SRPP

(Sistema de Recomendación Personalizado de Películas)

Informe final del proyecto

Bases de Datos No Relacionales

3º Curso de Ciencia e Ingeniería de Datos
Escuela de Ingeniería Informática
Universidad de Las Palmas de Gran Canaria

Aris Vazdekis Soria y Alejandra Ruiz de Adana Fleitas

MEMORIA SOBRE EL PROYECTO SRPP

El Sistema de Recomendación Personalizado de Películas tiene como objetivo recopilar y analizar datos de reseñas de películas de la fuente en línea IMDb, una página web que contiene reseñas de miles de películas. Esta recopilación ha sido realizada de manera manual y actualizada para asegurar la calidad. También ha sido filtrada para asegurar que los datos tengan consistencia antes de ser introducidos en una base de datos MongoDB.

En conjunto con esta recopilación y almacenaje, se ha desarrollado un algoritmo de recomendación en dos fases completamente ajustado al conjunto de datos para que este lo analice correctamente. Dicho algoritmo tiene como propósito sugerir a los usuarios recomendaciones personalizadas sobre películas basadas en las reseñas, características de la película y calificaciones de otros usuarios.

Para lograr que todo funcione en conjunto, el proyecto ha sido desarrollado en un único entorno de programación unificado: *notebook jupyter.* Este tiene como resultado final una celda en donde se proponen una serie de preguntas al usuario para que este sistema, a través del algoritmo de recomendación, le da al usuario una recomendación de una película basada según lo que haya contestado.

ÍNDICE

DESCRIPCIÓN DEL SISTEMA

El SRPP se centra en el scraping de datos desde la plataforma IMDb para obtener información detallada sobre películas, incluyendo título, género, director, actores principales, calificaciones y reseñas. Estos datos se almacenan en una base de datos no relacional MongoDB, utilizando documentos JSON.

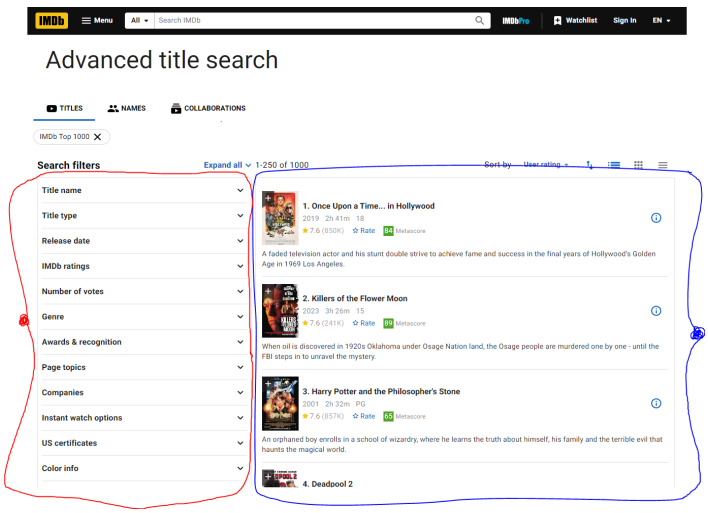
Posteriormente se generará un acceso a dicha base de datos para obtener los documentos almacenados de la colección y el algoritmo de esta manera poder acceder a la información. Este algoritmo procesa, de manera interactiva mediante una serie de preguntas realizadas directamente al usuario, los datos almacenados para así poder filtrar aquellas películas que no sean de interés al usuario dejando así las más relevantes. El objetivo de dicho filtrado es reducir los documentos de la base de datos según las especificaciones del usuario en sus respuestas para así poder generar una respuesta basada en sus intereses.

En última instancia, a la lista finalista se le aplicará una función aleatoria para que esta seleccione la película a mostrar como la recomendada y así evitar que en futuras peticiones idénticas se muestre siempre un mismo resultado.

SCRAPING Y IMDb WEB PAGE OVERVIEW

El scraping es una técnica común para la extracción de datos que implica recopilar información de páginas web de manera automatizada. En este caso el scraping ha sido diseñado con el fin de garantizar una información actualizada y customizada.

¿Por qué nuestro scraping es customizable? Para responder a esta pregunta primeramente es necesario conocer en qué formato se presenta la página web a la hora de la busqueda de informacion sobre peliculas:



Como podemos observar tenemos dos paneles interactivos. El señalado de color azul es meramente informativo y muestra un listado de las peliculas de donde extraemos la informacion. El panel rojo es el que nos permite personalizar nuestra búsqueda de películas a través de variadas opciones. Cada opción que es seleccionada genera un enlace.

El scraping ha sido desarrollado de manera que tome una lista de enlaces y así los vaya recorriendo y extrayendo la información necesaria. Estos enlaces se pueden modificar a gusto del usuario para que así la información recopilada sobre las películas sea de un tipo específico como industria de terror o cine para la familia, así como ampliar los datos o seleccionar otros grupos de información de la página web como los más populares o las películas más recientes.

Esta posibilidad permite al usuario poder modificar la información que se colecciona en la base de datos para así poder adecuar más todavía las recomendaciones. El seleccionado por el grupo de trabajo ha sido el grupo de películas IMDb top 1000 de las cuales han sido recopiladas 829.

