicación. Las librerías libcurl, Apache HttpClient, Jsoup, htmlcleaner, BeautifulSoup, scrapy, Web-Harvet se seleccionaron como herramientas de soporte debido a la compatibilidad que ofrecían.

3. Propuesta de un sistema de recuperación de información

Las técnicas de *Web Scraping* se utilizan para extraer información de sitios Web de manera automática mediante el análisis y manipulación de la estructura HTML. El *Web Scraping* rastrea, recupera y estructura información incrustada en etiquetas HTML de las páginas Web, lo cual es complejo [3] debido a los diferentes tipos de datos que se encuentran en los sitios Web.

La descarga y estructuración de estos contenidos da una comprensión mayor y una ventaja en la obtención de conocimiento de valor tomando en cuenta que los contenidos tienen volúmenes altos, actualizaciones veloces y una variedad de contenidos que demandan soluciones eficientes para su procesamiento. En este artículo presentamos nuestra propuesta de un sistema para recuperación de información (*Web Scraping*), el cual está organizado en 4 etapas (Figura 1):

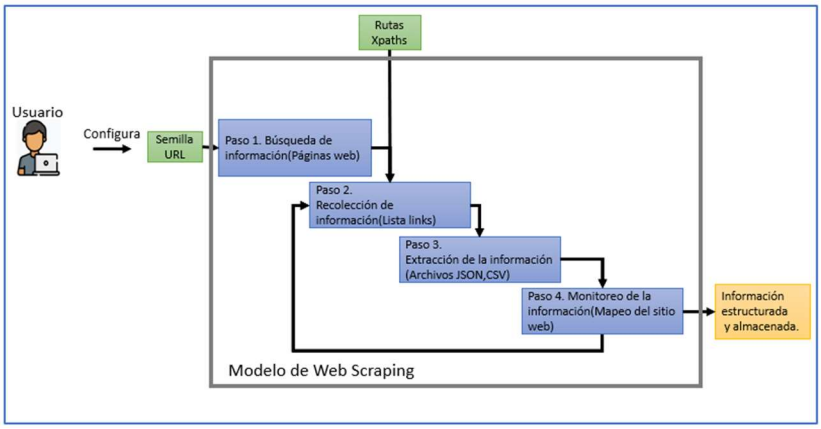

Fig. 1. Vista general del método propuesto.

Tabla 1. Ejemplo de configuración del Proyecto COVID CONACYT.

Nombre Proyecto	Dominio de búsqueda	URL semilla	Ruta XPath
COVID CONACYT	datos.covid-19.conacyt.mx	https://datos.covid-19.conacyt.mx/#DownZCSV	//*[@id="bPvM1"]/a/span [2] //*[@id="bPvM2"]/a/span [2]

- Búsqueda de información.
- Recolección de información del sitio Web.
- Descarga de información del sitio Web.
- Monitoreo de información del sitio Web.

Con el objetivo de validar nuestra propuesta hemos elegido un caso de estudio para la recuperación de información para COVID-19, la cual es una enfermedad respiratoria que se ha propagado por todo el mundo desde el 2019. Las dependencias de salud han sido los encargados de publicar la cantidad de casos confirmados, casos de defunciones, casos recuperados y la ubicación de estos de manera constante a través de los diferentes medios y canales de transmisión de información [10].

Uno de los medios más utilizados para compartir información del COVID-19 son las páginas Web de carácter gubernamental y de los servicios de salud, creando así múltiples fuentes de información que pueden ser consultados por investigadores para realizar diferentes estudios científicos referentes al COVID-19. El sistema de información propuesto permite la recolección de bases de datos publicadas en la Web que contienen información sobre el COVID-19 en México.

La información que se busca recolectar son casos positivos, casos de defunciones, casos negativos, casos sospechosos y la ubicación de estos índices, con el fin de generar un conjunto de datos unificado que pueda ponerse a disposición de investigadores que lo requieran para efectos de análisis de la información. Las fuentes de información

utilizadas para este caso de estudio son sitios de dependencias gubernamentales, así como de dependencias de salud, las cuales se listan a continuación:

- IMSS: https://coronavirus.gob.mx/.
- INSABI: https://datos.gob.mx .
- CONACYT: https://datos.covid-19.conacyt.mx.
- UNAM: https://covid19.ciga.unam.mx/.
- Instituto Politécnico Nacional: https://www.ipn.mx .
- Universidad de Hopkins: Https://data.humdata.org.
- Apple: https://covid19.apple.com.

