Miguel Casasnovas Bielsa
Marc Asenjo Papiol

Web Scraping en IMDb

Calificaciones y popularidad de videojuegos

Introducción

IMDb

Internet Movie Database (IMDb) es una amplia **base de datos en línea** que proporciona información relacionada con películas, programas de televisión, podcasts, vídeos y videojuegos, incluyendo valoraciones y reseñas de usuarios y profesionales. [1]

IMDb es considerada una de las fuentes más reconocidas, confiables y completas en la industria del entretenimiento [2], presentando más de 200 millones de visitantes mensuales. Además, la plataforma cuenta con más de mil millones de calificaciones y alberga información de más de 9 millones de títulos. [3] En particular, IMDb ofrece información detallada sobre los videojuegos recopilados, proporcionando detalles específicos como el título del juego, su género, la fecha de lanzamiento, el reparto, la guía parental, los premios, y las reseñas y calificaciones de los usuarios y críticos, entre otros.

La base de datos de IMDb se actualiza regularmente con datos de diversas fuentes y contribuciones de los usuarios, asegurando la precisión y fiabilidad de la información mediante controles rigurosos. [4, 5]

Por lo tanto, IMDb emerge como una fuente confiable y relevante para la extracción de datos relacionados con los videojuegos, lo que posibilita el análisis de datos para obtener conocimiento sobre tendencias, preferencias de los usuarios, entre otros aspectos relevantes de la industria del entretenimiento digital.

Objetivo del proyecto

El objetivo de este proyecto es obtener un conjunto de datos detallado y completo sobre los videojuegos listados en la plataforma IMDb mediante técnicas de *web scraping*. Específicamente, pretendemos recopilar información sobre una amplia variedad de videojuegos, incluyendo detalles generales como el título, el género y fecha de lanzamiento, así como las nominaciones, premios, y calificaciones de los usuarios.

El conjunto de datos resultante posibilitará un análisis en profundidad de la industria de los videojuegos, permitiendo identificar tendencias emergentes, patrones de popularidad o lanzamiento y relaciones entre atributos como las calificaciones de los usuarios y el género del juego, entre otros. Los hallazgos proporcionarán una mejor comprensión de las preferencias de la comunidad, beneficiando tanto a desarrolladores como a entidades del sector al permitirles tomar decisiones informadas y adaptarse eficazmente a las demandas del mercado.

La elección de IMDb como fuente de datos se fundamenta en su reputación como plataforma referente en la industria del entretenimiento, con una amplia base de datos y una comunidad activa de usuarios.

IMDb ofrece conjuntos de datos gratuitos para uso personal y no comercial con información actualizada diariamente, aunque su alcance es limitado. [6] Por otro lado, la plataforma proporciona conjuntos de datos masivos actualizados así como acceso a los datos a través de una API [7]. Sin embargo, ambos servicios requieren una suscripción con costes significativos, alcanzando precios de hasta 400,000$ anuales [3]. En consecuencia, hemos optado por realizar *web scraping* para obtener información detallada y accesible sobre los videojuegos sin incurrir en costos adicionales, permitiéndonos repetir y replicar el proceso de extracción de datos.

Es importante destacar que la información disponible en IMDb es de dominio público. Además, nos comprometemos a obtener los datos de manera ética, sin causar ningún perjuicio al servidor y sin fines comerciales.

Recursos

El código en Python empleado para extraer los datos de los videojuegos listados en IMDb mediante *web scraping* se encuentra disponible en un repositorio público en GitHub. Asimismo, el conjunto de datos obtenido está disponible tanto en GitHub como en Zenodo (10.5281/zenodo.10982152) para su acceso público. Además, se ha creado un video tutorial alojado en Google Drive, el cual ofrece una visión general del proyecto y simplifica la comprensión del proceso de *web scraping* (para visualizarlo se requiere de un correo asociado a la UOC)

Antecedentes

En el ámbito del *web scraping* en IMDb, se han desarrollado diversos proyectos utilizando *Python* y las bibliotecas `BeautifulSoup` para el análisis del contenido HTML, `requests` para realizar solicitudes HTTP, y `pandas` para el procesamiento y análisis de datos.

Específicamente, estos proyectos suelen enfocarse en el ámbito cinematográfico, proporcionando un *script .ipynb* para la tarea de *web scraping* que permita extraer datos del *top* películas listadas en IMDb:

- Web Scraping Movies from IMDB:
  En este proyecto se proporciona un *script* para extraer de forma automatizada las películas más populares de un género específico en formato CSV, descargando el contenido de *n* páginas que enumeran únicamente 50 títulos, mediante modificaciones en la URL. Es decir, con *n* igual a 5, se extraería el top 250 películas. Sin embargo, se limita a obtener la información incluida en la página de búsqueda avanzada de IMDb, como el *ranking*, título, URL, año de lanzamiento, duración, género, calificación, número de votos, director y elenco.
- Scraping TOP 100 Movies from the IMDB:
  En este proyecto se proporciona un *script* para extraer de forma automatizada únicamente el top 100 películas en formato XLSX, basándose en la puntuación de los usuarios. De manera similar al proyecto anterior, se limita a obtener la información listada en la página de búsqueda avanzada de IMDb, incluyendo el *ranking*, título, año de lanzamiento, duración, descripción, calificación, número de votos, director, elenco, *metascore* y recaudación.
- Scraping IMDB Top Movies using Beautiful Soup:
  En este proyecto se proporciona un *script* para extraer de forma automatizada únicamente el top 250 películas en formato CSV, basándose en la puntuación de los usuarios. A diferencia de los anteriores, este proyecto utiliza la lista proporcionada en las tablas de IMDb. Además, modifica el User-Agent para simular el comportamiento de un navegador web convencional. Sin embargo, únicamente extrae el *ranking*, título y calificación.

Por otro lado, hemos encontrado diversos conjuntos de datos relacionados con las calificaciones de los videojuegos listados en IMDb. Estos *datasets*, obtenidos mediante *web scraping*, presentan una menor cantidad de atributos en comparación con nuestro conjunto de datos y fueron extraídos en el año 2022:

- IMDB Video Games:
  Este *dataset* recoge alrededor de 20,800 títulos de videojuegos de 9 géneros en un archivo CSV, con el objetivo de conocer las tendencias de popularidad de los géneros de los videojuegos. Los 16 campos del conjunto de datos incluyen el título, el año de lanzamiento, la clasificación por edad, la calificación, el número de votos, la trama, así como el género, presentado mediante codificación binaria.
- Video game ratings from imdb:
  Este *dataset* recoge alrededor de 12,600 títulos de videojuegos en un archivo CSV, centrándose en proporcionar un conjunto útil para tareas analíticas

relacionadas con las puntuaciones de los videojuegos. Los 7 campos del conjunto de datos incluyen el título, año de lanzamiento, calificación, número de votos, trama, género y director.

En la literatura encontramos proyectos de análisis de datos que emplean conjuntos de datos obtenidos de IMDb mediante *web scraping*:

- IMDB Top1000 movie data analysis:
  Este proyecto aplica diferentes técnicas de análisis por redes sociales (SNA) para visualizar relaciones específicas en la industria cinematográfica, incluyendo los actores más recurrentes en las películas de un director en particular o la identificación del género predominante en la filmografía de ciertos actores, entre otros.
- IMDB Video Games - Answering Questions:
  Este proyecto emplea el dataset *IMDB Video Games* para comprender las tendencias y relaciones entre los diferentes géneros de videojuegos, las calificaciones recibidas, las preferencias de la audiencia y la evolución a lo largo del tiempo.
- A deep dive into the video games dataset:
  Este proyecto de minería de datos emplea el dataset *Video game ratings from imdb* para predecir la calificación de un videojuego mediante regresión, empleando los campos correspondientes al año y la cantidad de votos. El análisis revela que los videojuegos más recientes y aquellos con mayor número de votos tienden a obtener mejores calificaciones.