La estructura y funcionamiento del scraping incorpora una serie de filtrados utilizando patrones regex para poder adecuar la información así como poderla interpretar correctamente. Algunas de ellas son *convertir_a_minutos* o *convertir_a_numero* que permiten convertir la duración de la película a minutos en vez del formato abreviado de horas y minutos, así como tomar la popularidad que se muestra abreviada en millones o miles y convertirlo a un número sin abreviación para una mayor facilidad a la hora de generar un filtrado y lectura en el algoritmo de recomendación.

2h 41m → 161
(850K) → 850000

Una vez nuestra información está recopilada y filtrada, esta será almacenada en un archivo JSON para su posterior incorporación a la base de datos MongoDB.

```
for url in urls:
    contador += 1
    headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 1
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.content, 'html.parser

    #storing the meaningfull required data in the varia
    movie_data = soup.findAll('li', attrs= {'class': 'i

    contadorkk = 0

    for store in movie_data:
            movie = {}

            name = store.div.h3.text
            name = quitar_numero(name)
            movie['name'] = name

            reference = store.find('a', class_ = "ipc-t
            href = reference.get('href')

            year_of_release = store.findAll('span', cla
            movie['year'] = year_of_release
```

La parte más importante del scraping digno de mencionar es el sistema de dobles bucles desarrollado. El primer bucle recorre las URLs de una lista y este realiza el acceso mediante `requests.get`. La biblioteca `BeutifulSoup` nos permitirá a través de `response.content` obtener en formato HTML todo el contenido de la URL. Serán extraídos todos los objetos de la página (estos serían todas las películas) y serán almacenados en `movie_data`. Esta variable es la recorrida en el segundo bucle donde cada store es cada película individual de la cual se extrae la información necesaria para el JSON final.

INCORPORACIÓN DE LOS DATOS A LA BBDDNR

A la hora de incorporar los datos recolectados estos se almacenan en una base de datos MongoDB bajo el nombre de “SGRP_BDNR”, en la colección “movies”. El programa accede a esta colección, junto con la incorporación de los datos mediante un enlace conteniendo el usuario y la contraseña de la base de datos no relacional de MongoDB.

Los datos resultantes del scraping se muestran en formato JSON en donde cada documento (pelicula) contiene la siguiente información:


```
{
    "name": " Killers of the Flower Moon",
    "year": "2023",
    "duration": "3h 26m",
    "ratings": "R",
    "calification": "7.6",
    "popularity": "232K",
    "metacritic_score": "89",
    "description": "When oil is discovered i
    "Actors": [
        "Leonardo DiCaprio",
        "Robert De Niro",
        "Lily Gladstone",
        "Jesse Plemons",
        "Tantoo Cardinal",
        "John Lithgow",
        "Brendan Fraser",
        "Cara Jade Myers",
        "Janae Collins",
        "Jillian Dion",
        "Jason Isbell",
        "William Belleau",
        "Louis Cancelmi",
        "Scott Shepherd",
        "Everett Waller",
        "Talee Redcorn",
        "Yancey Red Corn",
        "Tatanka Means"
    ],
    "Director": "Martin Scorsese",
    "Genres": [
        "Crime",
        "Drama",
        "History"
    ],
    "Reviews": [
        "The good: 2 brilliant actors, Leona
        "\"Killers of the Flower Moon\" is a
        "Leonardo Di Caprio returns from Wor
        "Some films warrant long runtimes. E
        "Martin Scorsese follows up his slop
        "Oil is discovered under Osage Natio
        "Obviously this isn't bad. It's from
        "I'm not a die-hard Martin Scorsese
        "Greetings again from the darkness.
        "Well that was a real snorer.Scorses
    ]
},
```


- name: nombre de la película
- year: año de lanzamiento
- duration: duración de la película
- ratings: restricción de edad recomendada
- calification: calificación de la película en la página web IMDb
- popularity: popularidad de la película
- metatritic_score: puntuación dada en la página metacritic
- description: sinopsis
- actors: actores principales y secundarios
- director: director
- genres: géneros principales
- reviews: 10 opiniones aleatorias escritas en la página IMDb.

Estos datos son filtrados antes de almacenarse (como observamos en la figura 1) con el fin de comprobar si son existentes o no para así determinar si deben de ser actualizados o insertados. Esta comprobación es necesaria puesto que MongoDB no tiene la capacidad de distinguir si los datos ya son existentes o no.

```python
for movie in movies_data:
    # Busca si ya existe la película
    existing_movie = collection.find_one({"name": movie["name"]})
    if existing_movie:
        # Si existe, actualiza los datos
        collection.update_one({"_id": existing_movie["_id"]}, {"$set": movie})
        print(f"La película{movie['name']} se ha actualizado en la base de datos
    else:
        # Si no existe, la inserta como nueva
        collection.insert_one(movie)
        print(f"La película{movie['name']} se ha añadido a la base de datos.")
```

Figura 1:Como observamos este accede a la colección de MongoDB, obtiene su nombre y define según `update_one` o `insert_one` si lo actualiza o lo inserta.

ALGORITMO DE RECOMENDACIÓN

Una vez los datos se encuentren almacenados en MongoDB estos serán consultados por un algoritmo el cual seguirá dos fases de preguntas:

- **FASE 1: APRENDIZAJE DEL USUARIO**

En esta faceta se intenta a través de una serie de preguntas personas conocer el usuario en detalle y predecir ciertos gustos estereotipados. Este aplica ciertos criterios genéricos que no son en todos los casos acertados pero permiten minimizar las opciones de la base de datos.

