universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Sistemas de Información de Gestión y Business Intelligence

Sistema híbrido de recomendación de películas empleando grafos de conocimiento

Autor: D. Álvaro González Jiménez

Tutor: D. Enrique López González

(Diciembre, 2020)

ÍNDICE

1. DESCRIPCIÓN DEL PROBLEMA

Tal vez si pensamos en un sistema de recomendación, el primero que se nos viene a la mente es un sistema de recomendación de películas. Todas y cada una de las plataformas de cine en streaming emplean un complejo sistema de recomendación encargado de analizar gigantescos volúmenes de datos y realizar sugerencias a sus usuarios en tiempo record. Aunque mucha gente lo desconozca, dichos sistemas no sólo tienen en cuenta los gustos del usuario para efectuar una recomendación, sino que analizan una gran cantidad de variables tales como el género, la edad, la raza, el dispositivo en el que sueles ver contenido, las horas del día en las que acostumbras a visualizar contenido o incluso el lugar de residencia con el fin de obtener recomendaciones lo más certeras posibles.

Estos sistemas, que se encuentran en constante desarrollo y evolución, son mucho más que simples sistemas de recomendación de películas. Si bien es posible que aunque utilices dichas plataformas con asiduidad no hayas reparado en ello, aunque estos sistemas recomienden la misma película o serie a dos personas diferentes, la portada mostrada no tiene por qué coincidir. Esto tiene una explicación muy sencilla, esos dos usuarios no son iguales, o al menos no lo son a ojos del sistema. Según la asunción que el sistema tenga del usuario, la portada que se le muestra difiere. Aunque desde luego este tipo de comportamientos no se pueden generalizar para toda la población, en la práctica ocurre que es infinitamente más probable que acabemos visualizando un contenido si la portada (que no deja de ser nuestra primera impresión sobre el contenido) es de nuestro agrado. Debido a esto, estas plataformas intentan que dicha primera impresión sea lo más fructífera posible, mostrando al usuario una portada que se asemeje al estilo de contenido que al usuario le guste visualizar, así como mostrando personajes que concuerden en raza y género con la asunción que el sistema tenga de esas dos características del usuario.

Tras esta breve descripción del estado del arte, supongo que lo más lógico es pensar ¿por qué iba a querer alguien desarrollar un sistema de recomendación de películas dado el extremo desarrollo y complejidad que caracteriza a los sistemas existentes en el mercado? Pues, aunque parezca increíble, tiene su respuesta. Vayamos por partes:

1. **Extensión del catálogo:** Como es evidente, los sistemas de recomendación de las grandes plataformas de streaming única y exclusivamente trabajan sobre el conjunto de películas que, en ese momento, están disponibles en la plataforma. Esto supone

una gran diferencia con mi aplicación, que consta con una base de datos con información y valoraciones de algo más de 9.000 películas, mientras que por ejemplo Netflix apenas cuenta con unas 2.500, Prime Video con unas 4.400 y HBO con unas 900.

2. **Antigüedad de las películas:** En las plataformas de streaming, las películas que más abundan son aquellas más modernas, que por lo general tienden a gustarle mucho más al gran grueso de la población. Sin embargo, precisamente por la necesidad de tener en el catálogo películas que agraden a la mayoría, quedan en el olvido un sinfín de clásicos como *Con la muerte en los talones* de Alfred Hitchcock, auténticos clásicos del séptimo arte discriminados únicamente por su antigüedad. En mi sistema, esta discriminación no existe. Pueden encontrarse películas desde el año 1903 hasta la actualidad, recomendándose todas ellas por igual, todo en base a ciertos criterios y reglas matemáticas que serán perfectamente descritas en capítulos posteriores de este mismo documento.
3. **Ausencia de intereses externos:** Con el paso de los años, las plataformas de streaming han dejado de ser simples intermediarias entre productoras cinematográficas y usuarios, pasando a convertirse cada vez con más frecuencia en productoras del propio contenido que ofrecen. Aquí se plantea un dilema, ¿acaso alguien considera que los sistemas de recomendación no van a valorar especialmente bien las películas que sus propias compañías han producido? Esas películas no se pueden ver en una sala de cine, tampoco se pueden adquirir en un centro comercial. Que tengan éxito o no depende solamente de que una gran cantidad de usuarios las visualicen, y ahí, los sistemas de recomendación tienen la misma importancia que una buena campaña de marketing. Mi sistema, por el contrario, no se ve influenciado por ningún tipo de vicio ni interés externo. Es, si se quiere ver así, totalmente puro e imparcial.

Habiendo ya explicado en qué se diferencia de los sistemas de recomendación convencionales, queda por explicar exactamente cómo funciona el sistema, qué problemas resuelve y, en definitiva, cuál es su razón de ser.

En su función más elemental, es un sistema de recomendación híbrido capaz de recomendar películas en base a dos aspectos totalmente diferentes que son aunados finalmente para efectuar una recomendación final. Estos dos aspectos son los siguientes:

- **Algoritmo de recomendación basado en contenido:** En base a las características que definen a las películas valoradas por el usuario, se efectúan recomendaciones de películas similares a las valoradas.
- **Algoritmo de recomendación de filtrado colaborativo:** En base a las valoraciones asignadas a las películas, se recomiendan al usuario las películas mejor valoradas por otros usuarios, en este caso aquellos que más se parecen a él.

Además de como sistema de recomendación, la aplicación también puede funcionar como una agenda en la que apuntar aquellas películas que hemos visto y la puntuación que les hemos asignado, aunque claramente se trata de una función secundaria.

Me gustaría resaltar que a la vez que se ha ido desarrollando este proyecto, se han ido actualizando un cuaderno de trabajo [1] y un repositorio de GitHub [2] cuya revisión puede resultar extremadamente interesante .

2. HERRAMIENTAS UTILIZADAS

En lo relativo a las tecnologías utilizadas, se pueden dividir en 3 grupos perfectamente diferenciados: base de datos, interfaz de usuario y servidor.

1. **BASE DE DATOS:** Entre las diversas tecnologías de bases de datos que pueden encontrarse en el mercado, hay dos que se llevan toda la atención: las bases de datos relacionales (MySQL, Oracle, MSSQL) y las bases de datos orientadas a documentos (MongoDB, CouchDB). Aunque sus usos son extremadamente variados (no en vano copan casi todo el mercado) para el problema que deseaba resolver no eran, ni en el mejor de los casos, lo más mínimamente viables.

   El motivo es sencillo, no son capaces de computar cálculos sobre el más de millón de relaciones de las que dispone la base de datos del sistema en un periodo de tiempo aceptable. Por ello, me decidí a utilizar una base de datos orientada a grafos de conocimiento, Neo4j para ser más exactos. Su estructura y funcionamiento, en el que las relaciones son el aspecto más importante, las convierten en la base de datos más apropiada para resolver este tipo de problemas, así como otros problemas relacionados con detección de fraudes o incluso investigaciones genéticas.

   Habiendo decidido el tipo de base de datos a utilizar, tan sólo quedaba una cosa, aprender a interactuar con ella. Neo4j implementa un lenguaje propio de consultas basado en SQL denominado Cypher, lo que hace que, si bien no sea necesario aprender la lógica que opera detrás de las consultas, sí que resulte extremadamente necesario aprender la sintaxis que emplea Cypher. Para ello, decidí completar el curso **Introduction to Neo4j 4.0** [3]**,** un curso de unas 16 horas de duración en el que pude aprender todas las funcionalidades de Cypher que posteriormente tuve que utilizar en el desarrollo del sistema, es decir, funcionamiento de una base de datos

orientada a grafos de conocimiento, creación de nodos y relaciones, consultas, indexación, importación de datos…

Sobre la base de datos hay muchos más aspectos que comentar, de modo que he decidido crear un capítulo que hable en exclusiva del origen del base de datos, procesamiento, importación, estructura, etc.

2. **INTERFAZ DE USUARIO (FRONTEND):** A la hora de diseñar la interfaz de usuario, que generalmente se denomina frontend, decidí utilizar las tecnologías que ya conocía previamente. Así, al no tener que aprender desde 0 ninguna tecnología, puede realizar el desarrollo de una forma mucho más eficiente, obteniendo además un mejor resultado. Las tecnologías en cuestión son las siguientes:

    1. **JavaScript** [4]**:** Lenguaje de programación que aporta interactividad dinámica a la página web.
    2. **Vue** [5]**:** Framework de JavaScript que permite la construcción de interfaces de usuario y aplicaciones de una única página.
    3. **Vuetify** [6]: Framework de Vue que permite la creación de interfaces de usuario basándose en componentes prediseñados.
    4. **Vue router** [7]: Librería encargada de gestionar los movimientos entre vistas en una aplicación creada con Vue.
    5. **Vuex** [8]: Librería encargada de gestionar los estados de una aplicación creada con Vue. Además, sirve como repositorio central de datos de la aplicación.
    6. **Vuex persistedstate** [9]: Librería encargada de persistir la información de Vuex al refrescar una página.
    7. **Axios** [10]: Cliente HTTP basado en promesas que permite efectuar peticiones API a un servidor.