Evaluación inicial

Aspectos legales

IMDb permite un **uso personal y no comercial de sus datos**, sujeto a la aceptación de sus términos y condiciones de uso. [8] Sin embargo, IMDb establece en sus condiciones de uso que el uso de **herramientas de recopilación de datos** como *robots* o *screen scraping* **no está permitido sin su consentimiento expreso por escrit**o. [9] Por otro lado, el archivo ***robots.txt*** de IMDb establece restricciones de acceso a ciertas secciones del sitio web. No obstante, las rutas esenciales para nuestra tarea de web scraping, como el directorio de búsqueda o las páginas de detalles de los títulos, guía parental y calificaciones, no están sujetas a estas restricciones.

Tamaño

Es esencial estimar el tamaño de la página web objetivo para optimizar el proceso de *web scraping* y determinar el método de *crawling* más adecuado. En el caso de IMDb, se

estima que presenta alrededor de 152,000,000 páginas, de las cuales aproximadamente 61,600,000 se relacionan directamente con un título (véase las figuras 1 y 2).

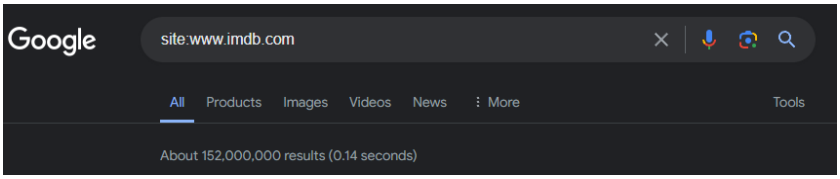

Fig. 1. Búsqueda avanzada de Google del dominio de IMDB.

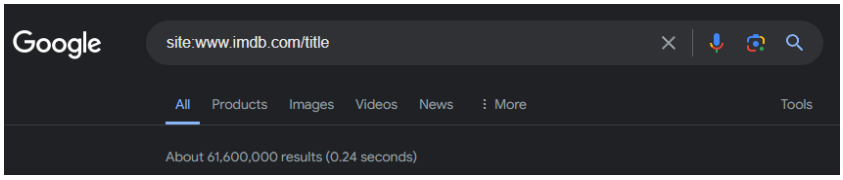

Fig. 2. Búsqueda avanzada de Google de títulos en IMDB.

Sin embargo, considerando que nuestro proyecto se enfoca en los datos de videojuegos, estimamos que el número de páginas a *scrapear* es de 152,057, una cifra no manejable dada la capacidad de procesamiento de nuestro sistema. Específicamente, IMDb contiene 38,014 títulos de videojuegos, de los cuales queremos rastrear las páginas de detalles, guía parental, calificaciones e imágenes. Además, queremos descargar el poster del videojuego y se debe considerar la página de búsqueda avanzada de títulos de IMDb que lista 50 títulos de videojuegos dinámicamente. Para mejorar la eficiencia del *scraper*, en lugar de rastrear secuencialmente con espaciado entre solicitudes HTTP para evitar ser bloqueados, lo haremos de forma concurrente e incluiremos en el rastreador un parámetro para indicar la cantidad de videojuegos a rastrear (*n*). En nuestro caso scrapearemos los 100 primeros videojuegos en el *ranking* de popularidad el día 16/04/2024.

Tecnología y propietario

La biblioteca **`whois`** en Python permite obtener información detallada sobre la propiedad del dominio https://www.imdb.com. IMDb es propiedad de IMDb.com, Inc., una subsidiaria de amazon.com [2] registrada en Washington, Estados Unidos (US) y gestionada por MarkMonitor, Inc.

```
import whois

print(whois.whois('https://www.imdb.com'))

# Out:
{
  "domain_name": [
    "IMDB.COM",
```

```
    "imdb.com"
  ],
  "registrar": "MarkMonitor, Inc.",
  "whois_server": "whois.markmonitor.com",
  ...
```

```
  "name_servers": [
    "NS1.AMZNDNS.CO.UK",
    "NS1.AMZNDNS.COM",
    "NS1.AMZNDNS.NET",
    "NS1.AMZNDNS.ORG",
    "NS2.AMZNDNS.CO.UK",
    "NS2.AMZNDNS.COM",
    "NS2.AMZNDNS.NET",
    "NS2.AMZNDNS.ORG",
    ...
  ],
  ...
  "org": "IMDb.com, Inc.",
  ...
  "state": "WA",
  ...
  "country": "US"
}
```

Por otro lado, la biblioteca **`builtwith`** en Python permite identificar las tecnologías utilizadas en el diseño web de IMDb, incluyendo herramientas como *D3* y *Javascript Infovis Toolkit* para el manejo de gráficos en JavaScript, así como los frameworks *Prototype*, *React* y *RequireJS*, junto con la biblioteca *basket.js* para la gestión de dependencias.

```
import builtwith

print(builtwith.builtwith('https://www.imdb.com'))

# Out:
{'javascript-graphics': ['D3', 'Javascript Infovis Toolkit'], 'javascript-frameworks': ['Prototype',
'React', 'RequireJS', 'basket.js']}
```

Análisis de la estructura de datos

El *sitemap* XML de una página web permite conocer la estructura de datos de la página web. Sin embargo, no hemos conseguido localizar el *sitemap* de IMDb mediante búsqueda manual de URLs como https://www.imdb.com/sitemap.xml, https://www.imdb.com/sitemap o https://www.imdb.com/sitemap_index.xml, ni utilizando operadores de búsqueda como *site:imdb.com filetype:xml*, *site:imdb.com*

inurl:sitemap o *site:imdb.com intitle:sitemap*. Además, el archivo *robots.txt* de IMDb no proporciona una referencia al *sitemap* de la página. [10]

Por otro lado, hemos inspeccionado la estructura de las páginas de interés para identificar las etiquetas HTML que contienen la información que buscamos extraer.
La URL inicial para el proceso de *web scraping* será la página de búsqueda avanzada de videojuegos de IMDb, ordenada por popularidad. La URL contiene los parámetros necesarios para filtrar específicamente los títulos de videojuegos, incluyendo aquellos considerados para adultos. En esta página, los títulos de videojuegos se listan dinámicamente, mostrando inicialmente 50 videojuegos y cargando 50 adicionales mediante un botón de carga dinámica. Una vez cargada la lista completa de videojuegos, el *scraper* procederá a descargar la página para obtener el enlace a la página de detalles de cada videojuego (***imdb_url***). La estructura de la página de búsqueda avanzada de videojuegos incluye un listado para cada videojuego, específicamente, un *tag* <*li*> de clase *ipc-metadata-list-summary-item* para cada uno. Dentro de cada una de estas etiquetas, la referencia a la página del título se encuentra contenida en el atributo *href* de la única etiqueta descendiente <*a*> de clase *ipc-title-link-wrapper* (véase la figura 3). De esta página obtendremos también la posición en el ranking, contenida junto al título del videojuego dentro de un tag <*h3*> de clase *ipc-title__text* que tendremos que parsear con expresiones regulares para obtener únicamente el *ranking*.