¿Qué edad tiene?
(Introducir edad)

Es la primera pregunta la cual marca bastantes aspectos en el algoritmo. Una persona adolescente tiende a consumir películas de menor duración, cierta popularidad y más acordes con la época actual, mientras que una persona de 40 años es más propensa a ver películas de mayor duración, de buenas críticas y no le importa que la película ya tenga una antigüedad considerable.

Es importante destacar que ha sido minuciosamente creado un filtro con ciertas restricciones de edad para los menores de 18 años y un filtro muy restrictivo para menores de 13 años puesto que trata de un público muy sensible. Este último filtro cuenta con restricciones sobre géneros de

películas, con el fin de evitar películas de tensión, horror, turbias o de contenido sensible como guerras o desnudez. El contenido que se mostrará a los menores de 13 años será únicamente infantil.

¿Cuál es su género?
(Masculino, Femenino)

Un estudio realizado por AMC Networks International (una de las mayores productoras de canales telemáticos de cine) explica cómo sí existen diferencias entre hombres y mujeres en cuanto a preferencias de cine. Un algoritmo de recomendación busca ser discriminatorio, así que teniendo en cuenta que según el género de la persona reducimos el 50% de las opciones de la base de datos, significa una gran ayuda para un algoritmo de recomendación. *

** Este algoritmo estereotipa según los datos del estudio. Hombre: Sci-Fi & Action. Mujer: Romance, Comedia y Thriller*

¿Cómo de fanático al mundo cinematográfico se considera?
(Experto, Aficionado, Ocasional, Muy ocasional)

Aquí se busca conocer cómo de exquisita es la persona que esté utilizando el algoritmo para recomendar una u otra película. Una persona experta en cine busca películas de alto nivel crítico, con buenas reseñas, pero tampoco busca las clásicas de siempre puesto que ya las conoce. Para evitar esto se filtran las películas con un nivel de popularidad relativamente bajo. Una persona ocasional en el mundo del cine busca más bien películas clásicas del mundillo que muy probablemente no conozca así como películas más actuales, dejando de lado la crítica.

¿Cómo le gusta la idea de pasar una tarde entera viendo películas?
(Plan fascinante, Agobiante, Irrelevante)

Esta pregunta es bastante directa en cuanto a conocer si el usuario estaría dispuesto a ver películas de larga duración o baja duración. Son consideradas películas de baja duración aquellas que tienen menos de 2 horas de duración y larga duración aquellas de más de 2 horas.

- **FASE 2: SEGMENTACIÓN DEL USUARIO**

En esta segunda faceta se busca conocer y profundizar las preferencias del usuario. Las preguntas a realizar son directas y claras en donde el usuario debe expresar sus gustos personales en cuanto a cinematografía se refiere, así permitiendo canalizar y minimizar drásticamente la búsqueda realizada por el algoritmo de películas en la fase de aprendizaje.

Seleccione aquellos géneros de su preferencia:
(Seleccionar géneros de películas)

Con esta pregunta el algoritmo pretende escoger únicamente aquellas películas que son de los géneros seleccionados por el usuario. El programa permite seleccionar cuantos géneros se quiera bajo una clara advertencia de que menos que 3 géneros podría no devolver una película. Esto es debido a que dependiendo de las exigencias del usuario ciertas combinaciones muy reducidas y extrañas podrían no llegar a cumplir las exigencias del algoritmo. Esto podría ser solucionado fácilmente añadiendo más enlaces en la lista de urls mencionada en *Scraping y IMDb web page overview.* *

** Esta opción no aplica a menores de 13 años.*

¿Le gustan las películas +18 años o aptas para todas las edades?
(+18 únicamente, Apto para todas las edades, Todo)

Aquí se pretende observar cual es la principal preferencia del usuario, películas de público adulto únicamente, en donde aparece violencia, desnudez, eventos politicos, lenguaje soez, etc o donde el público prefiera películas para ver en familia de limitadas características. *

** esta opción no aplica a menores de 18 años, se mostrará una advertencia.*

¿Prefiere películas populares o aquellas menos conocidas?
(Prefiero las más populares, Prefiero descubrirlas por mi mismo)

Cada documento de la colección como bien mostramos en *Incorporación de los datos a la BBDDNR* tiene una variable de popularidad. El algoritmo discrimina aquellas películas que estén en menos de 300k de popularidad como resultado de una búsqueda popular y elimina aquellas que estén por encima de 300k como resultado de poco populares. *

** Esta opción no aplica a menores de 13 años.*

¿Prefiere películas antiguas o de estreno?
(Antiguas, Estreno, Indiferente)

Con esta pregunta se canalizará el gusto en cuanto a antigüedad de película del usuario. Hay personas que prefieren únicamente películas antiguas y otras que prefieren películas de estreno. Es importante recalcar que no es lo mismo una película antigua para una persona de 18 años que para una de 60 años por lo que el algoritmo tiene esto en cuenta y cuando se marca la opción de película antigua según la edad de la persona se considera un valor u otro.

Cabe destacar que el orden de las preguntas es muy importante puesto que según se responden las preguntas ciertos estereotipos son eliminados y se prevalecen las preferencias del usuario con el fin de otorgar una respuesta de calidad y no genérica. Póngase el ejemplo de la edad, puede suceder que una persona joven que tiene un filtro estereotipado de no mayor duración de 120 minutos, se encuentre dispuesto a ver una película de mayor duración. Más adelante, este tendrá la opción de especificar de manera que se elimine ese estereotipo y se muestran películas de larga duración como el usuario deseaba.