4. Algoritmo de recuperación de información

El algoritmo de recuperación de información tiene como objetivo la búsqueda y recuperación de información de bases de datos almacenados en páginas Web. La descarga de los datos servirá como base para la creación de una base de datos homogénea que se pone a disposición de los usuarios del sistema para el análisis de datos utilizando, por ejemplo, técnicas de minería de datos. El algoritmo recuperación de información requiere una configuración inicial que se realiza la primera vez que se desea ejecutar el algoritmo. Dicha configuración requiere los siguientes datos:

- el nombre del proyecto de búsqueda.
- el dominio de búsqueda.
- la URL donde se realizará el proceso de obtención de datos.
- las rutas Xpath.

Las direcciones XPath son rutas utilizadas dentro de un documento XML para seleccionar nodos o conjunto de nodos que permiten navegar a través de ello [11]. Estas rutas Xpath permitirán encontrar los elementos dentro del XML recuperado y ejecutar interacciones con el sitio Web emulando las actividades realizadas por una persona dentro del sitio Web cuando se descarga un dato.

Adicionalmente se requiere indicar el tiempo que se estará ejecutando el sistema de *Web Scraping.* La tabla 1 muestra un ejemplo de esta información. En este ejemplo, el algoritmo, estará recuperando información de manera automática de los dominios especificados por el usuario durante el periodo indicado.

4.1. Búsqueda de información del sitio Web

La primera etapa del algoritmo de búsqueda de información del sitio Web tiene dos tareas principales: a) validar la existencia de la URL semilla proporcionada por el usuario, y b) iniciar un navegador Web de forma automática en el cual se realizará procesos de interacción con el sitio Web que permita ver el contenido oculto y recuperar las estructuras HTML de las URLs ingresadas por el usuario.

Cada una de las URLs debe pertenecer al mismo dominio para ser un conjunto de URLs válidas para la extracción. El siguiente código corresponde al ingreso de URLs

al sistema, la cual puede recibir una única URL o un conjunto de ellas siempre y cuando estas tengan el mismo dominio:

```
start_urls = [URL Semilla]

options = webdriver.ChromeOptions()

options.add_argument('--start-maximized')

options.add_argument('--disable-extensions')
```

En esta etapa el algoritmo inicia el requerimiento de acceso a los servidores de cada una de las URLs ingresadas, validando la existencia del sitio Web. Si la URL proporcionada no existe, se envía una notificación al usuario de *"URL no encontrada"*. En caso contrario, el algoritmo extrae una copia del sitio Web (su contenido HTML) en un archivo con extensión .XML y ejecuta un navegador Web de forma automática con características que permite eliminar extensiones o complementos que se tengan instalados en el navegador. Todo esto se lleva a cabo por cada URL proporcionada por el usuario. A continuación, se muestra el uso de una ruta *XPath* del sitio Web que permite realizar un clic en un botón de la página desencadenando una interacción:

```
WebDriverWait(driver, 20).until(

            EC.element_to_be_clickable((By.XPATH,

            '//*[@id="bPvM2"]/a'))) \

            click()
```

Finalmente, cada archivo .XML recuperado es transformado en un archivo JSON, lo que permitirá la comunicación y envío de los datos a través de la librería SELENIUM [12]. A continuación, se muestra el requerimiento y ejecución del navegador:

```
def extraer_datos():

    driver = webdriver.Chrome("./chromedriver.exe", op

             tions=options)

    # inicialiar el navegador

    for url in start_urls:

              driver.get(url)
```

4.2. Recolección de información del sitio Web

La segunda etapa del algoritmo tiene dos tareas principales: a) analizar e interactuar con el contenido de cada XML utilizando las direcciones *XPath* y b) identificar los enlaces de los archivos que se desean descargar.

En esta segunda etapa se toman las rutas *Xpath* para interactuar con el elemento que apuntan esas rutas aplicando un clic o selección de valores hasta encontrar una etiqueta

de dirección tipo *<a>,* identificando el enlace de direccionamiento asociado a esa etiqueta y guardándolo dentro de una lista de links de descarga.