3. **SERVIDOR (BACKEND):** A la hora de diseñar el servidor de la aplicación, generalmente denominado Backend, se me presentaron dos alternativas entre las que tuve que decidir. Por un lado, tenía la opción de desarrollarlo en nodejs [11], un entorno en tiempo de ejecución basado en JavaScript [4] que permite el desarrollo de servidores. Por otro lado, dado que es en el servidor donde se ejecutan los algoritmos

de recomendación, estaba la opción de desarrollarlo en Flask [12], un framework de Python [13] que permite el desarrollo de servidores de forma sencilla.

Tras valorar pros y contras de ambas implementaciones me decidí por la segunda opción, especialmente porque Python iba a ser muchísimo más eficiente que JavaScript a la hora de realizar operaciones de análisis y procesamiento de datos.

Como no había desarrollado jamás una aplicación con Flask, me vi en la necesidad de aprender desde 0 su funcionamiento, el cual aprendí mediante dos vías diferentes:

1. **Visualización de un vídeo en YouTube** [14]: En este vídeo se explica a la perfección el funcionamiento del framework. Peticiones, respuestas, gestión de rutas, argumentos, validación de entradas… Tan solo visualizando los 50 primeros minutos de este vídeo ya pude ponerme sin ningún problema a desarrollar el backend del proyecto.
2. **Observación del siguiente proyecto en GitHub** [15]: En este repositorio, dentro de la carpeta flask-api puede encontrarse un ejemplo de desarrollo de una aplicación Flask con acceso a base de datos neo4j que me sirvió de ejemplo para observar cómo se hacen las consultas a la base de datos desde Python.

Tras aprender el funcionamiento del framework, proceso que me llevó aproximadamente un día, pude empezar a desarrollar el backend del proyecto sin ningún tipo de problema. Las tecnologías que empleé, además de por supuesto Python y Flask, fueron las siguientes:

1. **Pandas** [16]**:** Módulo de Python que sirve para la manipulación y análisis de datos.
2. **Numpy** [17]**:** Módulo de Python que da soporte a la creación de vectores y matrices, así como a la realización de una gran cantidad de operaciones matemáticas complejas.
3. **Neo4j** [18]**:** Módulo de Python que permite la comunicación con bases de datos Neo4j.
4. **Sklearn** [19]**:** Módulo de Python que aporta una gran cantidad de funcionalidades relacionadas con Aprendizaje automático.

5. **Scipy** [20]**:** Módulo de Python que aporta una gran cantidad de herramientas y algoritmos matemáticos.

3. BASE DE DATOS UTILIZADA

A la hora de desarrollar un buen sistema de recomendación, resulta estrictamente necesario que la base de datos sobre la que se trabaje sea lo más completa posible. Por muy robusto y efectivo que pueda ser el sistema, si la calidad de los datos no es la adecuada, las recomendaciones en la práctica no serán casi nunca certeras. Debido a esto, he considerado oportuno crear un capítulo dedicado única y exclusivamente a la propia base de datos en la que se traten diversos temas como la selección, procesamiento y carga de los datos, así como de la estructura de la propia base.

1. SELECCIÓN DE LOS DATOS

Para seleccionar un conjunto de datos sobre el que trabajar, decidí buscar en la conocida comunidad de científicos de datos Kaggle [21]. Allí, pueden encontrarse una gran cantidad de dataset, cursos y competiciones de temáticas más que variadas. Entre los dataset de películas que encontré, debo destacar los siguientes:

1. **TMDB 5000 Movie Dataset** [22]: Aunque se trata de un dataset bastante completo con numeras películas y valoraciones, faltaban algunos campos en la base de datos que, en mi opinión, eran muy importantes, como por ejemplo el reparto de la película.
2. **The movies dataset** [23]**:** Si bien se trata de un dataset extremadamente completo que contiene una gran cantidad de información sobre cada película, valoraciones, etc. A la hora de explorar el conjunto de datos en profundidad me di cuenta de que todo estaba almacenado en formato json dentro de un archivo csv. En base a esto, y aunque el dataset en principio tenía bastante buena pinta, decidí seguir buscando.
3. **Movielens** [24]**:** El dataset de películas por excelencia sobre el que desarrollar un sistema de recomendación, que incluye millones de valoraciones de usuarios a decenas de miles de películas. El único problema que encontré al dataset es que no almacena información sobre el casting de las películas, lo que hace que, a la hora de mostrar

información en mi aplicación sobre las películas, no pueda ser esta muy completa.

4. **Extensión de Movielens** [25]**:** Cuando ya había decidido utilizar Movielens, encontré por casualidad este dataset, una ampliación de Movielens realizada por algunos investigadores de la Universidad Autónoma de Madrid. En este dataset, además de toda la información existente en el dataset de Movielens, podemos encontrar otra información adicional como el país de origen de la película, el reparto, director, etc. Por si esto fuera poco, todo el dataset ha pasado previamente un proceso de limpieza, de modo que no hay que realizar un proceso de limpieza de datos muy inferior a los demás.

Tras analizar las ventajas e inconvenientes de seleccionar cada uno de los diversos dataset, decidí utilizar el último de todos (la extensión de Movielens) dado que contiene una gran cantidad de películas y valoraciones y, además, no es necesario realizar un proceso de limpieza demasiado exhaustivo.

2. CONVERSIÓN DE LOS DATOS

Para importar datos a una base de datos Neo4j desde un archivo, es necesario que éste se encuentre en formato CSV. En mi caso, el conjunto de datos elegido se encontraba en varios archivos en formato DAT, en el que las diferentes columnas se encuentran separadas por tabulaciones en lugar de por comas. Debido a esto, me vi en la obligación de desarrollar un pequeño script en Python que se encargara de transformar los archivos al formato deseado.

Dentro de este pequeño script, además de cambiar las tabulaciones por comas, también se realizan otro tipo de transformaciones como, por ejemplo, reemplazar todos los caracteres especiales por sus equivalentes en el lenguaje inglés, eliminando por tanto las tildes a las vocales, sustituyendo la letra ñ por la n, etc.

Aunque no lo había comentado hasta el momento, todos los archivos que forman la base de datos, un total de 6, deben ser transformados mediante el script en Python

procesar_csv.py que puede encontrarse en el repositorio de GitHub en la carpeta Scripts.

3. IMPORTACIÓN DE LOS DATOS

Con el fin de importar todos los datos en la base de datos, y aunque es posible hacerlo en un único script, decidí crear un script por cada archivo, de modo que fuera más sencillo tanto depurar errores y, además, el tiempo de carga fuera menor.

Aunque la estructura de dichos scripts difiere en base a cuál sea el archivo que se quiere cargar, aquí puede encontrarse un ejemplo de la carga de todas las películas que contiene la base de datos:

```
:auto USING PERIODIC COMMIT 1000
LOAD CSV WITH HEADERS FROM 'file:///movies.csv' as line
CREATE (m:Movie{id:toInteger(line.id), imdbID:line.imdbID, title:line.title,
year:toInteger(line.year), picture:line.rtPictureURL})
```

Figura 3.1.- Importación datos películas

Al efectuar la carga, decidí importar el contenido de los archivos en su totalidad, de modo que la carga de las valoraciones puede llegar a demorarse unas cuantas horas. En el caso de querer realizar esta carga en un periodo de tiempo mucho menor, bastaría con añadir una limitación en el script de carga, indicando que solo se quieren cargar las x primeras líneas.

Los scripts de carga pueden encontrarse dentro del repositorio de GitHub en la carpeta Scripts. Al ejecutarlos, es importante incluirlos dentro de la carpeta Import en Neo4j, de no ser así, el gestor no sería capaz de encontrarlos y, por tanto, no se podría efectuar la carga de manera satisfactoria.

Muy relacionados con la carga de los datos está la indexación. En vistas a que el sistema de búsqueda implementado en la interfaz de usuario resulte más efectivo,

siempre es conveniente añadir una serie de índices que, bien implementados, son capaces de reducir los tiempos de carga de una manera increíble.