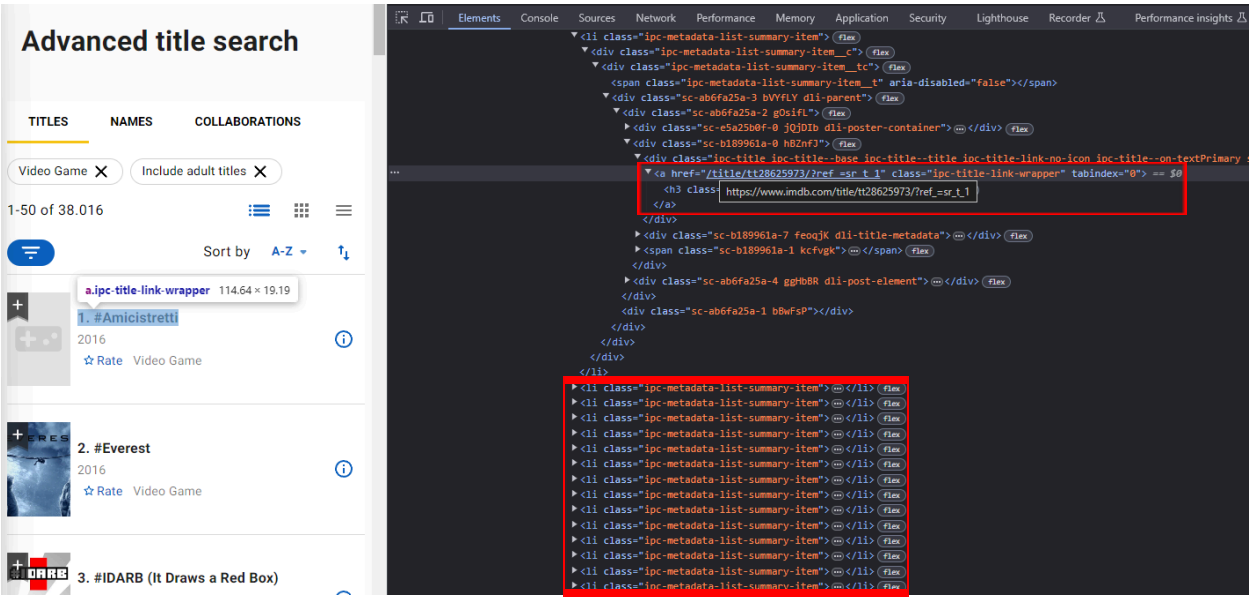

Fig. 3. Inspeccionar elementos en la página de búsqueda avanzada de IMDB: URL del título

Una vez obtenida la URL de la página de cada título, el *scraper* procederá a descargar la página para obtener información relevante de cada videojuego. Específicamente:

- **El nombre del videojuego (*title*)** se encuentra contenido dentro de una etiqueta ** de clase *hero__primary-text*, anidada dentro de un encabezado *<h1>* con clase *hero__pageTitle* (véase la figura 4).

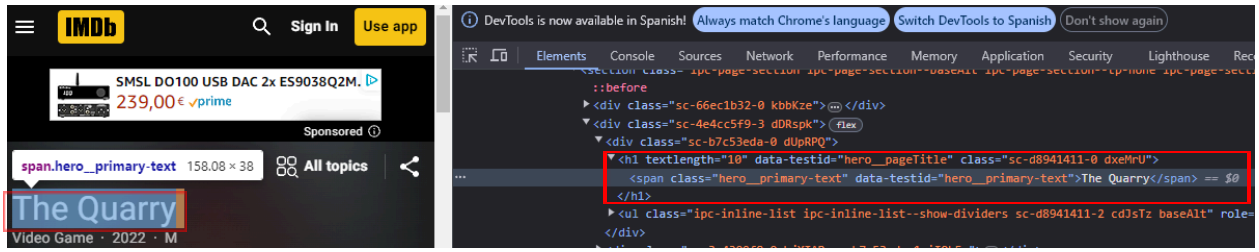

Fig. 4. Inspeccionar elementos en la página de un título en IMDB: Título

- **La fecha de lanzamiento del videojuego (*release_date*)** se encuentra contenida dentro de una etiqueta *<a>*, anidada dentro de una lista *<li>* con *data-testid title-details-releasedate* (véase la figura 5). Esta incluye el lugar de lanzamiento al que corresponde la fecha de lanzamiento, por consiguiente, habrá que parsear la cadena de texto y convertirla en una fecha.

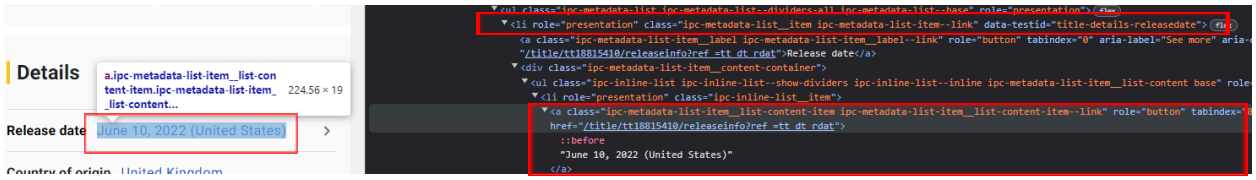

Fig. 5. Inspeccionar elementos en la página de un título en IMDB: Fecha

- **Los países de origen del videojuego (*countries*)** se encuentra contenida dentro de una etiqueta *<a>*, anidada dentro de la correspondiente lista *<li>* con *data-testid title-details-origin* (véase la figura 6).

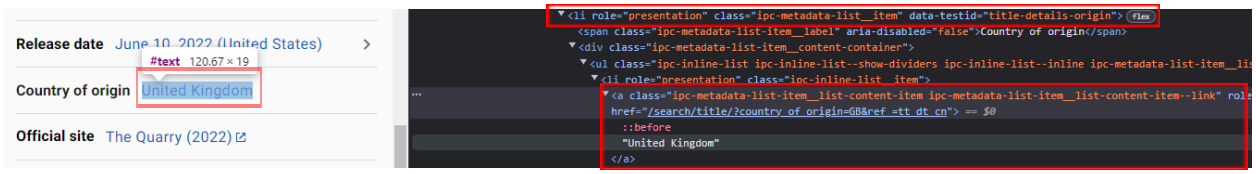

Fig. 6. Inspeccionar elementos en la página de un título en IMDB: Origenes

- **Las webs oficiales del videojuego (*sites*)** se encuentran contenidna en el *href* de una etiqueta *<a>*, anidada dentro de una lista *<li>* con *data-testid details-officialsites* (véase la figura 7).

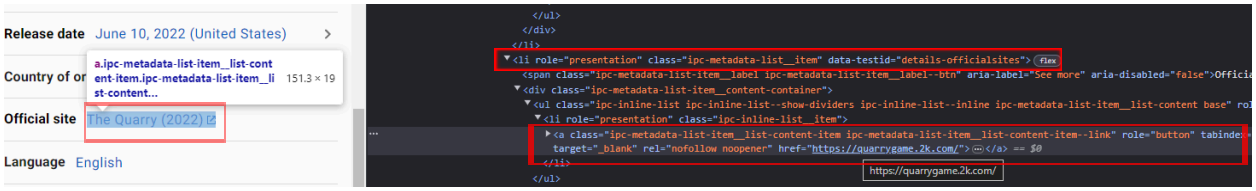

Fig. 7. Inspeccionar elementos en la página de un título en IMDB: Web

- **Los idiomas principales del videojuego (*languages*)** se encuentran contenidos dentro de una etiqueta *<a>*, anidada dentro de una lista *<li>* con *data-testid title-details-languages* (véase la figura 8).