Una vez haya sido generada lista de películas de recomendación, es aplicada una función de selección aleatoria para escoger una película aleatoria de dicha lista con el fin de evitar obtener una única respuesta a consultas idénticas. Asimismo, cada película recomendada se guarda en una lista la cual se comprueba antes de mostrar otra recomendación con el fin de que nunca aparezca el mismo resultado dos veces:

```
resultados = collection.find(consulta)

resultados_lista = list(resultados)

if resultados_lista:
    peliculas_disponibles = [pelicula for pelicula in resultados_lista if pe
    if peliculas_disponibles:
        pelicula_aleatoria = random.choice(peliculas_disponibles)
        print("Película recomendada:")
        print("- ", pelicula_aleatoria["name"])
        peliculas_mostradas.append(pelicula_aleatoria["name"])
        display(button)
    else:
        print("Ya no hay más películas disponibles que coincidan con los cri
else:
    print("No se encontraron películas que coincidan con los criterios de bú
```

Se muestra como la consulta se realiza a la colección de la base de datos. Las películas consultadas por el algoritmo se almacenan en `resultados_lista` *y posteriormente se comprueban que no hayan sido mostradas. En caso de no serlo, formarán parte de* `peliculas_disponibles`*. Si ya han sido mostradas se pasa a la siguiente película*

ESTADO DEL ARTE

Hasta la fecha, se ha completado la implementación del scraping de IMDb utilizando Python, con el uso de bibliotecas como `requests` y `BeautifulSoup` para manejar las solicitudes web y analizar el HTML, respectivamente. Se han superado desafíos como garantizar la precisión y consistencia de los datos recopilados, así como abordar la completitud de los mismos. Se ha modificado el enfoque del scraping para lograr que este no obtenga únicamente un enlace y su información relativa, sino que consten de varios para poder lograr una mayor recopilación de información.

También ha sido desarrollada correctamente la incorporación de los datos a la base de datos no relacional, superando todos los inconvenientes expuestos en la sección de *Problemas encontrados y sus soluciones.*

Nuestro actual algoritmo de recomendación es capaz de clasificar y filtrar las películas de la base de datos y reducir la lista final. Asimismo se proporciona una interfaz sencilla y cómoda al usuario para una sencilla interacción.

ANÁLISIS Y SELECCIÓN DE HERRAMIENTAS

Para el almacenamiento de datos, se ha seleccionado MongoDB debido a su capacidad para manejar documentos JSON y su compatibilidad con la herramienta de visualización MongoDB Charts. Se ha enfrentado el desafío de conexión con la base de datos MongoDB, optando por la solución de Atlas para una conexión más profesional y compatible a diferencia de MongoDB Compass, la cual únicamente ha sido utilizada con el fin de visualizar la base de datos.

Para el desarrollo del scraping y el algoritmo de recomendación ha sido utilizado Python, una herramienta rápida y potente bajo el entorno de ejecución de Google Collab, para una ejecución eficiente y compartir el código en tiempo real entre los integrantes del grupo.

DATOS USADOS, FUENTE Y PREPROCESO APLICADO

Los datos utilizados provienen de la plataforma IMDb, específicamente de la agrupación de IMDb top 1000. El motivo por el que decidimos conjuntamente escoger dicha plataforma es debido a que tiene una base estable y una fiabilidad óptima para la realización del trabajo, además de que los datos proporcionados por

la página web HTML se mostraban organizados y en el formato más óptimo acorde con los conocimientos del grupo.

La fuente lleva activa y bajo mantenimientos constantes desde el año 1990, lo que la hace una de las fuentes más accesibles, amplias y seguras de todo internet para visualizar reseñas e información acerca de las películas. A día de hoy se encuentra bajo la propiedad de Amazon, misma empresa que también opera una de las plataformas de streaming de series y películas más grande del panorama actual, lo que nos asegura que se mantenga al día y en constante actualización.

Se han aplicado técnicas de preprocesamiento, como el uso de expresiones regulares para limpiar y estructurar la información obtenida, así como filtros y uso de funciones generales de librerías de Python como `BeautifulSoup` y `Pandas`. Durante el preprocesado de datos también han sido utilizados bucles de recorrido de colecciones, así como cláusulas `try` y `except` para prevenir que la aplicación se detenga en caso de que alguna información no prioritaria sea faltante.

```python
# funcion para aplicar regex al nombre de pelicula
def quitar_numero(nombre):
      return patron.sub('', nombre)

# funcion para acceder a metadatos
def convertir_a_enlace_de_reviews(enlace_original):
      match = re.search(r'([^/]+)/?$', enlace_original)
      if match:
          ultimo_slash = match.group(1)
          enlace_reviews = f"{enlace_original}{ultimo_slash}/reviews"
          return enlace_reviews
      else:
          return None
```

Se muestra como se utilizan diversas funciones regex con el fin de filtrar y estructurar la información. En la función `quitar_numero` se accede a la variable nombre la cual contiene el nombre de la película y este busca un espacio y copia lo siguiente con el fin de eliminar el formato del scraping de "1. Nombre película" (en rojo lo innecesario y en verde lo necesario). En la función `convertir_a_enlace_de_reviews` observamos cómo se busca encontrar una barra "/" que continúe de un query "?$" con el fin de modificar el enlace para poder acceder a diferentes contenedores que contienen los metadatos de cada película.