En mi caso, el índice principal y más importante es el siguiente, que permite indexar las películas en base a su título:

```
CREATE INDEX movie_titles_index FOR (m:Movie)
ON (m.title)
```

Figura 3.2.- Indexación películas en base a título

Al igual que todos los scripts, los archivos encargados de la creación de índices pueden encontrarse dentro del repositorio GitHub en la carpeta Scripts.

4. PROCESAMIENTO DE LOS DATOS

Habiendo importado todos los datos, llega el momento de efectuar una serie de transformaciones en los datos que nos ayuden a eliminar duplicados y a facilitar el trabajo a los algoritmos de recomendación que se ejecutarán en el backend.

Dentro de este procesamiento, destacan 3 tipos totalmente diferenciados:

a. **Eliminación nodos repetidos**: En vistas a una mayor robustez del sistema de recomendación, resulta necesario eliminar aquellos nodos que se encuentran repetidos en la base de datos. Aunque el dataset se encontraba ya bastante limpio en un principio, sí que se podían encontrar algunas inconsistencias que debían ser solventadas.

Entre los directores, el id asignado para su identificación se encontraba en algunas ocasiones repetido, de modo que existían dos o más nodos para un mismo director. Esto también ocurría con algunos actores y películas, que también tenían el mismo identificador o la misma portada en el caso de las películas.

Para eliminar estos nodos repetidos, decidí implementar un sencillo Script en Cypher. El siguiente ejemplo muestra como eliminar todas las películas con la misma portada

a excepción de 1. Sin embargo, el script es perfectamente extrapolable a cualquier otro tipo de nodo

```
MATCH (m:Movie)
WITH m.picture AS picture, COLLECT(m) AS repeated
WHERE SIZE(repeated) > 1
UNWIND repeated[1..] AS non_valid
detach delete non_valid
```

Figura 3.3.- Eliminar películas repetidas

b. **Eliminación nodos incompletos**: Al igual que con los nodos repetidos, los nodos incompletos también podían suponer un problema al sistema. Debido a esto, decidí eliminar todos aquellos nodos que carecieran de alguna propiedad, ya sirviera esta de identificación o fuera secundaria.
c. **Procesamiento de texto**: Aunque en un principio no había reparado en ello, al ir desarrollando el algoritmo de recomendación basado en contenido reparé en que, por la forma de separar tokens en Python, no podían existir guiones ni espacios en los nombres de las características que fueran a ser utilizados como criterios de recomendación. Debido a esto, decidí desarrollar un script en Cypher que lo solucionara. El siguiente ejemplo muestra como sustituir los guiones y espacios contenidos en el texto por barras bajas:

```
MATCH (c:Country)
WHERE c.name contains " " or c.name contains "-"
set c.name = replace(c.name, "-", "_")
set c.name = replace(c.name, " ", "_")
```

Figura 3.4.- Sustitución espacios y guiones por barra baja

5. ESTRUCTURA DE LA BASE DE DATOS

Para terminar el capítulo, considero oportuno explicar de forma detallada cuál es la estructura de la base de datos, es decir, por qué tipo de nodos y relaciones está formada.

Al haberse realizado una carga de la totalidad de los datos, el tamaño de la base es considerable, unos 110.000 nodos y 1.000.000 de relaciones. Dado que la gran

mayoría de relaciones son valoraciones de los usuarios a las películas, y como ya comenté anteriormente, en el caso de querer que los tiempos de respuesta de la base de datos sean menores y el tiempo de importación de los datos sea también menor, tan sólo es necesario establecer una limitación a la hora de cargar las valoraciones en las que se indique el número de éstas que se quieran cargar.

En total, la base de datos está formada por 6 nodos diferentes y 5 relaciones entre ellos, siendo el nodo central de la base el que contiene el título de las películas. El esquema de visualización de la base de datos sería el siguiente:

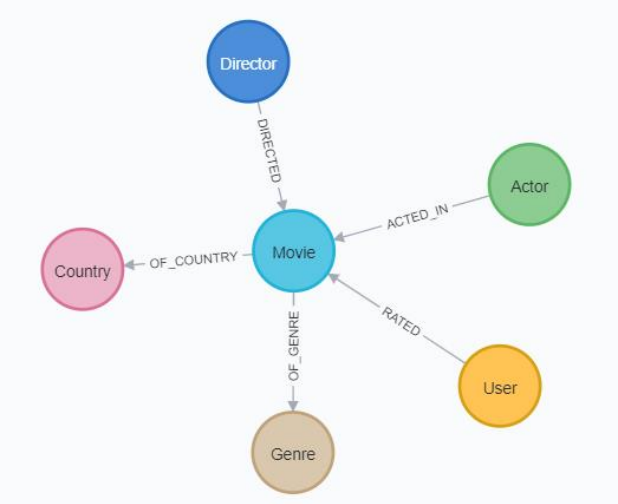

Figura 3.5.- Esquema visualización BBDD

A su vez, se podría definir cada uno de los nodos de la siguiente manera:

- **Movie**: Nodo que contiene la siguiente información sobre una película: identificador único, identificador de la película en IMDb, título, año, link a la fotografía de la portada. En la base hay un total de 9.060 películas diferentes.
- **Genre**: Género al que puede pertenecer una película. En la base se pueden encontrar un total de 20 géneros diferentes.
- **Country**: País de origen de una película. En la base hay un total de 71 países diferentes.
- **Actor**: Nodo que contiene la información de cada actor en la base, identificador único y nombre del actor. Existen un total de 95.320 actores diferentes.
- **Director:** Nodo que contiene la información de cada director en la base, identificador único y nombre del director. Existen un total de 4.054.

- **User:** Nodo que contiene la información de cada usuario. Para los usuarios generados a partir del csv, la única información disponible será el identificador. Sin embargo, para los usuarios generados a partir de la aplicación web mediante el sistema de registro, también se encontrará almacenado su nombre, username y su contraseña debidamente cifrada.

Una vez descritos de forma correcta los distintos nodos existentes en la base de datos, creo necesario definir de igual modo las relaciones existentes entre ellos.

- **OF_GENRE**: Relación entre Movie y Genre. Cada película puede pertenecer a un género o a varios. Existen un total de 18.366 relaciones de este tipo.
- **OF_COUNTRY**: Relación entre Movie y Country. Cada película solo puede estar relacionada con un único país de origen. Existen un total de 9.060 relaciones de este tipo.
- **ACTED_IN**: Relación entre Movie y Actor. Cada actor puede actuar en tantas películas como se desee. Existen un total de 207.985 relaciones de este tipo.
- **DIRECTED**: Relación entre Director y Movie. Cada director puede dirigir tantas películas como sea necesario y, además, cada película puede estar dirigida por más de un director. Existen un total de 9.060 relaciones de este tipo.
- **RATED:** Relación entre User y Movie. Cada User puede valorar tantas películas como quiera y cada película puede estar valorada por más de un User. La relación tiene un atributo que contiene la valoración que el usuario ha asignado a dicha película, un número entre 0 y 5. Existen un total de 758.836 relaciones de este tipo.

4. DESCRIPCIÓN DE LA APLICACIÓN

La aplicación es una aplicación web que, por tanto, opera bajo el paradigma cliente-servidor. Este tipo de aplicaciones están formadas por 3 partes esenciales: base de datos, servidor web y servidor de aplicación.

Aunque ya he explicado anteriormente las tecnologías a utilizar en cada una de estas 3 partes principales, creo que es necesario explicar de forma sencilla qué función cumple cada una, el tipo de relación que existe entre ellas, de qué forma se comunican, etc. Para ello, me gustaría resaltar el siguiente esquema que muestra de forma sencilla las interacciones que se producen entre esas diferentes partes.

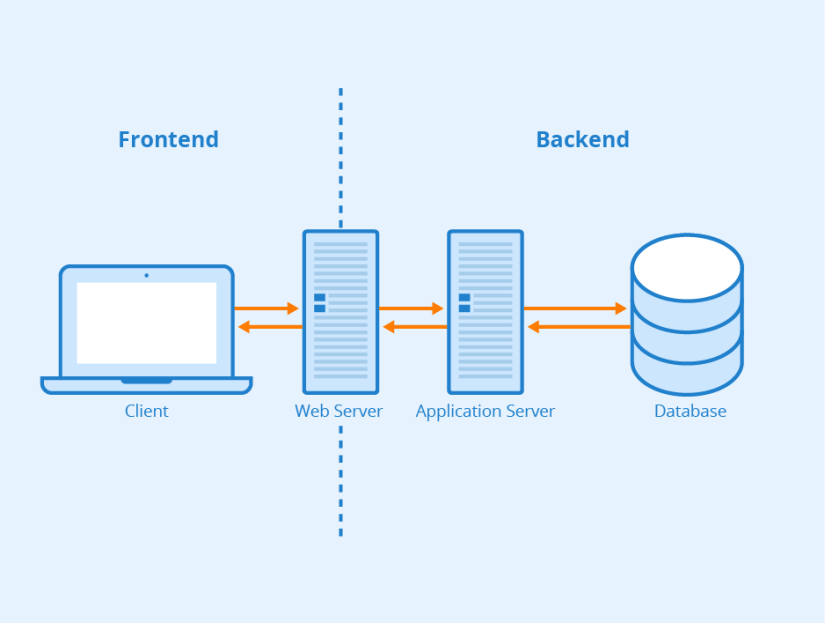

Figura 4.1.- Interacción diferentes partes

Para comenzar, debo explicar qué función tiene cada una de las partes.