Es importante hacer notar que el sistema debe replicar las acciones que realizaría un humano al intentar acceder a la información de un sitio, por ejemplo, desplegando y seleccionando opciones de diversos menús desplegables. A continuación, se muestra cómo se pasa una ruta *XPath* dentro de una función que permite recuperar los enlaces de direccionamiento de un archivo descargable:

```
Dato = WebDriverWait(driver, 20).until(

            EC.element_to_be_clickable((By.XPATH,

            '//*[@id="bPvM2"]/a'))) \

            .getattribute("href")

            Listalink.append(dato)
```

Al final de este proceso se contará con una lista de las direcciones de almacenamiento de los archivos que se pueden descargar.

4.3. Descarga de información del sitio Web

La tercera etapa del algoritmo tiene como objetivo realizar las descargas de la lista de links identificado en la etapa anterior. La descarga se lleva acabo, realizando una iteración en la lista de links recuperados y ejecutando un "clic" de forma automática sobre el enlace en turno, esto ejecutará un evento de descarga del archivo. Este proceso se repite por cada uno de los links almacenados en la lista de archivos a recuperar. A continuación, se muestra el código del algoritmo:

```
Def descargaArchivos(listalink):
    S=0
    For d in listalink:
            r= requests.get(d, allow_redirects=TRUE)
            file_name = 'ruta del archivo + un nombre + .csv'
            output = open(file_name, 'wb')
            output.write(r.content)
            output.close()
            s+1
            print(file_name)
```

4.4. Monitoreo de la información del sitio Web

La cuarta etapa del algoritmo tiene dos tareas principales: a) revisar de manera constante los sitios Web de los cuales se haya extraído información y b) detectar actualizaciones en el contenido del sitio Web que requieren un nuevo proceso de descarga de información. El monitoreo debe ser configurado por el usuario utilizando una de las dos siguientes opciones del sistema:

En la primera opción el usuario configura el periodo de tiempo en el cual el sistema ejecutara la descarga de los datos, el cual puede ser por minuto, horas o días. De esta forma, se ejecutará el proceso de recolección de información cuando se cumpla el periodo de tiempo establecido. El código asociado a este proceso es el siguiente:

```
Schedule.every(1).minutes.do(extraer_datos) #Cada 1 minuto ejecuta la
función de extraer_datos

While True:

            Schedule.run_pending()

            Time.sleep(1)
```

La segunda opción, el sistema realizará un mapeo del sitio Web, el cual sirve como punto de comparación para la revisión constante del sitio Web en busca de un cambio o actualización del sitio. Cuando se detecta un cambio el sistema se ejecuta nuevamente el proceso de descarga de información. A continuación, se muestra el código de la configuración por mapeo de sitio Web. En caso que en algún momento la información sea borrada, se informará al usuario del cambio o inconsistencia de los datos y se solicitara una nueva configuración:

```
Response = urlopen(url).read()

currentHash = hashlib.sha224(response).hexdigest()

time.sleep(30)

response = urlopen(url).read()

newHash = hashlib.sha224(response).hexdigest()

if newHash == currentHash:

            continue
```

5. Pruebas al algoritmo Web Scraping

Las pruebas realizadas al algoritmo *WebScraping* están enfocadas a la estrategia de búsqueda del caso de estudio COVID-19. La información que se busca recolectar son casos positivos, casos de defunciones, casos negativos, casos sospechosos y la ubicación de estos índices, con el fin de generar un conjunto de datos unificado que pueda ponerse a disposición de investigadores que lo requieran. Las pruebas realizadas fueron sobre cinco dominios (datos.covid-19.conacyt.mx, data.humdata.org, datos.gob.mx, covid19.apple.com y https://serendipia.digital), con un total de 8 rutas *Xpath*. La ejecución de la búsqueda se realizó el día 25 de marzo del 2022.

Tabla 2. Configuración de las pruebas realizadas al algoritmo *WebScraping*.

Nombre Proyecto	Dominio de búsqueda	URL semilla	Ruta *XPath*
COVID CONACYT	datos.covid-19.conacyt.mx	https://datos.covid-19.conacyt.mx/#DownZCSV	//*[@id="bPvM1"]/a/span[2] //*[@id="bPvM2"]/a/span[2]
Human data	data.humdata.org	https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases	//*[@id="data-resources-0"]/div/ul/li[1]/div[3]/a[1] //*[@id="data-resources-0"]/div/ul/li[2]/div[3]/a[1] //*[@id="data-resources-0"]/div/ul/li[3]/div[3]/a[1]
datosGob	datos.gob.mx	https://datos.gob.mx/busca/dataset/informacion-referente-a-casos-covid-19-en-mexico	//*[@id="datasets-list"]/div/div[3]/ul/div[1]/a
Covid19apple	covid19.apple.com	https://covid19.apple.com/mobility	//*[@id="download-card"]/div[2]/a
Serendipia	https://serendipia.digital	https://serendipia.digital/covid19-mx/datos-abiertos-sobre-casos-de-coronavirus-covid-19-en-mexico/	// *[@id ="datos_abiertos_table"]/table/tbody/tr['+ str(i) +']/td[3]/a