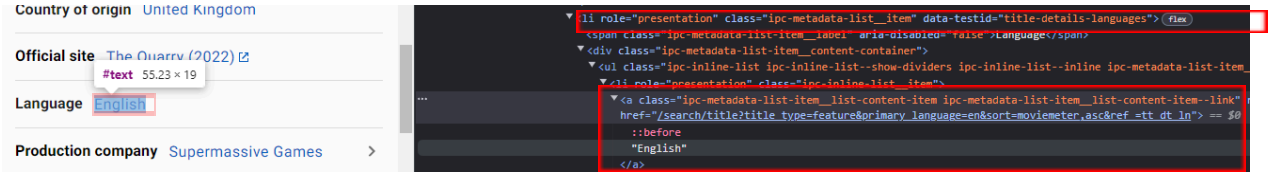

Fig. 8. Inspeccionar elementos en la página de un título en IMDB: Idioma

- **Las empresas productoras del videojuego (*companies*)** se encuentran contenidas dentro de una etiqueta *<a>*, anidada dentro de una lista *<li>* con *data-testid title-details-companies* (véase la figura 9).

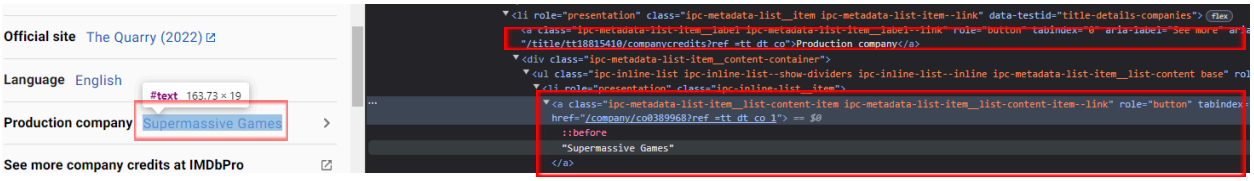

Fig. 9. Inspeccionar elementos en la página de un título en IMDB: Empresas

- **El elenco principal del videojuego (*top_cast*)** se encuentra contenido cada uno dentro de una etiqueta *<a>* con *data-testid title-cast-item__actor* (véase la figura 10).

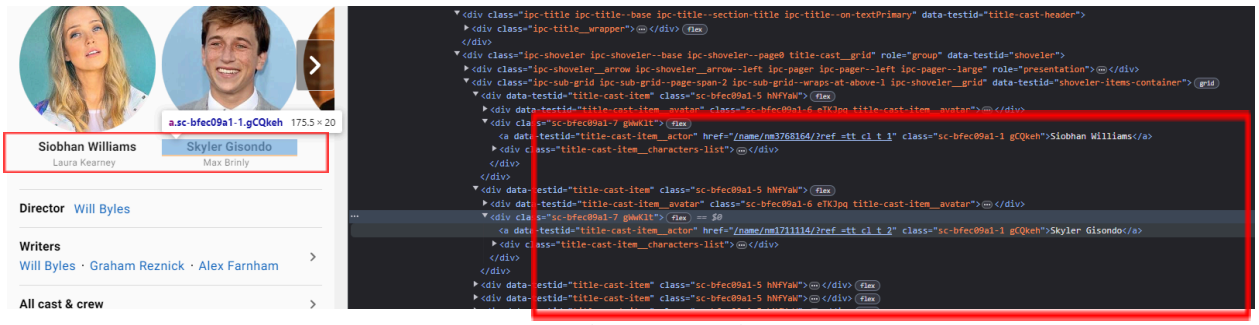

Fig. 10. Inspeccionar elementos en la página de un título en IMDB: Actores y actrices

- **Las nominaciones recibidas y premios obtenidos por el videojuego (*nominations, awards*)** se encuentran contenidos en un mismo string que habrá que *parsear* dentro de una etiqueta **, anidada dentro de un lista <li> con *data-testid award_information* (véase la figura 11).

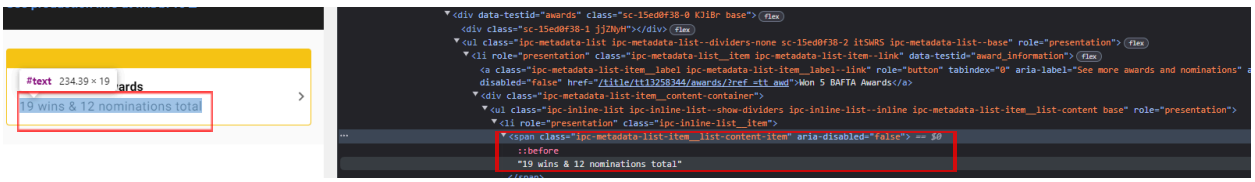

Fig. 11. Inspeccionar elementos en la página de un título en IMDB: Premios y nominaciones

- **Los géneros del videojuego (*genres*)** se encuentran contenidos dentro de una etiqueta *<a>*, anidada dentro de una lista *<li>* con *data-testid storyline-genres* (véase la figura 12).

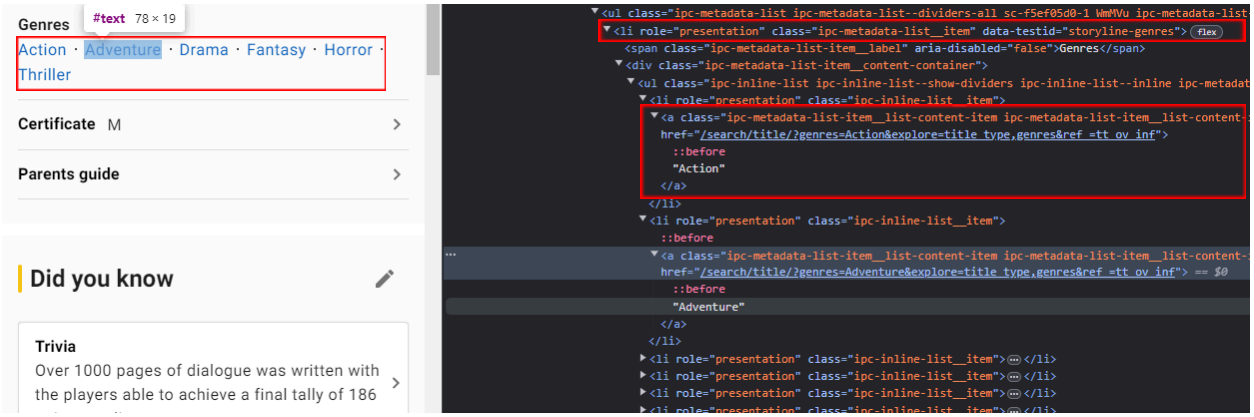

Fig. 12. Inspeccionar elementos en la página de un título en IMDB: Géneros

- **La URL a la página de control parental del videojuego** se encuentra en el atributo *href* de una etiqueta *<a>* con *aria-label Certificate: see all* (véase la figura 13).

Fig. 13. Inspeccionar elementos en la página de un título en IMDB: URL página control parental

- **La URL a la página de calificaciones del videojuego** se encuentra en el atributo *href* de una etiqueta *<a>* con *aria-label View User Ratings* (véase la figura 14).