- **Servidor web (frontend)**: Es el encargado de devolver el código de la aplicación al cliente. Se encarga por tanto de mostrar la interfaz de usuario al cliente y de realizar peticiones al servidor de aplicación. Este servidor no tiene acceso ni a la lógica de negocio de la aplicación ni, por supuesto, a la base de datos.
- **Servidor de aplicación (backend)**: Es el encargado de gestionar toda la lógica de negocio de la aplicación. Esto quiere decir que se encarga de gestionar los registros, inicios de sesión, calcular las recomendaciones y, por supuesto, de enviar al servidor

web toda la información que necesite mostrar al cliente. También se encarga de realizar consultas a la base de datos tanto de inserción como de búsqueda.

- **Base de datos**: Su principal función es, además de persistir toda la información necesaria para la aplicación, la inserción de nuevos datos y la devolución al servidor de aplicación de los datos que cumplen las condiciones indicadas en la consulta.

Con respecto a las interacciones entre las partes, como se puede observar en la figura 6.1, son las siguientes (no se indica la relación entre cliente y servidor web porque no es relevante para el funcionamiento de la aplicación):

- **Servidor web y servidor de aplicación**: Como en la mayoría de aplicaciones web, la conexión entre ambos servidores se realiza mediante una API REST. Utilizando este mecanismo, ambos servidores se comunican mediante peticiones http. El servidor de aplicación (backend) se encuentra permanente escuchando este tipo de peticiones. Al recibir una, dependiendo de la ruta a la que se haya efectuado y el método indicado, se realizan unas acciones u otras y se devuelve al servidor web un objeto json que contiene la información solicitada.
- Servidor de aplicación y base de datos: La conexión se realiza mediante un driver especial de Neo4j. Utilizando este driver, se puede abrir una conexión a la base de datos directamente desde el servidor de aplicación, solicitar la información necesaria y, al terminar el trámite requerido, cerrar la conexión.

Habiendo explicado ya las diferentes partes que forman la aplicación y cómo interactúan entre ellas, llega el momento de explicar con detenimiento el funcionamiento de la aplicación.

El punto de entrada de la aplicación, es decir, la primera pantalla que nos encontramos, es la pantalla de Inicio de sesión. Aquí, se presenta al usuario la posibilidad de iniciar sesión en la aplicación introduciendo su nombre de usuario y su contraseña. En el caso de que el usuario carezca de cuenta en la aplicación, también se le presenta la oportunidad de dirigirse a la pantalla de registro.

La pantalla de inicio de sesión tiene el siguiente aspecto:

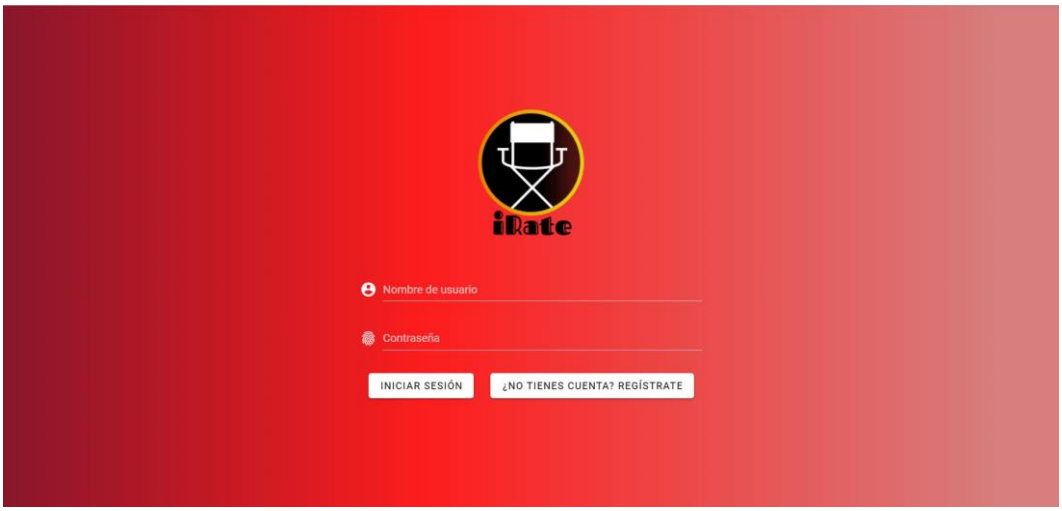

Figura 4.2.- Ventana de inicio de sesión

En la pantalla de registro, el usuario tiene la opción de crear una cuenta en la aplicación indicando su nombre, apellidos, nombre de usuario y contraseña. El nombre de usuario es único, lo que quiere decir que no puede existir dos usuarios en la base de datos con el mismo nombre de usuario. Por motivos de seguridad, la contraseña se almacena cifrada en la base de datos en lugar de en crudo.

Además de crear una cuenta, el usuario tiene también la opción de volver a la pantalla de inicio de sesión en caso de que ya haya creado una cuenta. La apariencia de la pantalla de registro es la siguiente:

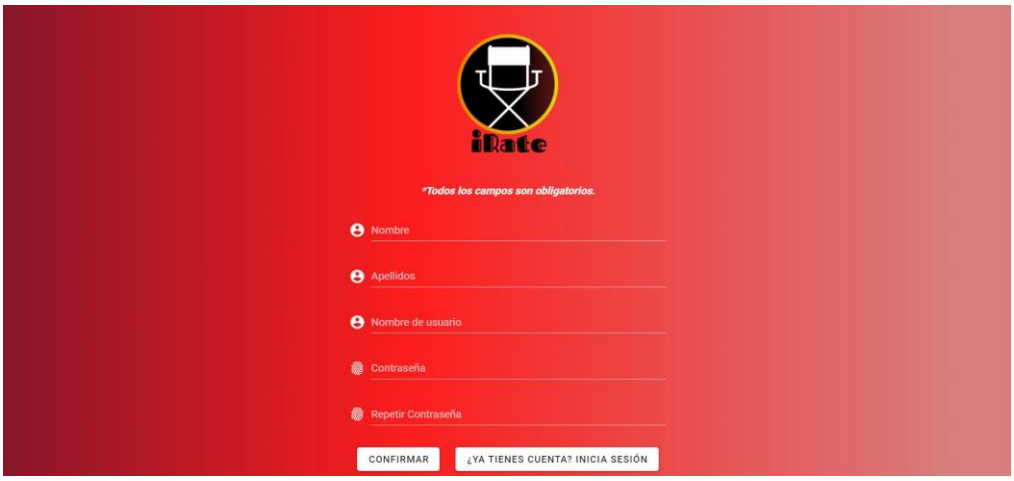

Figura 4.3.- Ventana de registro

Esta pantalla, como es lógico, tan sólo será utilizada por un usuario una vez para crear una cuenta. A partir de ese momento, tan sólo tendrá que iniciar sesión cuando quiera acceder a la aplicación.

Habiendo explicado ya estas dos ventanas, que como tal no aportan ninguna funcionalidad especial al sistema más de allá de la autenticación, llega el momento de explicar qué pasa cuando iniciamos sesión en el sistema.

La realidad es que al iniciar sesión en el sistema pueden pasar dos cosas diferentes dependiendo de si iniciamos sesión por primera vez o si, por el contrario, ya hemos utilizado la aplicación como mínimo en una ocasión.

En el que caso de que estemos iniciando sesión por primera vez, nada más acceder a la aplicación nos encontramos con una pantalla especial en la que se nos pide que valoremos al menos 5 películas que hayamos visionado. El motivo de esto es muy sencillo, para efectuar una recomendación por simple que sea, se necesita tener un mínimo de conocimiento sobre el usuario.

Aunque 5 pueda parecer un número excesivamente pequeño, en el capítulo 6 nos daremos cuenta de que es un número más que suficiente para realizar recomendaciones robustas.