PROBLEMAS ENCONTRADOS Y SUS SOLUCIONES

- **LIMITANTE DE INFORMACIÓN**

Nos encontramos con una limitación de información pues cada búsqueda únicamente contenía 100 resultados desde el primer vistazo. Esto es un problema a la hora de formar una base de datos amplia con una mayor colección que únicamente 100 películas. Al ser demasiado complejo interaccionar de manera humana mediante un cursor por la página a través de Python, debido a el uso de varias librerías poco conocidas y el posible bloqueo de la página por detectar movimientos robóticos, fuimos capaces de encontrar una solución óptima.

La solución dada fue editar el enlace añadiendo un parámetro llamado "*count=250"* (máximo permitido). Este parámetro permite mostrar de un único vistazo 250 películas (150 más que antes).

Descubrimos también que otro parámetro que se podía seleccionar era el de "*sort".* Este permite al usuario mostrar un orden distinto de películas dándonos así dentro del grupo de IMDb top 1000 otras distintas 250 películas al orden por defecto que tiene la página. Esto nos permite poder seleccionar varios filtrados a través de enlaces y luego recorrer estos enlaces en el scraping para así poder conseguir exactamente 829 películas del grupo de 1000 películas. Este grupo podría eliminarse y seleccionar películas de la plataforma en general pero hemos preferido mantenerlo así, pues es un número lo suficientemente amplio, además de que el grupo de IMDb top 1000 únicamente contiene películas con puntuaciones de calidad e información no faltante.

- **CONEXIÓN CON MONGODB**

Un problema bastante tedioso al que nos tuvimos que enfrentar fue a la creación y conexión con la base de datos MongoDB. La aplicación en Python no era capaz de lograr una conexión con la base de datos y esto sucedía por dos motivos: el cierre de puertos que hace la red de la universidad, pues gran parte del trabajo fue desarrollado en las aulas. El segundo motivo fue que utilizábamos MongoDB compass para la creación y gestión de la base de datos en vez del servicio de Atlas que ofrece MongoDB como solución para mapear objetos de manera directa en un entorno de programación.

Al utilizar compass que trata de una aplicación de visualización principalmente está al generar una conexión espera que el usuario introduzca un usuario y contraseña de manera interactiva. Esto genera un problema a la hora de conectar y automatizar procesos de guardado y lectura de MongoDB mediante código de Python, por lo que esta opción no era viable, además de que compass no permite crear contraseñas en bases de datos.

Finalmente, surgió un ligero problema a la hora de introducir los datos en MongoDB pues cada elemento cada vez que se ejecutaba el trozo de código que mete la información a la base de datos si ya esta se duplica. Este fue solucionado leyendo los títulos de las películas antes de introducirlos en la base de datos asegurándonos de que estas no estén ya dentro de la misma, en cuyo caso serían actualizadas.

Respecto a la problemática de la conexión con MongoDB la solución al problema fue la planteada en una consulta al profesor: crear un usuario y contraseña de login y proporcionarse a través del enlace de conexión, de manera que el programa no espere ningún tipo de respuesta por el usuario, sino que en la misma conexión ya se especifica toda la información. En complementación con esta solución también se incluye el utilizar una red privada fuera del recinto de la universidad, pues estas redes no tienen ningún tipo de restricción de conexión mediante puertos redireccionados.

CONSULTAS REALIZADAS PARA CUMPLIR OBJETIVOS

El programa comienza realizando consultas a la lista de urls;

```
urls = ['https://www.imdb.com/search/title/?groups=top_1000&count=250&sort=user_rating,asc',
        'https://www.imdb.com/search/title/?groups=top_1000&count=250&sort=runtime,asc',
        'https://www.imdb.com/search/title/?groups=top_1000&count=250&sort=year,desc',
        'https://www.imdb.com/search/title/?groups=top_1000&count=250&sort=boxoffice_gross_us,desc',
        'https://www.imdb.com/search/title/?groups=top_1000&count=250',
        'https://www.imdb.com/search/title/?groups=top_1000&count=250&sort=moviemeter,desc']
```

la cual contiene diferentes enlaces a scrapear explicados en la sección de *Scraping y IMDb web page overview.*

Las consultas son realizadas a través de la función de `requests.get(url, headers=headers)` para lograr acceder a la url y su ruta especificada en el header. Esta información del header se almacena en la variable `response` cuya se accede mediante `BeautifulSoup(response.content, 'html.parser')` en donde tomará la ruta almacenada en response y leerá la información en formato html. Esta se almacena en la variable `soup`.

La información de las diferentes películas de la página será accedida mediante `soup.findAll()`, la cual se almacenará en la variable `movie_data` y resultará en una lista de todas las películas encontradas en el html previamente extraído. Esta lista se recorre accediendo así a cada película individual.

Para cada película en cuestión se extraerán ciertos datos relevantes para el proyecto explicados en el apartado de *Incorporación de los datos a la BBDDNR.* La

manera en la que se extrae cada metadato es mediante las funciones de `find` y `findAll`. Toda la información es almacenada en la variable `json_data` a través de la función `json.dumps` para posteriormente guardarlo en el archivo movies_data.json de la siguiente manera:

```
json_data = json.dumps(movies, indent=4)

with open('movies_data.json', 'w') as f:
    f.write(json_data)
```

A la hora de introducir la informacion a la base de datos se realizaron las siguientes consultas:

```
client = MongoClient('mongodb+srv://aris:1234@cluster0.5ss2jng.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0')

db = client['SGRP_BDNR']
collection = db['movies']

# Lee datos del json
with open('movies_data.json') as f:
    movies_data = json.load(f)
```

En donde a través de `MongoClient` se accede a la base de datos utilizando el sistema de identificación mediante enlace con el usuario aris y contraseña 1234. Se accede al cluster0 y a través de la función `client['SGRP_BDNR']` se accede a la base de datos almacenada bajo la variable `db`.