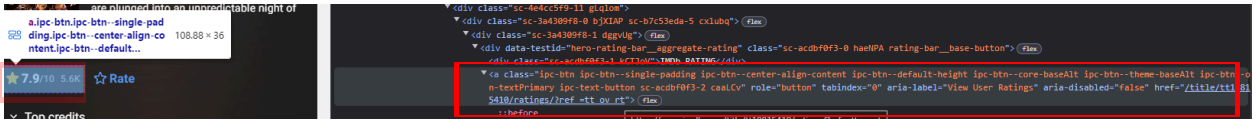

Fig. 14. Inspeccionar elementos en la página de un título en IMDB: URL página calificaciones

- **La URL a la página donde se encuentran las imágenes del videojuego** se encuentra en el atributo *href* de una etiqueta *<a>* contenida en un * * con *data-testid hero-media__poster* (véase la figura 15).

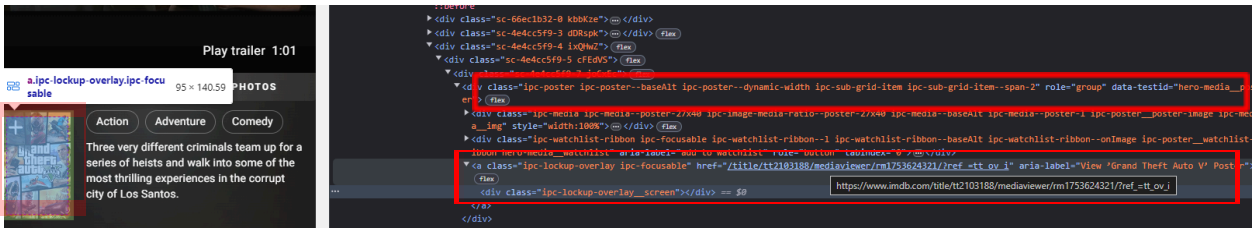

Fig. 15. Inspeccionar elementos en la página de un título en IMDB: URL página calificaciones

Una vez obtenida la URL de la página de control parental, el *scraper* procederá a descargar la página para obtener la siguiente información:

- **El nivel de contenido de desnudez o sexualidad en el videojuego (*nudity*)**
- **El nivel de violencia o gore en el videojuego (*violence*)**
- **El nivel de lenguaje soez o vulgar en el videojuego (*profanity*)**
- **El nivel de presencia de consumo de alcohol, drogas o tabaco en el videojuego (*alcohol*)**
- **El nivel de escenas de terror o intensas en el videojuego (*frightening*)**

Estos datos se encuentran contenidos cada uno dentro de una etiqueta ** específica, anidada dentro de una sección *<section>* con *id advisory-nudity*, *advisory-violence*, *advisory-profanity*, *advisory-alcohol* y *advisory-frightening*.

Por otro lado, una vez obtenida la URL de la página de calificaciones, el *scraper* procederá a descargar la página para obtener la siguiente información:

- **La calificación promedia ponderada (*rating*)** se encuentra contenida dentro de una etiqueta ** con clase única *sc-5931bdee-1 gVydpF*, anidada dentro de una sección * * con *data-testid rating-button__aggregate-rating__score* (véase la figura 16).

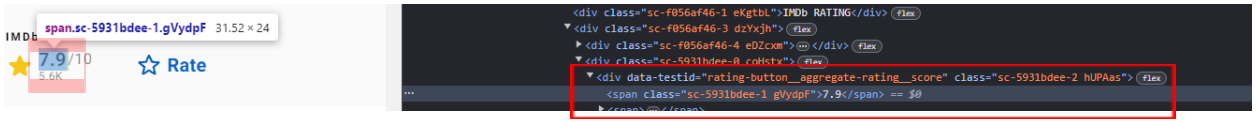

Fig. 16. Inspeccionar elementos en la página de un título en IMDB: Calificación promedia

- **La cantidad de votos de los usuarios para cada calificación (*user_ratings*)** se encuentra contenida dentro de una etiqueta *<tspan>*, anidada dentro de un *<text>* con *id chart-bar-1-labels-**, dónde * es el identificador de cada calificación (véase la figura 17). Como se observa, habrá que parsear el contenido para extraer el número de votos.

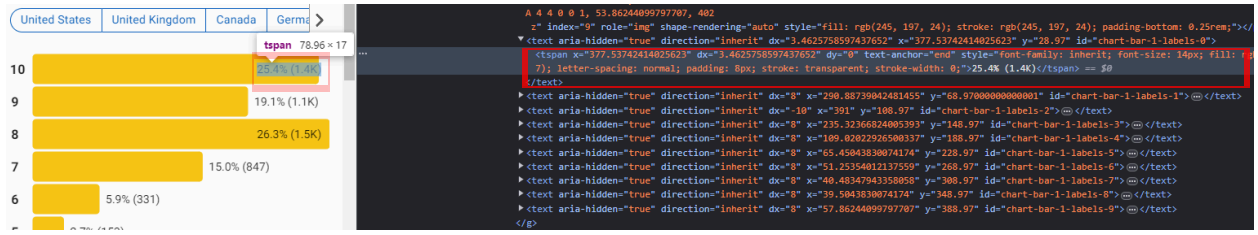

Fig. 17. Inspeccionar elementos en la página de un título en IMDB: Votos por calificación

Una vez obtenida la URL de la página con imágenes de video juego, el *scraper* procederá a descargar la página para descargar y almacenar en la carpeta *img* la imagen de portada de cada videojuego. La imagen de portada se encuentra en una etiqueta *<img>* con atributo *data-image-id* acabado en *curr*, a diferencia del resto que acaban en *prev*.

Estrategias de *web scraping*

Durante el proceso de *web scraping* en IMDb, es importante tener en cuenta posibles métodos de prevención del *scraping* implementados por el sitio web. A continuación, se presentan algunos de los métodos más comunes, incluyendo estrategias para superarlos:

- Los sitios webs pueden implementar **bloqueos de direcciones IP** que generen un alto volumen de solicitudes en un corto período. Por consiguiente, es recomendable el uso de múltiples direcciones IP para evitar ser bloqueado. Esto se puede lograr cambiando manualmente la dirección IP periódicamente o utilizando una lista de proxies. Además, se pueden implementar estrategias para espaciar aleatoriamente las solicitudes HTTP para simular el comportamiento humano y evitar sobrecargar el servidor.
- Los sitios webs pueden implementar **sistemas de CAPTCHA** para verificar si el usuario es humano y evitar el acceso automatizado, generalmente mediante un desafío visual o escrito. En caso de que IMDb presente estos mecanismos de prevención de bots, el *scraper* debería ser capaz de detectarlo y resolverlo de forma automatizada.
- Los sitios webs pueden implementar ***honeypots* o trampas de araña**. Los honeypots son áreas de una página web diseñadas específicamente para atraer bots de *web scraping*. Las trampas de araña son situaciones en las que un *bot* queda atrapado en un bucle infinito de enlaces al intentar acceder a páginas web dentro de un sitio, como contenido generado dinámicamente, bucles de redireccionamiento infinitos o parámetros de URL dinámicos. En caso de que IMDb implemente estas trampas, el *web scraper* debería actuar como un usuario real. En específico, hemos detectado que el contenido en la página de búsqueda avanzada se genera dinámicamente. En este caso el *bot* deberá interactuar con

el botón para mostrar todos la lista completa, realizando las interacciones con cierto espaciado.
- Los sitios webs pueden introducir **variaciones sutiles en su estructura HTML/CSS**, dificultando la automatización del proceso de *web scraping*.

Además de las técnicas anteriormente mencionadas para la resolución de obstáculos en el proceso de *web scraping*, se puede modificar el User-Agent y otras cabeceras HTTP para simular el comportamiento de un navegador web convencional o configurar *timeouts* u otras excepciones para manejar situaciones inesperadas durante la extracción de datos, como conexiones interrumpidas o respuestas lentas del servidor.