Como es lógico, inicialmente no pueden mostrarse un número muy elevado de películas, podría resultar contraproducente y acabar agobiando al usuario. Lo primordial es que, independientemente del número de películas que se muestren, tengan temáticas variadas. Por ello, la decisión que tomé fue solicitar a la base las películas más valoradas de cada género y permutarlas de forma aleatoria.

No es exactamente un indicativo 100% fiable de la popularidad de una película, pero parece bastante lógico pensar que las películas con más valoraciones en la base de datos serán, por norma general, las películas más populares.

Si observamos la ventana, podemos observar que las películas mostradas son verdaderamente conocidas:

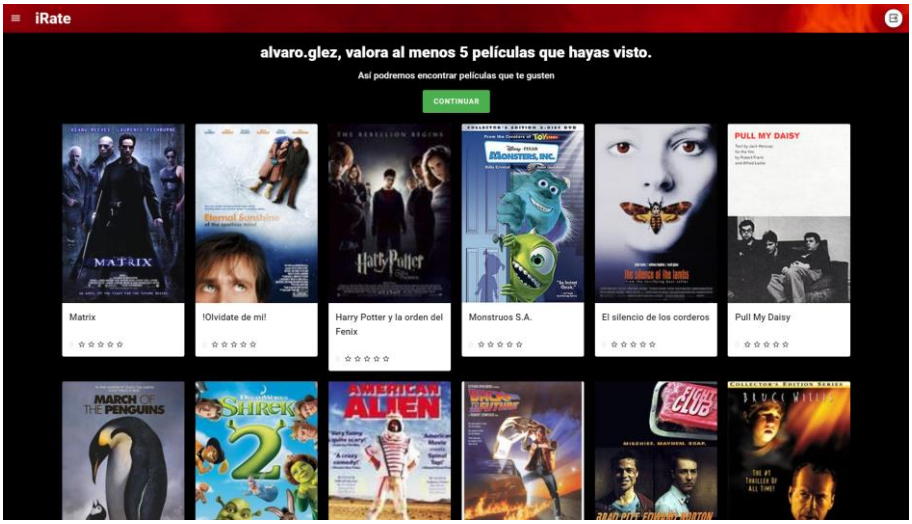

Figura 4.4.- Ventana valoraciones iniciales

Con respecto a las tarjetas que muestran las películas, en el caso concreto de esta ventana he decidido optar por un diseño bastante minimalista. De esta forma, creo que cobra mucha más importancia la fotografía y el título, que son los aspectos más relevantes a la hora de identificar una película. De añadir más información superflua como año de lanzamiento, director o géneros se podría distraer de forma involuntaria al usuario, dificultando por tanto que encuentre de forma exitosa las películas que haya visto en el pasado.

Cabe destacar cómo funciona el sistema de puntuaciones de la aplicación. Para realizar una valoración a una película, un usuario tan solo tiene que seleccionar, en la tarjeta de la película que quiere valorar, una puntuación entre 0.5 y 5 estrellas. Simplemente con eso, la valoración se almacena automáticamente en la base de datos. En el caso de que se quiera cambiar una valoración, tan solo es necesario cambiar la selección de estrellas en la tarjeta, de forma automática se actualiza en la base de datos. Si por ejemplo realizamos una valoración de 5 estrellas a la película Memento, el aspecto de la tarjeta es el siguiente.

Figura 4.5.- Ejemplo tarjeta

Ya para terminar con esta ventana que, como recordatorio, sólo se mostrará a los usuarios que todavía no hayan realizado ninguna valoración, cabe destacar que cuando hayamos valorado al menos 5 películas tan solo tendremos que pulsar en el botón de continuar, situado en la parte superior de la ventana, y acceder a la aplicación como tal. En caso de que no hayamos valorado todavía el número de filmes indicado, no ocurrirá nada, es condición necesaria y suficiente el haber valorado como mínimo 5 películas para acceder a la aplicación.

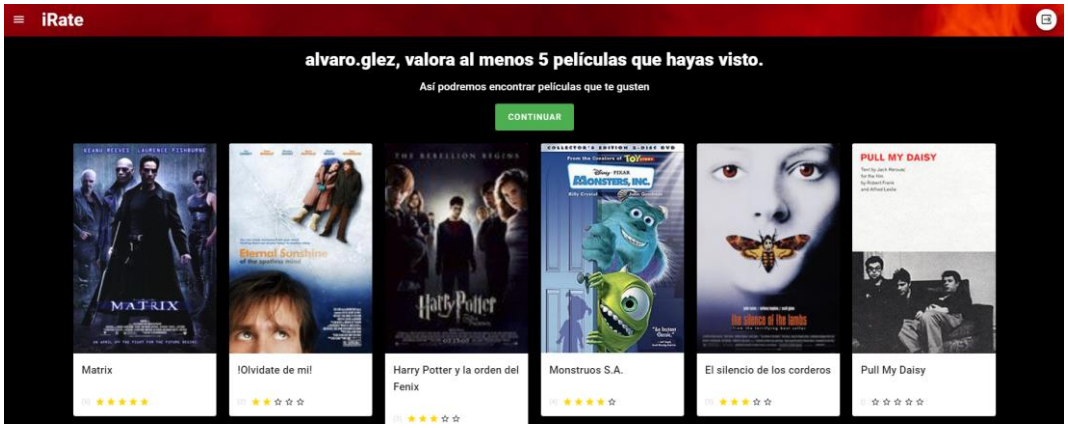

Figura 4.6.- Ejemplo condición acceso superada

Cuando accedamos a la aplicación, ya sea porque acabamos de valorar esas 5 películas iniciales o porque ya seamos usuarios habituales, nos encontraremos con la pantalla de Búsqueda de películas, que tiene el siguiente aspecto:

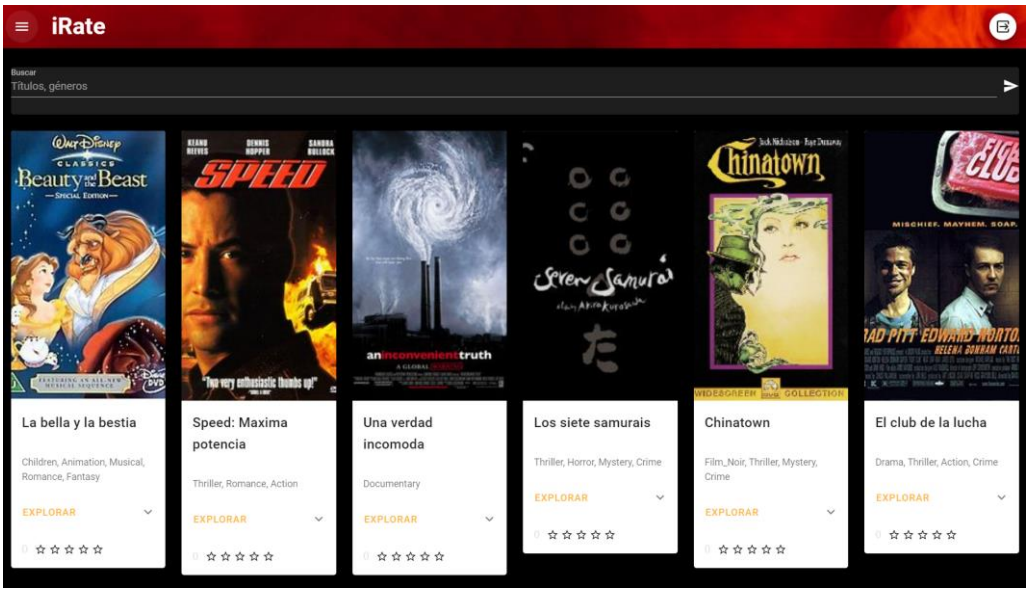

Figura 4.7.- Estado inicial ventana Buscar

Como se puede observar, la primera vez que accedemos a la ventana, las películas mostradas son aleatorias. O bueno, aleatoria no es la palabra, son las películas más populares que no han sido valoradas por el usuario hasta el momento. Películas parecidas a las mostradas en la ventana de valoración inicial de películas.

La utilización de esta ventana es extremadamente sencilla. En la parte superior se puede apreciar un cuadro de texto en el que se debe introducir el título o género que se desea buscar. Al pulsar el botón situado a la derecha o pulsar la tecla Enter en el teclado, efectuamos una consulta al servidor de aplicación, que nos devuelve las películas que contienen el patrón de búsqueda introducido en su título o que pertenecen al género introducido.

Por cuestiones de eficiencia, tan solo se devuelven las 50 películas (en caso de existir ese número) con mayor número de valoraciones y que, por supuesto, cumplen los requisitos introducidos en la búsqueda. Si por ejemplo buscamos la palabra Shrek, el servidor de aplicación nos devuelve las siguientes películas.