Para acceder a la colección en donde vamos a almacenar toda la información accederemos a la variable que contiene la base de datos `db` y especificamos mediante corchetes la colección de *"movies"*.

A su misma vez se leen los datos del json creado previamente para poder iterar cada objeto y almacenarlo en la colección utilizando las funciones:

- `collection.update_one` para actualizar una película existente.
- `collection.insert_one` para introducir una película nueva.

A la hora de desarrollar el algoritmo fueron realizadas varias consultas a diferentes funciones de diversas librerías como `ipywidgets, IPython.display, pymongo, sys y random.`

Cada librería aporta al algoritmo varias funciones interesantes de las cuales destacaremos las principales:

- `pymongo:` aporta el acceso a la base de datos de la misma manera que lo hace en el guardado de datos previamente explicado.

- `ipywidgets:` esta librería aporta todos los aspectos visuales como widgets, menús selectores y casillas de introducción de texto utilizados para la interfaz del entorno. Funciones como `widgets.Dropdown, widgets.ToggleButton, widgets.Button y widgets.Text` aportan diferentes widgets interactivos a la hora del usuario responder las preguntas del algoritmo.
- `IPython.display:` simplemente se utiliza para mostrar en el entorno de ejecución todos los widgets y demás información tipo caracteres.
- `sys:` es utilizada para poder recoger la información escrita por el usuario y procesarla en la función más importante desarrollada por nosotros `ejecutar_consulta()` la cual contiene el algoritmo entero.
- `random:` es utilizada para la selección aleatoria de películas de la lista resultado del algoritmo. Esta se complementa con la lista de `peliculas_disponibles` explicada en el final de la sección de *Algoritmo de recomendación .*

TRABAJO FUTURO

En cuanto a trabajo futuro sería una opción muy interesante el crear varias bases de datos según las preferencias de datos. El proyecto actual solamente considera una única base de datos conteniendo un registro de varias películas generales. En casos específicos donde el usuario especifique demasiado la recomendación podría ser considerable tomar ciertas bases de datos para poder resolver la recomendación.

Esto sería posible teniendo en cuenta la implementación dada de múltiples url en el scraping. El trabajo futuro podría ser el desarrollar un algoritmo que clasifique qué tipo de respuestas está dando el usuario para poder alternar entre múltiples bases de datos y así lograr abarcar un mayor rango posible de películas.

Un ejemplo práctico sería el de recomendación de películas para niños de 6 años. Nuestro proyecto actualmente recomienda una película tipo infantil, familiar o de animación sobre la base de datos. El problema reside en si esa recomendación tiene especificaciones muy restrictivas, por ejemplo palabras clave o exigencia de popularidad o clasificación específicas, entonces el algoritmo podría no encontrar ninguna coincidencia. La opción de crear un algoritmo aparte para determinar que se está segmentando la recomendación a películas infantiles podría ser interesante que en ese entonces accediera únicamente a una base de datos con películas infantiles, esto con el fin de mejorar la respuesta.

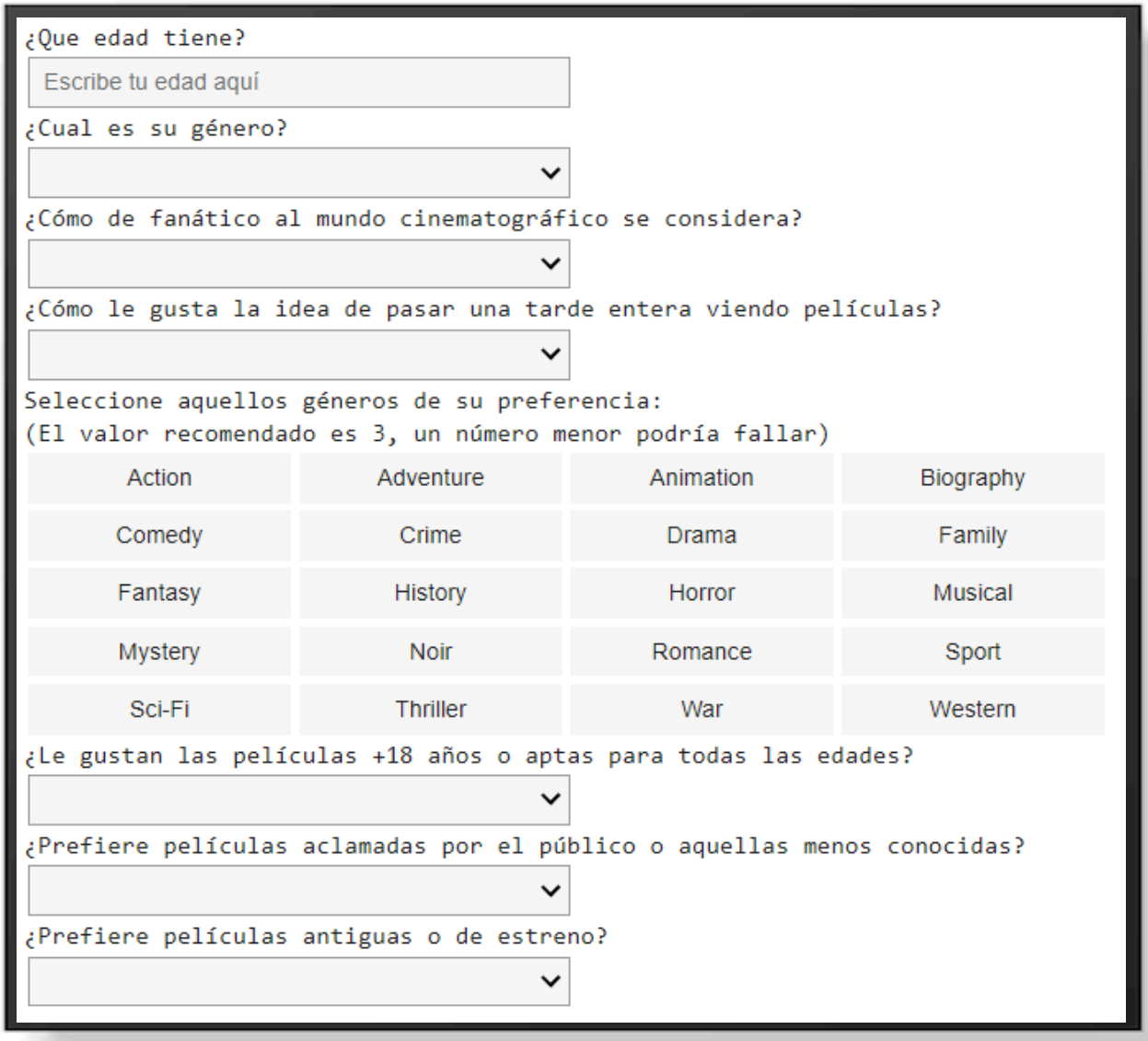
Otro trabajo futuro podría ser el implementar una interfaz de usuario cómoda a través de una aplicación para que los usuarios generales puedan acceder al sistema