Por otro lado, es recomendable respetar el archivo *robots.txt*, rastrear solo información pública y utilizar la información extraída de manera justa y no comercial. En nuestro caso, cumplimos con las restricciones del archivo *robots.txt* de IMDb y utilizaremos el conjunto de datos extraído con carácter personal. Sin embargo, es importante resaltar que IMDb no proporciona una casilla de verificación explícita o enlace del tipo *«Acepto»* que obligue al *scraper* a aceptar activamente los términos.

Conjunto de datos: IMDb Popularity Video Games Dataset

Descripción

El conjunto de datos extraído el 15 de abril de 2024 se presenta en formato JSON, aunque también es extraíble en formato CSV. Consiste en 92 tuplas y **15 campos**, incluyendo los datos de videojuegos obtenidos por el *scraper* el día 16/04/2024 del top 100 en el *ranking* de popularidad de IMDb. El comando ejecutado fue el siguiente: *python -m src.main 100 'json' True*. Es decir, el scraper ha conseguido extraer efectivamente el 92% de los videojuegos en el *ranking* en 2057.72 segundos, es decir, 34 minutos aproximadamente. Sin embargo, algunos títulos de videojuegos no se pudieron extraer debido a una excepción de tipo *timeout* que aún no hemos logrado resolver.

Entre los campos incluidos en el *dataset* se encuentran:
- **title**: Nombre del videojuego *(str)* e.g., *The Quarry*
- **ranking**: Posición en la clasificación de popularidad del videojuego *(int)* e.g., *1*
- **release_date**: Fecha de lanzamiento del videojuego, incluyendo el día, mes y año y país *(timestamp)* e.g., *1447113600000*
- **countries**: Países de origen del videojuego (list:*str*) e.g., *United Kingdom, Belgium*

- **sites**: Lista de URLs de la web oficial del videojuego (list(*str*)) e.g., *https://quarrygame.2k.com/*
- **languages**: Idiomas principales del videojuego (list(*str*)) e.g., *English*
- **genres**: Géneros del videojuego (*list(str)*) e.g., *Action, Adventure, Drama, Fantasy, Horror, Thriller*
- **companies**: Empresas productora del videojuego (list(*str*)) e.g., *Supermassive Games*
- **top_cast**: Elenco principal del videojuego (*list(str)*) e.g., *Siobhan Williams, Ted Raimi, Miles Robbins, ...*
- **nominations**: Nominaciones recibidas por el videojuego (*int*) e.g., *4*
- **awards**: Premios obtenidos por el videojuego (*int*) e.g., *0*
- **parental_guide**: Nivel de contenido de desnudez o sexualidad (*nudity*), violencia o gore (*violence*), lenguaje soez o vulgar (*profanity*), presencia de consumo de alcohol, drogas o tabaco (*alcohol*), y de escenas de terror o intensas (*frightening*) (dict(str:*str*)) e.g., *nudity: Mild, violence: Severe, ...*
- **rating**: Calificación promedia ponderada (*float*) [11] e.g., *7.9*
- **user_ratings**: Distribución de votos de los usuarios, calificación:votos (*dict(int:str)*)e.g., *10: 1.4K, 9: 1.1K, ..., 1: 122*
- **imdb_url**: Enlace a la página correspondiente del videojuego en IMDb (*str*) e.g., *https://www.imdb.com/title/tt18815410*

A continuación, se presenta una representación gráfica del dataset:

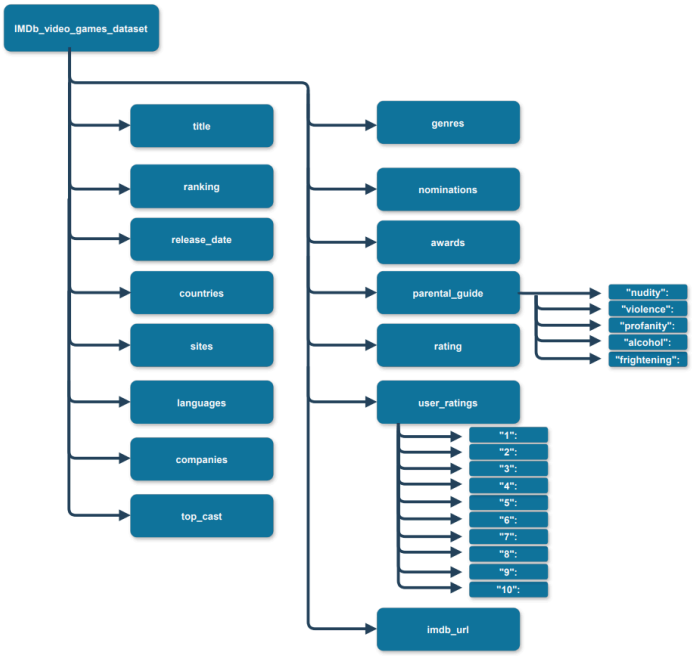

Fig. 18. Representación gráfica de la estructura del dataset

Los datos del conjunto son válidos, completos y coherentes, es decir, se ajustan a la sintaxis definida y no presentan datos duplicados o inconsistentes. Sin embargo, presentan algunos valores vacíos lo que requiere de una limpieza apropiada.

Además del conjunto de datos que contiene información detallada sobre los videojuegos enumerados en IMDb, se ha llevado a cabo la extracción de las portadas de cada videojuego mediante *web scraping*. Las imágenes están disponibles en la carpeta *img* del repositorio de GitHub asociado al proyecto. Cada imagen ha sido nombrada con el título del videojuego correspondiente.

Licencia y uso

Considerando que IMDb establece en sus condiciones de uso que el uso de herramientas de recopilación de datos no está permitido sin su consentimiento expreso por escrito, la licencia que hemos seleccionado solamente permite un **uso personal y no comercial de los datos con atribución específica a IMDb (los datos son para fines educativos)**: CC BY-NC-SA 4.0 DEED [13]. De acuerdo con esta licencia, debes dar el pertinente crédito a IMDb como titular de los derechos de autor del trabajo orginal; proporcionar un enlace directo a la licencia e indicar en todo momento si se han realizado cambios. En cuanto al significado de las siglas, junto con el de la licencia, se encuentra en el contenido referenciado.

Las premisas de esta licencia son 3:

- **BY**-Cualquier contenido derivado tiene que dar crédito al creador.
- **NC**-Solo se permite la explotación con fines no comerciales.
- **SA** -Estas mismas condiciones se aplican a cualquier mezcla, transformación, o cambio en general, sobre el dataset.

Nos gustaría comentar que en un principio estuvimos investigando la posibilidad de escoger dos licencias diferentes: una de uso personal y explotación no comercial para el dataset; y otra para el script, en la que nos reserváramos la autoría total del proyecto, pero fuera de libre uso. Sin embargo, considerando que esto implicaba la creación de dos repositorios, hemos decidido emplear únicamente una licencia para ajustarnos a los requisitos de la práctica.