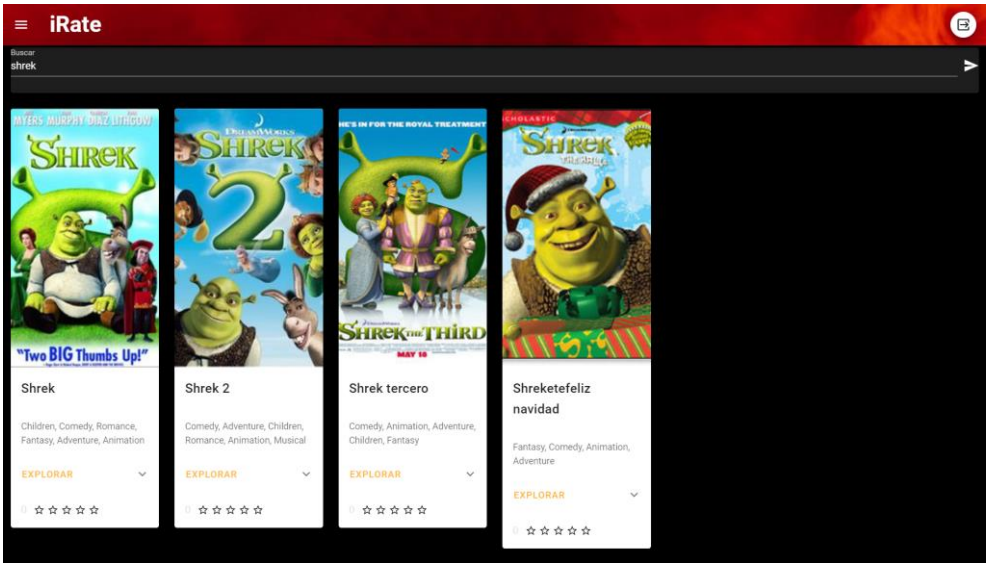

Figura 4.8.- Búsqueda por título

Como se puede apreciar, basta con que el título de la película contenga el patrón de búsqueda introducido para que se devuelvan las películas. No es obligatorio ni mucho menos realizar una búsqueda exacta.

Si ahora probamos a buscar un género, *Drama* por ejemplo, observamos que las películas obtenidas son las que pertenecen a dicho género (además de las películas que contengan

Drama en el nombre si es que existe alguna). Es, por tanto, totalmente indiferente el patrón introducido, el propio sistema se encarga de gestionar esto internamente.

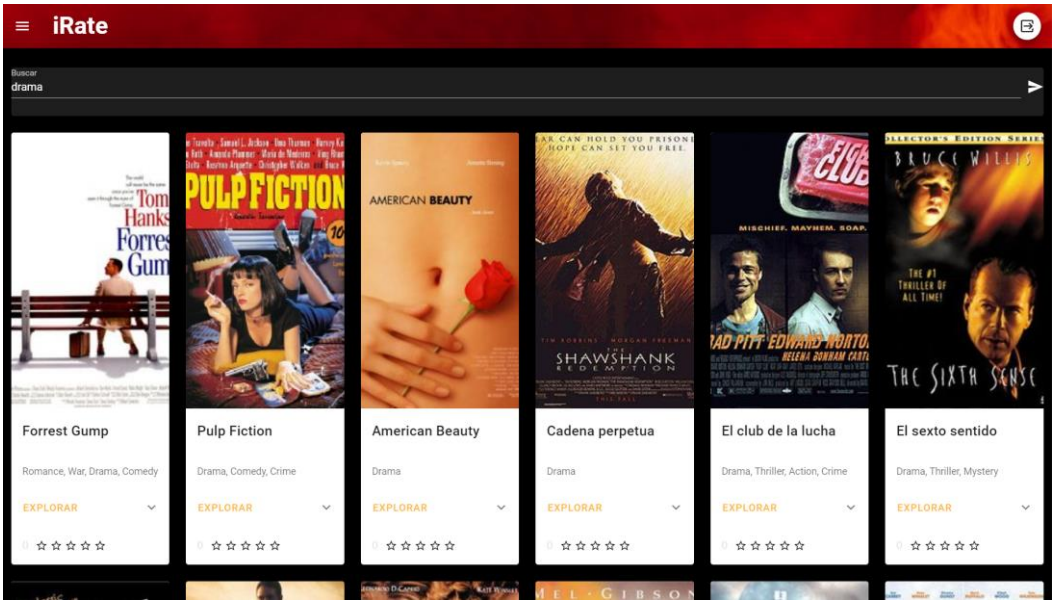

Figura 4.9.- Ejemplo búsqueda Drama

Si observamos las tarjetas de las películas, comprobaremos que difieren bastante de las mostradas en la ventana de valoraciones iniciales. En dicha ventana, la tarjeta tenía un aspecto bastante minimalista, mostrando tan solo una fotografía, el título y las estrellas de valoración. En este caso, es necesario aportar mucha más información sobre la película, así que la tarjeta mostrada tiene que tener un aspecto sustancialmente más recargado.

Además de la portada, título y valoración mostradas en las tarjetas de valoración inicial, se han añadido una serie de campos más o menos importantes. Para empezar, justo debajo del título, se indican los géneros a los que pertenece la película. Considero que, a la hora de elegir una película u otra, esta información es bastante relevante. Para visualizar el resto de la información, es necesario pulsar en el botón *Explorar*, que aumenta la altura de la tarjeta y, por tanto, también su contenido.

Figura 4.10.- Ejemplo expansión tarjeta

En esta expansión de la tarjeta, como se puede apreciar en la figura superior, nos encontramos el director de la película, año de estreno, país de origen y un link a IMDb. Debido a las limitaciones de la base de datos, no estoy en posición de mostrar toda la información sobre la película como quisiera. En base a esto, para intentar paliar esos defectos existentes en la base de datos, decidí añadir ese botón con un link a IMDb. Haciendo clic en él, se nos abre la página de IMDb de la película, donde podemos visualizar el tráiler, ver el reparto completo, una descripción de la película, presupuesto, etc. Puede que no sea la forma más elegante de hacer las cosas, pero es la única forma que se podía implementar en un periodo de tiempo tan corto.

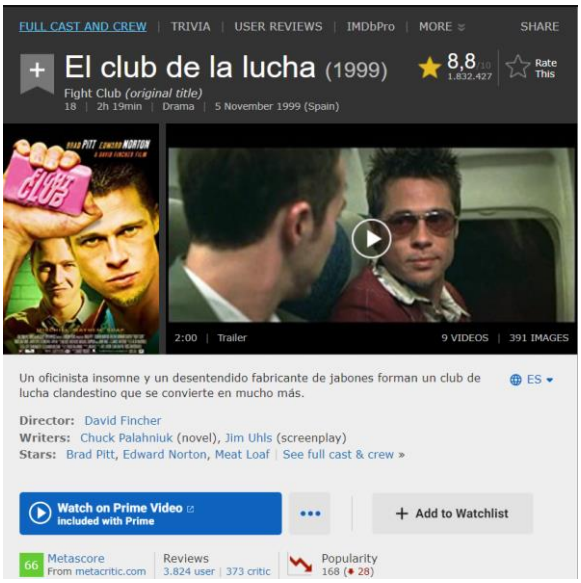

Figura 4.11.- Perfil IMDb

Además de buscar películas para valorarlas, cabe destacar otra de las principales funcionalidades que tiene la aplicación, que no es otra que el control y administración de películas valoradas.