de recomendación sin tener que pasar por un entorno de ejecución en Python. Esta interfaz de usuario podría ser en forma de web HTML a través de un host gratuito como *GitHub Pages* en donde a través de un simple enlace el usuario accediera a el sistema de recomendación. Otra opción más compleja sería el desarrollar una aplicación de ordenador o teléfono móvil.

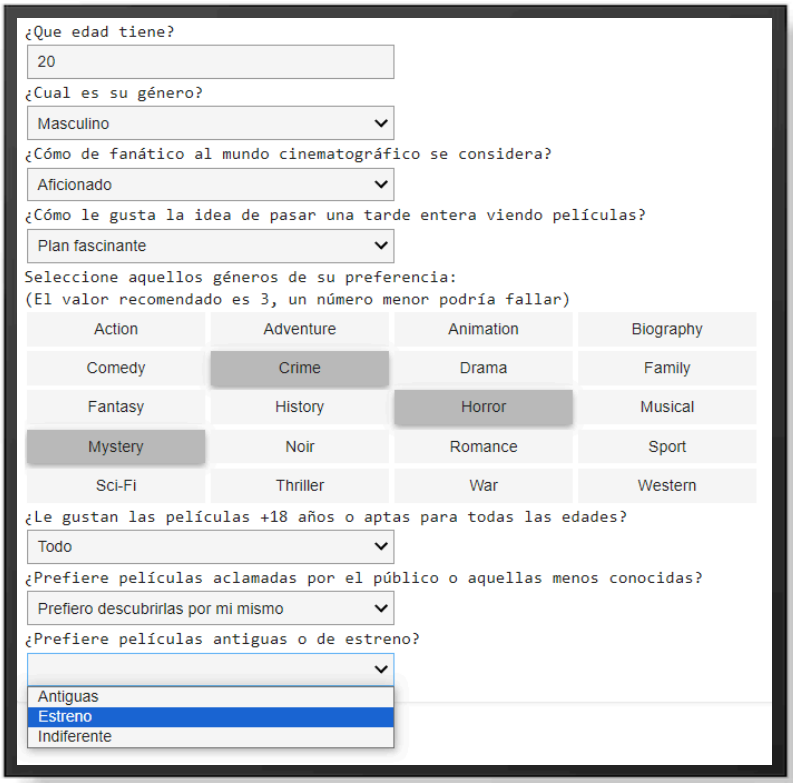
PRODUCTOS FINALES OBTENIDOS

El producto final principal ha resultado en un sistema de recomendación que en la gran mayoría de los casos devuelve más de una recomendación acorde con los gustos del usuario. Se puede interaccionar con este sistema a través de una interfaz simple que utiliza la libreria `ipywidgets` en el mismo entorno de ejecución. Esto lo hace conveniente pues es de sencilla ejecución.