5.1. Configuración del Web Scraping

La configuración de los cinco dominios utilizados para las pruebas se muestra en la Tabla 2. En esta tabla se muestra el nombre del proyecto, el dominio de búsqueda, la URL semilla utilizada y sus correspondientes rutas *Xpath*.

5.2. Búsqueda de información del sitio Web

La búsqueda de información del sitio Web se llevó a cabo validando la existencia de la URL Semilla. No se encontró error en ninguna de las URLs del caso de estudio, En todos los casos fue posible establecer una conexión con el sitio Web, se recuperó la estructura HTML del sitio Web y se ejecutó un navegador Web de forma automática cargando todos los elementos de la página Web abierta.

5.3. Recolección de información

En esta etapa, el algoritmo toma como entradas las rutas *Xpath* e inicia un rastreo de estos elementos dentro del archivo .XML realizando interacciones con la página Web de forma automática sobre los elementos a los cuales apuntan los *XPath*. Una vez encontrado el enlace donde se almacenaba el archivo, se copia la ruta agregándola a la

['http://archivos.serendipiadata.com/data_covid/covid-19-mexico-220131.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-sospechosos-220131.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-220130.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-sospechosos-220130.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-220129.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-sospechosos-220121.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-220120.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-sospechosos-220120.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-220119.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-sospechosos-220119.csv', 'http://archivos.serendipiadata.com/data_covid/covid-19-mexico-220118.csv']

Fig. 2. Vista parcial de la lista de links encontrados dentro de una estructura HTML.

Tabla 3. Vista parcial de los datos encontrados en el Proyecto COVID CONACYT.

URL semilla	Página Web	Descripción	Archivos recuperados
https://datos.covid-19.conacyt.mx/#DownZCSV	Casos diarios por estado + nacional	Casos confirmados Casos sospechosos Casos negativos Casos defunciones	Casos_Diarios_Estado_Nacional Confirmados_20220309 Casos_Diarios_Estado_Nacional Sospechosos_20220309 Casos_Diarios_Estado_Nacional Negativos_20220309 Casos_Diarios_Estado_Nacional Defunciones_20220309 Casos_Diarios_Municipio_Confirmado 20220309 Casos_Diarios_Municipio_Sospechoso 20220309 Casos_Diarios_Municipio_Negativos_0220309 Casos_Diarios_Municipio_Defuncione 20220309

lista de archivos descargables. Este proceso se realiza de forma automática para cada uno de los elementos que se recuperaron, obteniendo al final una lista de links de direccionamiento donde se encontraban alojados los archivo a recuperar.

A continuación, Se presenta un ejemplo parcial de la lista de links encontrados dentro de la estructura HTML de un sitio Web.

5.4. Descarga de información del sitio Web

La recuperación de la información se realizó ocupando la lista de direcciones almacenadas en el paso anterior, la cual contiene las direcciones de los archivos

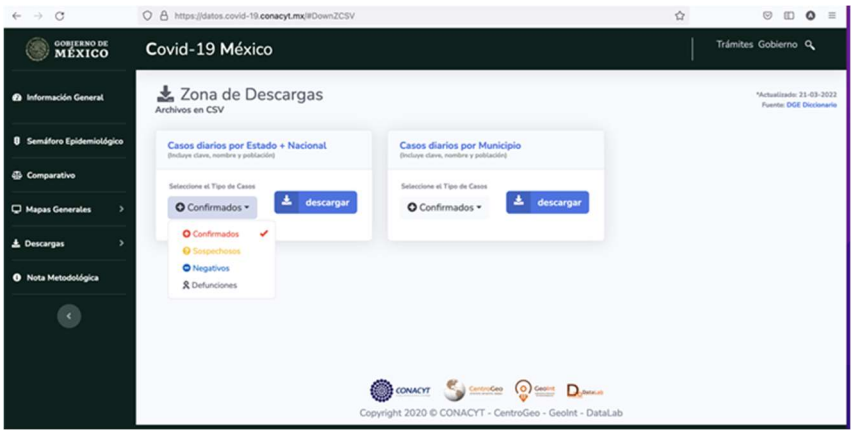

Fig. 3. Página Web con sus archivos para ser descargados por el Web Scraping.

descargables de cada sitio Web revisado. El sistema inicia el proceso de descarga ejecutando una solicitud de acceso a cada uno de los links recuperados, realizando un proceso de lectura y descarga del archivo almacenado en cada uno de los links. Este proceso se realiza por cada archivo de la lista de links de descarga de cada uno de los proyectos. En la Tabla 3, podemos observar el número de archivos descargados del día: 25-02-2022.