De forma sencilla, en la ventana denominada Mis valoraciones podemos visualizar todas las películas que hemos valorado y modificar dichas valoraciones en el caso de que lo creamos oportuno. Por simples cuestiones de accesibilidad, se permite al usuario ordenar las películas mostradas en base a diversos criterios:

- Mejor valoradas primero
- Peor valoradas primero
- Más nuevas primero
- Más antiguas primero
- Orden alfabético (A-Z)
- Orden alfabético (Z-A)

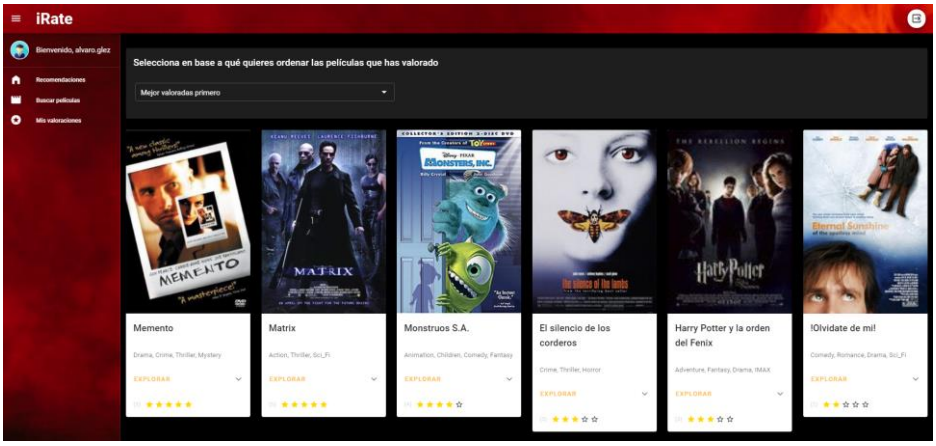

Figura 4.12.- Ventana Mis valoraciones

Habiendo descrito ya las otras dos ventanas principales de la aplicación, llega el momento de centrarse en la funcionalidad más importante de la aplicación, el sistema de recomendación. Dicho sistema, al igual que en cualquier aplicación similar, se ejecuta íntegramente en el servidor de aplicación. Sin embargo, a la hora de mostrar los resultados, es necesario mostrar al usuario las recomendaciones en un formato atractivo, lo que hace que por tanto sea necesaria una vista en la aplicación que implemente dicha funcionalidad.

Esa vista en cuestión se denomina *Recomendaciones*, y sirve precisamente para eso, mostrar al usuario las recomendaciones efectuadas por el sistema. La interfaz de la vista, al igual que

el resto de la aplicación, es extremadamente minimalista. Podemos encontrar una serie de imágenes de películas famosas que se van alternando de forma cíclica, un desplegable en el que elegir el tipo de algoritmo de recomendación a utilizar (basado en contenido, filtrado colaborativo o híbrido) y otro desplegable en el que indicar el número de recomendaciones a solicitar. Por último, se pueden visualizar las películas recomendadas ordenadas en base a la predicción realizada por el sistema.

Al acceder a la vista, se realiza de forma automática una petición al servidor de aplicaciones para que efectúe una recomendación híbrida de 40 películas. Esto se debe a que, como se comprobará en capítulos posteriores, es sin duda alguna el algoritmo más robusto de los 3 implementados.

Tras esperar unos pocos segundos, obtendremos una respuesta del servidor de aplicación y se mostrarán en la vista las películas recomendadas.

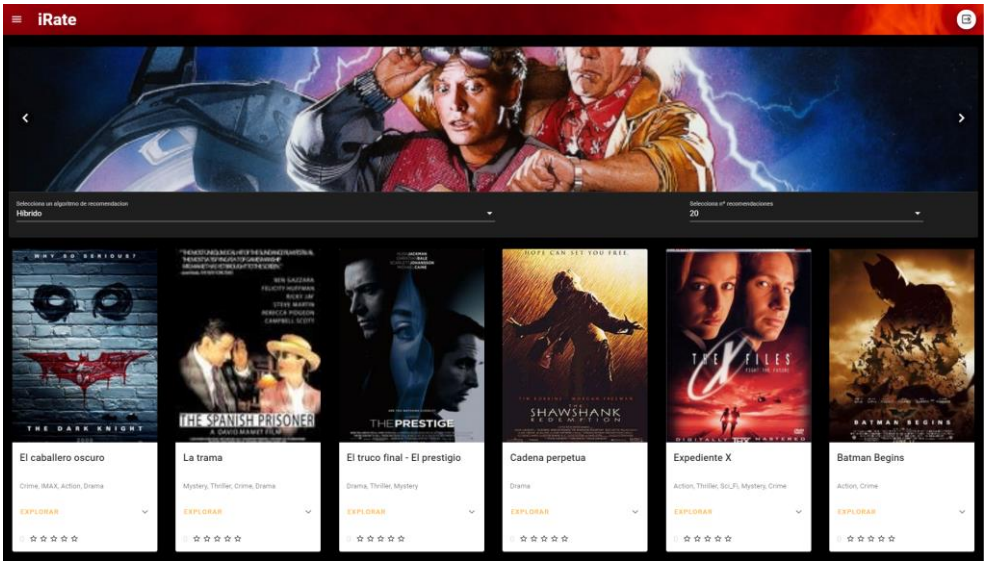

Figura 4.13.- Ventana Recomendaciones

El tiempo de espera puede variar mucho, pues depende totalmente de la potencia de cálculo que tenga el servidor de aplicación. En mi caso, estoy utilizando un ordenador portátil un poco viejo, lo que hace que las recomendaciones se demoren 2 segundos o así. En cualquier caso, considero que es un tiempo más que aceptable para la complejidad del sistema de recomendación implementado y el tamaño de la base de datos.

Seguramente, pueda chocar que en el primer desplegable se pueda elegir entre los diversos algoritmos de recomendación, la lógica diría que lo mejor es recomendar utilizando única y

exclusivamente el algoritmo híbrido dado que es el que, con diferencia, obtiene mejores resultados. Sin embargo, mi visión de la situación es totalmente diferente.

Bajo ningún concepto se debería olvidar que este proyecto está desarrollado en un ámbito 100% educativo. Yo he aprendido muchísimo realizándolo y me gustaría que, si en el futuro alguien encuentra este proyecto y decide experimentar un poco con él, pueda tener las mismas oportunidades que yo he tenido desarrollándolo. Esto quiere decir que debe poder ser capaz de visualizar las diferentes salidas de los diferentes algoritmos de una manera cómoda, compararlas o, en general, hacer lo que quiera con ellas.

5. EXPLICACIÓN DEL ALGORITMO

El algoritmo, como ya comenté al principio de esta memoria, es un algoritmo de recomendación híbrido compuesto de 2 algoritmos de diferentes, uno basado en contenido y otro de filtrado colaborativo.

Para mejor comprensión del algoritmo, que en la práctica no es un algoritmo sino tres, considero que es adecuado describir cada uno de esos tres algoritmos por separado: el basado en contenido, el de filtrado colaborativo y la suma de ambos.

Para la descripción de estos algoritmos, en lugar de basarme en el código de la aplicación, he considerado oportuno realizar una explicación teórica y, posteriormente, un ejemplo práctico muy sencillo que clarifique el algoritmo. De explicarlo sobre el código, sería más complejo de entender debido a que sería necesario tener cierto conocimiento sobre algunos módulos específicos de Python.

Cabe destacar que para el diseño e implementación de estos algoritmos me basé en 2 libros de texto:

- **Graph Powered Machine Learning** [26]: Libro en el que se habla de todo tipo de algoritmos de aprendizaje automático aplicados a bases de datos orientadas a grafos de conocimiento.
- **Recommender Systems. The textbook** [27]**:** La biblia de los sistemas de recomendación. Libro en el que se explica de forma detallada el funcionamiento de los diversos sistemas de recomendación existentes, profundizando en la lógica matemática sobre la que operan, planteando diversos escenarios, etc.

A. ALGORITMO BASADO EN CONTENIDO

Este tipo de algoritmos suelen dividirse por norma general en tres partes perfectamente diferenciadas, de modo que yo, para explicar el diseño, voy a dividir también la explicación en 3 partes.

- **Pre procesamiento y extracción de características**.

  En este punto, se extraen las características de las diversas películas presentes en la base de datos. Tras un proceso de análisis exhaustivo del problema, decidí que las características de la película que se utilizarán serían género, director y país de origen.

  Para cada película se generará un vector que tendrá tantas columnas como características diferentes existan en la base de datos. Es decir, si existen 10 géneros, 5 directores, y 3 países el tamaño del vector será de 10 + 5 + 3 = 18 características. Además, aquellas películas que hayan sido valoradas por el usuario tendrán una columna más en la que se refleje la puntuación asignada a dicha película.

  En el vector, aquellas características que no estén presentes en la película estarán representadas por un número 0. A su vez, para las características que sí estén presente se pueden dar dos supuestos diferentes:

  - Si la característica es un género o un país, se encontrará un 1 en la columna correspondiente.
  - Si la característica es un director, se encontrará un 2 en la columna correspondiente.

  Esto se debe a que, por norma general, el director de una película influye mucho más en la atracción que dicha película genera en el espectador que los géneros o el país de origen de la misma.

  Este proceso de extracción de características se realizará para todas las películas existentes en la base de datos, sin importar si han sido valoradas por el usuario o no.

- **Creación de un perfil de usuario basado en contenido**

  Una vez extraídas todas las características de cada película, es necesario modelar un perfil que represente los gustos del usuario. Para ello, tan solo se tendrán en cuenta aquellas películas que han sido valoradas por el usuario, apartando el resto de películas por el momento.