Este se muestra de la siguiente manera:

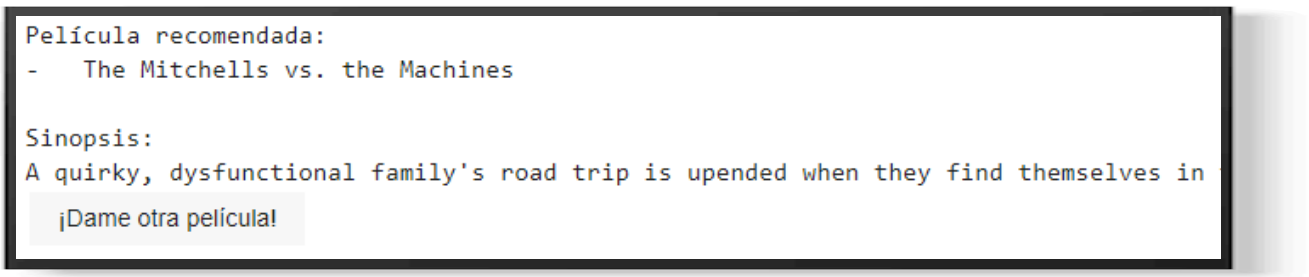


Como podemos observar este muestra las preguntas desarrolladas en el apartado *Algoritmo de recomendación* en donde se explica detalladamente cual es el fin de cada pregunta y cómo se interpreta internamente.



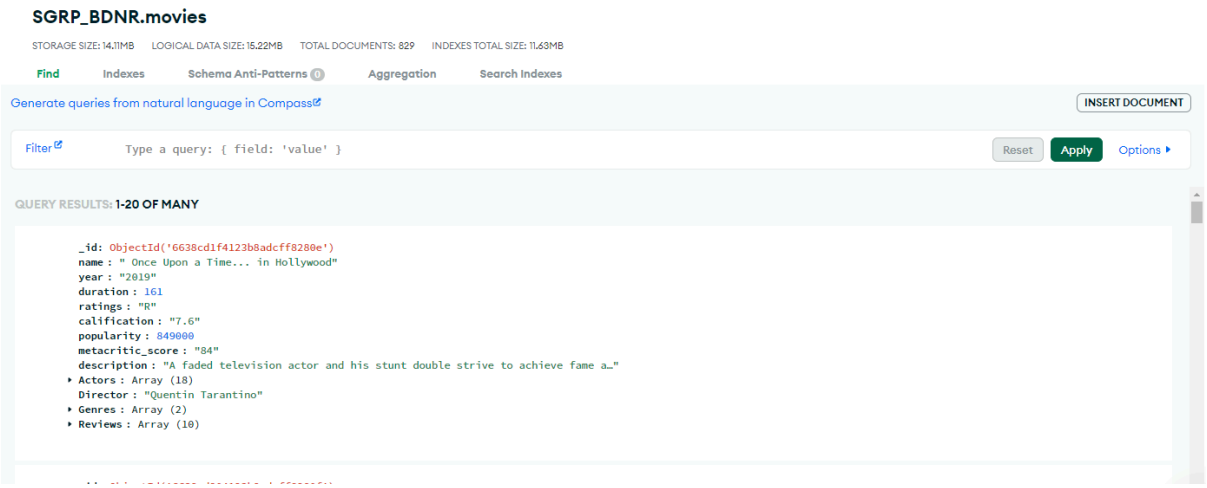
Aquí podemos observar un ejemplo de cómo se vería el formulario completo y también podemos observar como se muestra un desplegable al hacer clic, observando como se muestran las múltiples opciones preconfiguradas.

Una vez hayamos rellenado el formulario automaticamente sera enviado y procesado por el algoritmo y mostrará de la siguiente forma el resultado:



Vemos como muestra una película cumpliendo los requisitos dados por el usuario y proporcionando una recomendación acorde con su perfil. También el programa devuelve la sinopsis de la película para que rápidamente el usuario pueda descartar la película si no le interesa pulsando el botón de *“¡Dame otra película!”.*

Estos resultados son por parte del programa, por parte de la base de datos nos ha resultado una colección de 829 documentos:



Esta base de datos con estructura no relacional almacenada en el servicio de mongodb almacena cada documento y a través del casillero filter podemos realizar consultas de una forma rápida y sencilla.

CONCLUSIONES

El proyecto SRPP, ha demostrado ser una iniciativa exitosa para la recopilación, almacenamiento y análisis de datos de reseñas de películas de IMDb. El proceso de scraping de datos personalizado permite la extracción de información específica de la plataforma IMDb. Esta personalización proporciona flexibilidad para ajustar la búsqueda de películas según criterios que defina el usuario a través de la página web.

Utilizar MongoDB como base de datos no relacional para almacenar la información de las películas nos ha proporcionado un almacenamiento eficiente y flexible de los datos en formato JSON, facilitando su manipulación y consulta. También su servicio en nube permite a cualquier usuario sea donde sea poder ejecutar el programa sin instalaciones previas.

El algoritmo de recomendación tiene un funcionamiento bastante óptimo puesto que lo hemos sometido bajo varias pruebas demandantes con el fin de asegurar que responda correctamente. También ciertas personas han testeado el algoritmo con el fin de comprobar su calidad dando resultados muy óptimos. Esto nos sirvió también para obtener feedback, entre los cuales se encuentran recomendaciones como crear la interfaz de usuario interactiva más sencilla de lo planeado y así como introducir una sinopsis y un botón de siguiente película, los cuales no estaban proyectados en un principio.

Pese a todos los problemas a abordar desde un principio del proyecto como la limitación de información en el scraping, así como las conexiones con la base de datos MongoDB, fueron solventadas tanto con ayuda del profesor como entre los miembros del grupo.

Para cerrar la conclusión, SRPP ha sido un proyecto exitoso con unas puertas abiertas cara a mejoras, evoluciones y futuras versiones finales más avanzadas en donde no solo se podría extender la base de datos, sino que crear múltiples que almacenen información de usuarios como el uso de inteligencia artificial para cada uno de ellos.