A pesar de que algunos de los proyectos anteriores, mostrados en el apartado de "Antecedentes", tienen una licencia *Released Under CCo: Public Domain License* (véase IMDB Video Games o Video game ratings from imdb), consideramos que la mejor licencia es una que imponga restricciones de uso comercial. Adicionalmente, hemos

añadido en el README la siguiente línea que estipula que los datos provienen de IBdm:"***Information courtesy of IMDb (https://www.imdb.com).***"

Código

El *scraping* se ha realizado usando Python y las bibliotecas **`Selenium`** para el análisis y extracción del contenido HTML, **`requests`** para realizar las solicitudes HTTP de descarga de imágenes, **`threading`** para procesar las tareas de scraping de manera paralela y **`pandas`** para el procesamiento de datos.
A continuación, se encuentra un diagrama representando el flujo del proceso de *scraping* desarrollado:

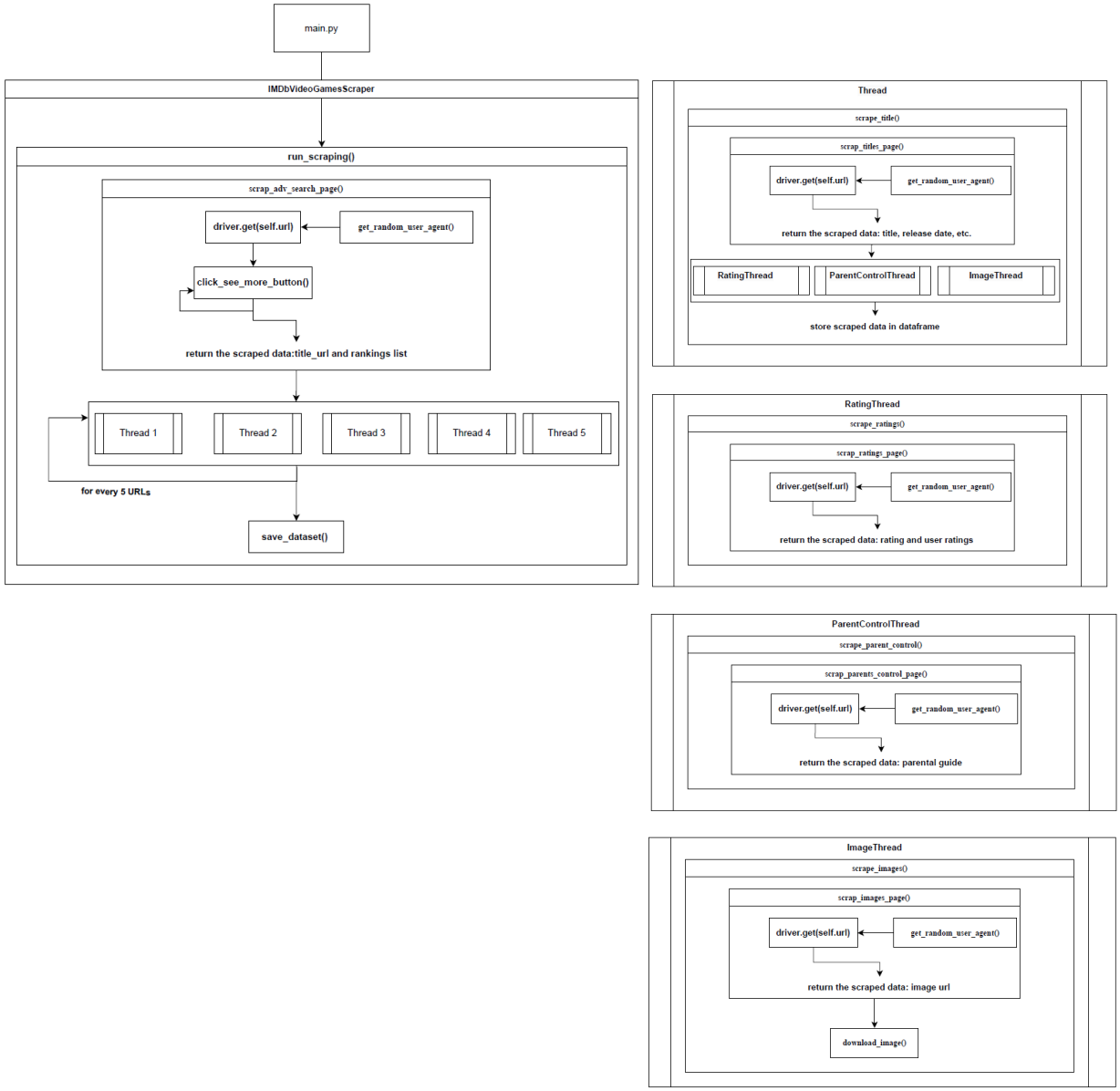

Fig. 19. Diagrama de flujo del proceso de *scraping*

El programa inicia con el método **run_scraping** que organiza el proceso de *scraping* gestionando la concurrencia.
En primer lugar, llama a la función que rastrea la página de búsqueda avanzada para obtener una lista con las URL de videojuegos y otra con las posiciones en el *ranking*. Esta función (scrap_adv_search_page()) se encarga de pulsar el botón de "50 more" de la página, que carga dinámicamente la lista de videojuegos. El número de pulsaciones al botón se realiza en base a la entrada que define el número de títulos de videojuegos a *scrapear* (*n*). Además, presentamos en pantalla el *User-Agent* utilizado por el *scraper*.

Una vez obtenida la información de la página de búsqueda avanzada,se divide esta en lotes de 5 para su procesamiento simultáneo mediante 5 *threads*. Hemos limitado el número de *threads* activos a 5 para no saturar el servidor. Cada lote se procesa de forma concurrente en un hilo separado utilizando *ThreadPoolExecutor*. [14] Estos *threads* se encargan primeramente de llamar a la función encargada de *scrapear* la página de título asignada.
Esta página contiene gran parte de la información del dataset y permite obtener la URL del resto de páginas a *scrapear*. Una vez extraída la información de esta página se ejecutan 3 threads en paralelo para *scrapear* las páginas de valoraciones, control parental y de imágenes. Todas las funciones encargadas de la tarea de scraping en primer lugar obtienen un *User-Agent* aleatorio de una lista que hemos definido que incluye los User-Agents de nuestros navegadores (Chrome, Edge y Firefox)extraídos mediante los motores de búsqueda (véase la figura 20) y un conjunto de User-Agents extraídos de fuentes externas [15] y validados mediante la herramienta en [16].

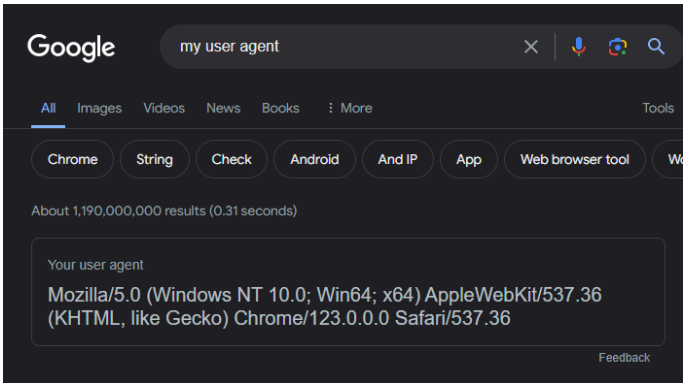

Fig. 20. Cálculo del User-Agent de google chrome

A continuación, emplean Selenium para abrir un navegador Firefox y *scrapear* posteriormente la información de interés contenida en el HTML de las páginas usando principalmente CSS selectors (By.CSS_SELECTOR), pero también la búsqueda por nombre de la clase (By.CLASS_NAME). La navegación en la estructura anidada de

clases y etiquetas de HTML se ha llevado a cabo en base al *Análisis de la estructura de datos* de las páginas objetivo de IMDb.
Cabe destacar que una vez obtenida la URL de la imagen que contiene el poster del videojuego, esta se descarga empleando la biblioteca requests y escribiendo el contenido en la carpeta *img*.
Una vez se obtienen todos los datos de las diferentes páginas para todos los videojuegos, se convierte la lista en un DataFrame de la biblioteca **`pandas`** y posteriormente se guarda en el formato indicado en la entrada, es decir, JSON o CSV.