Los datos mostrados de manera parcial tienen ejemplos de los nombres de los archivos recuperados, una descripción y la ruta de donde se descargaron. Los archivos recuperados de la búsqueda del proyecto CONACYT fueron 12 archivos en formatos CSV. Los datos completos de la descarga de los cinco proyectos configurados se pueden consultar en la siguiente url[1]. Los archivos mostrados en la tabla son los recolectados en un solo día, por lo cual en páginas como la del CONACYT con actualizaciones cada 24 horas se puede recuperar un total de 56 archivos en un periodo de 7 días.

La página Web de la URL semilla: https://datos.covid-19.conacyt.mx/#DownZCSV presentada en la figura anterior muestra una zona de descargas en la que un usuario debe seleccionar las diferentes opciones para la descarga de casos confirmados, sospechosos, negativos y defunciones (Figura 3).

5.5. Monitoreo de la información del sitio Web

La etapa de monitoreo de la información del Sitio Web se realiza para cada proyecto de *Web Scraping* configurado. En la Tabla 2 se observan los nombres de los proyectos configurados para la descarga de datos COVID 19 con sus respectivos dominios. Este proceso de monitoreo se lleva a cabo una vez que el sistema de recuperación ha descargado la información requerida, monitoreando los cambios que existen en el sitio Web de la información.

[1] https://drive.google.com/file/d/10tN0gheKToix1d0jCRiPfOhD07U7hXTo/view?usp=sharing

Cuando el sistema detecta que existen cambios de la información descargada, entonces se descarga nuevamente la nueva información generada. Para el ejemplo presentado del COVID-19, el monitoreo se lleva a cabo cada 24 horas a partir de la primera extracción. Un análisis previo de los sitios Web permitió detectar los periodos de cambio de información dentro de los dominios utilizados.

6. Discusión

El caso de estudio del COVID-19 abordado permitió cuantificar y evaluar los resultados obtenidos en la utilización del *Web Scraping.* Este caso de estudio permitió generar diversas preguntas de investigación: ¿qué tipo de información se busca recuperar?, ¿cuáles son los formatos permitidos para realizar posteriores análisis de datos?, ¿la información obtenida sirve realmente como punto de partida para investigaciones?, ¿el *Web Scraping* permitió una descarga de información constante a un menor tiempo y recursos?, estas preguntas tuvieron respuestas interesantes para nuestro proyecto de investigación.

Las fuentes de información Web son la materia prima de este proyecto. Son estas URLs las que permiten la aplicación del *Web Scraping* obteniendo información verdaderamente relevante y de utilidad, por ello, la selección de las fuentes de información y las URL correctas son de vital importancia para la configuración de un nuevo proceso de *Web Scraping.* Los dominios de los sitios Web utilizados para las pruebas del sistema propuesto permitieron recuperar información de utilidad. Cada dominio contenía información sobre el covid-19 en México y en diferentes partes del mundo, con información en índices como casos positivos, sospechosos, negativos y defunciones.

Los 5 dominios analizados permitieron probar las técnicas *Web Scraping* en diferentes situaciones y estructuras de programación Web, cada dominio requirió una configuración inicial de proyecto permitiendo ver el desempeño del *Web Scraping* en diferentes situaciones. Los formatos de la información disponibles en estos sitios con datos COVID-19 son de tipo CSV. Este formato es el más utilizado por las diferentes dependencias para el informe de la evolución de esta pandemia, gracias a su facilidad de difusión, formato sencillo y claro. La información en este tipo de formatos es fácilmente manipulable y puede ser consumida por diferentes sistemas o algoritmos.

El tiempo que se requiere para la descarga manual de información de sitios Web con datos gigantes puede ser alta. La disminución de esfuerzo-horas-hombre delegando el trabajo de recolección de datos a un sistema computacional con reglas y métodos definidos aseguran que la información que se recolecta sea correcta según la configuración de parámetros y características definidas. Los proyectos de Web Scraping configurados recuperaron una cantidad de archivos considerable en cuatro horas. Las extracciones a los cinco dominios ingresados permitieron una recolección y descarga de 1050 archivos con información del COVID-19.

Estos archivos se encontraban almacenados dentro de enlaces de descarga en formatos CSV con estado público por lo cual son de libre acceso y pueden ser utilizados de forma abierta en diferentes proyectos sirviendo como base para investigaciones.

El tamaño de la información recuperada fue de 35 GB de información como datos crudos sin ningún tipo de pre-procesamiento.

7. Conclusiones y trabajo a futuro

El sistema de recuperación de información propuesto en este artículo permitió la recuperación de datos de utilidad para posteriores análisis de minería de datos. El sistema permite delegar la actividad de recolección continua de datos a partir de la Web para temas tan dinámicos como es el COVID-19. El sistema propuesto es de utilidad en entornos donde la información cambia en múltiples ocasiones a la largo del día. El sistema monitorea de forma continua el contenido de las páginas Web para detectar cambios y realizar el proceso de descarga de información.

Las 5 etapas del Web Scraping propuesto permite obtener la información rápidamente debido a las rutas XPath que apuntan directamente a la información que requiere el usuario. Los trabajos de Web Scraping como el presentado por Zhao [5] permiten que el sistema analice dentro de una página información relevante con respecto a una instrucción inicial. Mientras que nuestro Web Scraping realiza una recolección de información partiendo de una ruta XPath asegurando que la información que se recupera es realmente lo que necesita el usuario según sus configuraciones.

El sistema valida la existencia de dicha información y recuperándola según un periodo de tiempo establecido. Este Web Scraping se puede clasificar como un Scraper recursivo según la clasificación de Rodríguez [6], recuperando datos almacenados con una estructura en diferentes fechas. A diferencia de otros trabajos de Web Scraping como el presentado por Uzun [7] que utiliza patrones de recolección. Los trabajos futuros del trabajo de investigación están centrados en el mapeo automático del sistema propuesto con herramientas de minería de datos que tomen como entrada los datos recuperados por el sistema.

Referencias

1. Hernandez, A., Suarez, G., Sánchez, K., Toscano, V. M., Martínez, V., Sanchez-Perez, H.: A Web Scraping Methodology for Bypassing Twitter API Restrictions. (2018) doi: 10.48550/arXiv.1803.09875.
2. Vargas, A.: Herramienta para recopilar la información en las noti-cias publicadas en los sitios Web de internet. Serie Científica de la Universidad de las Ciencias Informáticas, vol. 10, no. 5, pp.14–24 (2017)
3. Suganya, R., Devi, D., Manjula, Siddharth, R. K.: An Efficient Approach for Web Indexing of Big Data through Hyperlinks in Web Crawling (2015)
4. Moreno, J. P.: Una aproximación a big data. Revista de De-recho UNED, no. 14, pp. 471–506 (2014)
5. Zhao, B.: Web Scraping. Encyclopedia de big data, pp. 1–3, Seattle: Springer (2017) doi: 10.1007/978-3-319-32001-4_483-1.
6. Villanueva, U. J.: Investigación y Desarrollo de Técnicas de Scraping. Universidad de Alcala, Escuela Politécnica Superior (2019)

7. Uzun, E.: A Novel Web Scraping Approach Using the Additional Information Obtained from Web Pages. IEEE Access, vol. 8, pp. 61726–61740 (2020) doi: 10.1109/ACCESS.2020.2984503.
8. Ram, C., Santosh, P., Sadhu, B., Subarna, S.: Cloud based Web scraping for big data applications. In: IEEE International Conference on Smart Cloud, pp. 138–143 (2017) doi: 10.1109/SmartCloud.2017.28.
9. Glez-Peña, D., Lourengo, A., López, H., Reboiro, M., Jato, F., Riverola, F.: Web Scraping technologies in an-API world. Briefings in Bioinformatics, vol. 15, no. 5, pp. 788–797 (2014) doi: 10.1093/bib/bbt026.
10. Garrido, A.: Estrategia general de búsqueda de información. asociación de enfermería española. pp. 30–32 (2005)
11. w3schools. w3schools. https://www.w3schools.com/xml/xpath_intro.asp.
12. Selenium.https://www.selenium.dev/documentation/webdriver/getting_started.