Cabe destacar que durante el proceso de análisis de la estructura HTML de las páginas, reparamos en que, a través de la funcionalidad *copy* (accesible pulsando con el botón derecho del ratón sobre un elemento), se podía copiar el *selector* o el *xpath* (véase la figura 21), facilitando así el acceso a los elementos de interés mediante la función *find_element()* o *find_elements()* de Selenium. Sin embargo, decidimos descartar este uso debido a que se basaban en la posición relativa de los elementos en la estructura y esta podía ser variable según la información de los videojuegos.

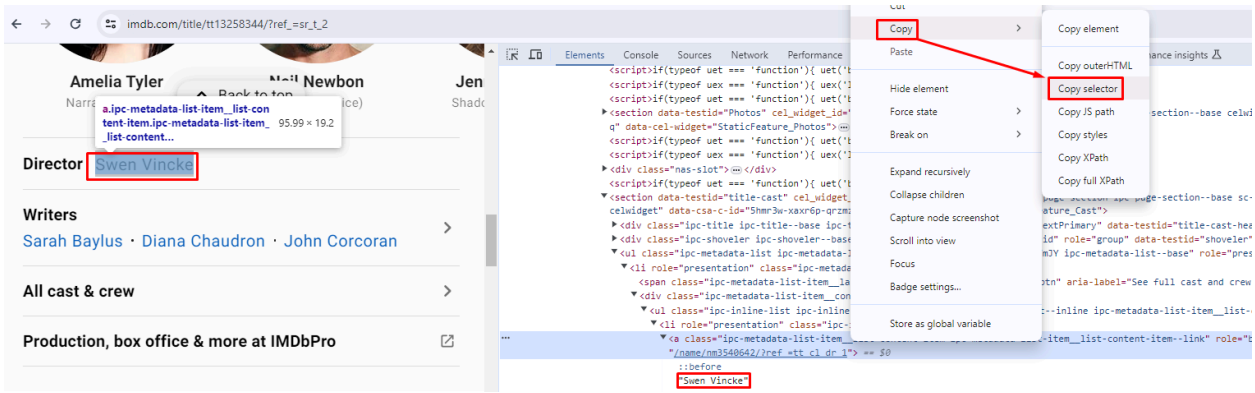

Fig. 21. Obtención del *selector* a través de "copy"

Algunas de las buenas prácticas desarrolladas en el código para *scrapear* de manera efectiva las páginas han sido las siguientes:

- **Modificar el User-Agent** en cada solicitud HTTP para simular el comportamiento de un navegador web convencional
- **Configurar *timeouts* y otras excepciones** para manejar situaciones inesperadas durante la extracción de datos, como conexiones interrumpidas o respuestas lentas del servidor.
- Hemos seguido los consejos indicados en [17], como el ***threading*** o programación concurrente y el **bloqueo de la carga de imágenes en las páginas**. La implementación de procesos en paralelo ha logrado una notable mejora en el tiempo de scraping. Anteriormente, la extracción de datos de cada videojuego requería aproximadamente 60 segundos. Ahora, el tiempo necesario para obtener información de un solo videojuego se ha reducido a 38 segundos,

mientras que el proceso para un grupo de 5 videojuegos ha disminuido a 72 segundos en total. Sin embargo, aunque indican que Chrome es más óptimo como navegador para la extracción de datos, hemos encontrado ciertos problemas con este navegador a la hora de pulsar el botón de "50 more".

- Inicialmente habíamos desarrollado la tarea de *web scraping* secuencialmente con **espacios entre solicitudes HTTP basados en el tiempo de procesamiento de las peticiones**. Finalmente, en el código final que realiza el proceso de scraping concurrentemente, hemos incluido esta buena práctica únicamente entre lotes de 5 threads, espaciando así las solicitudes cada vez que se *scrapean* las páginas asociadas a 5 videojuegos: *time.sleep(0.5 * response_delay)*. De esta manera,evitamos la sobrecarga del servidor y reducimos la posibilidad de bloqueo, manteniendo una ejecución más efectiva.

Las dificultades principales surgieron al obtener contenido generado dinámicamente, especialmente con el botón "50 more". Inicialmente, Chrome no pudo localizar el elemento, incluso después de probar varios métodos. La solución fue cambiar al navegador Firefox. Sin embargo, con Firefox, también nos enfrentamos a errores: el elemento no era clickable (*Element is not clickable at point*) o no estaba visible en la ventana (*target * is out of bounds of viewport*). Para superar esto, utilizamos un *Wait* de Selenium para esperar a que el botón sea *clickable*, maximizamos la ventana, desplazamos la ventana hacia abajo para ubicar el botón [18], esperamos un segundo y finalmente hicimos clic en el botón mediante una orden JavaScript.
Por otro lado, inicialmente enfrentamos dificultades al extraer el texto de ciertos elementos, requiriendo el uso de la función *.get_attribute("textContent").strip()* [20]. Además, encontramos problemas de compatibilidad entre el interpretador python que usábamos en PyCharm y Selenium, teniendo que cambiar al interpretador de Anaconda.

Para más detalles sobre código, consulte el archivo *readme.md* y los docstrings y comentarios del código.

Contribuciones

Contribuciones	**Firma**
Investigación previa	Miguel Casasnovas Bielsa, Marc Asenjo Papiol
Redacción de las respuestas	Miguel Casasnovas Bielsa, Marc Asenjo Papiol
Desarrollo del código	Miguel Casasnovas Bielsa, Marc Asenjo Papiol
Participación en el video	Miguel Casasnovas Bielsa, Marc Asenjo Papiol

Referencias

[1] Wikipedia. IMDb
[2] IMDb Help Center. What is IMDb?
[3] AWS Marketplace. IMDb
[4] IMDb Help Center. Where does the information on IMDb come from?
[5] IMDb Help Center. Adding new data
[6] IMDb Developer. IMDb Non-Commercial Datasets
[7] IMDb Developer. API Product Overview
[8] IMDb Help Center. Can I use IMDb data in my software?
[9] IMDB. IMDb Conditions of Use
[10] Semrush Blog. What Is a Sitemap? Website Sitemaps Explained
[11] IMDb Help Center. Weighted Average Ratings
[12] StackOverflow. Selenium - MoveTargetOutOfBoundsException with Firefox
[13] CreativeCommons. CC BY-NC-SA 4.0 DEED
[14] Docs Python. concurrent.futures — Launching parallel tasks
[15] ZenRows. Updated User-Agents List
[16] UserAgentString. User-Agent Analyser
[17] ZenRows. Selenium Slow? Discover Why and How to Speed Up
[18] SeleniumScrollDown. Scroll at the bottom of the page
[19] StackOverflow: Scraping webpage with show more button
[20] StackOverflow. Check the checkbox label using Selenium and behave in Python

Recursos

Subirats L. & Calvo M. (2019). Web scraping. UOC
Penman, R. (2015). Web scraping with python : Successfully scrape data from any website with the power of python. Packt Publishing, Limited.
Selenium Python. Getting started
Selenium Python. Navigating
Selenium Python. Locating Elements
Selenium Python. Waits
Selenium Python. WebDriver API
Selenium Python. FAQ
Developer Mozilla. CSS selectors
Developer Mozilla. Regular expression syntax cheat sheet
ZenRows. Web Scraping with Selenium and Python in 2024
ZenRows. Change the Selenium User Agent: Steps & Best Practices
Tutorialspoint. How to get userAgent information in Selenium Web driver
GeeksforGeeks. set_page_load_timeout driver method – Selenium Python