Para modelar el perfil, es necesario multiplicar cada vector de características por la valoración que el usuario ha realizado para cada película de modo que, si por ejemplo el usuario ha valorado la película con un 4, todas las características que no están presentes en la película continuarán siendo 0 mientras que las que sí están presentes pasarán a ser 4. Los directores, serán multiplicados además por 2, ya que como he comentado anteriormente tendrán el doble de peso en la recomendación que el resto de características.

A continuación, se debe calcular la media de cada una de las características, sin tener en cuenta los valores nulos (0). Posteriormente, todas las características son multiplicadas por su frecuencia de aparición. De este modo, no ocurre que una característica excepcionalmente valorada en una única ocasión altere los resultados del sistema, pues no se asume una particularidad como si de una generalidad se tratase.

Por último, es necesario aplicar una normalización al vector, de modo que los géneros y los países pasen a estar del rango [0,5] al rango [0,1] y los directores pasen a estar del rango [0,10] al rango [0,2]. Este paso es realmente necesario para calcular la similitud coseno entre el perfil y el resto de películas de la base de datos, pues las características de estas películas están definidas en esos rangos.

- **Filtrado y recomendación**

Para calcular las predicciones de puntuaciones para cada una de las películas de la base de datos no valoradas, emplearé la función de similitud coseno, expresada por la siguiente fórmula:

$$\text{Cosine}(\overline{X}, \overline{Y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}}$$

Figura 5.1.- Fórmula similitud coseno

Al calcular la similitud coseno entre el perfil generado y todas las películas no valoradas por el usuario obtendremos, para cada película, un valor entre 0 y 1. Cuanto

mayor sea esta similitud, mayor será también la probabilidad de que esta película le guste el usuario.

Una vez definidas las tres partes que forman el algoritmo y cómo se va a implementar cada una de ellas, es el momento de poner un ejemplo práctico que ayude a la comprensión del funcionamiento del mismo. Para ello, se partirá de un ejemplo sencillo con 4 películas, 2 valoradas por el usuario y 2 sin valorar.

- **Pre procesamiento y extracción de características**.

  Se partirá de un dataset compuesto por las siguientes 4 películas

Película	Características					Valoración
Película_1	Acción	Aventura	Romance	C. Nolan	USA	2.5
Película_2	Acción	Aventura	Crimen	Hitchcock	USA	5
Película_3	Acción	Crimen	Drama	Hitchcock	USA	
Película_4	Drama	Romance	Hallström	USA		

Figura 5.2.- Conjunto películas

El primer paso consiste en obtener todas las características diferentes presentes en el dataset. Dichas características son:

Características								
Acción	Aventura	Romance	Crimen	Drama	C. Nolan	Hitchcock	Hallström	USA

Figura 5.3.- Características totales

A continuación, generamos para cada película un vector de características. En los géneros y países presente en la película se pondrá un 1, mientras que en el director se pondrá un 2. En las características que no estén presentes, se pondrá un 0. En las películas valoradas, además, se añadirá una columna en la que se indicará la valoración efectuada.

	Características									
Película	Acción	Aventura	Romance	Crimen	Drama	C. Nolan	Hitchcock	Hallström	USA	Valoración
Película_1	1	1	1	0	0	2	0	0	1	2,5
Película_2	1	1	0	1	0	0	2	0	1	5
Película_3	1	0	0	1	1	0	2	0	1	
Película_4	0	0	1	0	1	0	0	2	1	

Figura 5.4.- Vectores de características generados

- **Creación de un perfil de usuario basado en contenido**

  En este paso tan solo son necesarias las dos primeras filas de la matriz, en las que se encuentran las dos películas valoradas por el usuario.

Para comenzar, se multiplica cada uno de los valores del vector por la valoración asignada, obteniendo el siguiente resultado:

	Características								
Película	Acción	Aventura	Romance	Crimen	Drama	C. Nolan	Hitchcock	Hallström	USA
Película_1	2,5	2,5	2,5	0	0	5	0	0	2,5
Película_2	5	5	0	5	0	0	10	0	5

Figura 5.5.- Vector características tras multiplicación

A continuación, se calcula la media aritmética de las características, sin tener en cuenta los 0. El perfil resultante obtenido es el siguiente:

Características								
Acción	Aventura	Romance	Crimen	Drama	C. Nolan	Hitchcock	Hallström	USA
3,75	3,75	2,5	5	0	5	10	0	3,75

Figura 5.6.- Perfil generado sin normalizar

Por último, es necesario normalizar las características para que se encuentren en el mismo rango que los vectores de características definidos en el paso 1, pudiendo así calcular la similitud coseno entre ellos. Para ello, tan solo hay que dividir cada valor entre 5. Además, hay que multiplicar cada característica por su frecuencia de aparición.

Características								
Acción	Aventura	Romance	Crimen	Drama	C. Nolan	Hitchcock	Hallström	USA
0,75	0,75	0,25	0,5	0	0,5	1	0	0,75

Figura 5.7.- Perfil normalizado

- **Filtrado y recomendación**

En este paso, partimos de los siguientes datos:

	Características								
Película	Acción	Aventura	Romance	Crimen	Drama	C. Nolan	Hitchcock	Hallström	USA
Perfil	0,75	0,75	0,25	0,5	0	0,5	1	0	0,75
Película_3	1	0	0	1	1	0	2	0	1
Película_4	0	0	1	0	1	0	0	2	1

Figura 5.8.- Películas a comparar

En base a estos datos, calculamos la similitud coseno entre el Perfil y las dos películas que no han sido valoradas por el usuario, utilizando la siguiente fórmula:

$$\text{Cosine}(\overline{X}, \overline{Y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}}$$

Figura 5.9.- Fórmula similitud coseno

Para la película 3, el cálculo de la similitud coseno sería el siguiente:

$$\cos(Perfil, p3) = \frac{0.75 * 1 + 0.5 * 1 + 1 * 2 + 0.75 * 1}{\sqrt{0.75^2 + 0.75^2 + 0.25^2 + 0.5^2 + 0.5^2 + 1^2 + 0.75^2} + \sqrt{1^2 + 1^2 + 1^2 + 2^2 + 1^2}} = 0.784$$

A su vez, el cálculo de la similitud coseno para la película 4 sería el siguiente:

$$\cos(Perfil, p4) = \frac{0.5 * 1 + 0.75 * 1}{\sqrt{0.75^2 + 0.75^2 + 0.25^2 + 0.5^2 + 0.5^2 + 1^2 + 0.75^2} + \sqrt{1^2 + 1^2 + 2^2 + 1^2}} = 0.209$$

Como se puede comprobar, en base a las similitudes coseno calculadas, es altamente probable que la película 3 resulte del agrado del usuario, mientras que es bastante poco probable que lo resulte la película 4. En base a esta puntuación calculada, será la película 3 la que el sistema recomiende al usuario.

B. ALGORITMO DE FILTRADO COLABORATIVO

Al igual que los algoritmos de recomendación basados en contenido, los algoritmos de recomendación de filtrado colaborativo también constan de 3 partes perfectamente diferenciadas.

- **Construcción matriz de valoraciones**

  Para construir la matriz, lo primero que hay que hacer es solicitar a la base que nos devuelva la información relativa a las valoraciones efectuadas por los usuarios. Como el tamaño de la base de datos es gigantesco, he decidido tras efectuar un análisis de la situación solicitar tan solo la información relativa a los 100 usuarios que comparten más valoraciones con el usuario al que se quiere recomendar. De esta manera, se reduce de forma significativa la carga computacional del sistema.

  A continuación, tras haber obtenido la información necesaria de la base de datos, ya se puede construir la matriz de valoraciones, que tendrá tantas filas como usuarios vayan a ser comparados y tantas columnas como películas diferentes hayan valorado

entre todos los usuarios. Por tanto, si quisiéramos comparar 3 usuarios que hubieran valorado entre todos 15 películas, la dimensión de la matriz de valoraciones resultante sería de 3x15.

Los valores colocados en cada posición de la matriz serán las valoraciones que cada usuario haya asignado a cada película y, en caso de no haber valorado una película, un 0. De este modo, en la posición i, j de la matriz podrá encontrarse la valoración que el usuario i asignó a la película j y, en caso de no existir una valoración para la película, un 0.

Como es lógico, la matriz de valoraciones será por norma general una matriz dispersa, pues lo más común es que un usuario haya valorado solo unas pocas películas de entre todas las existentes en la base de datos.

- **Cálculo similitud entre usuarios**

Tras haber construido la matriz de valoraciones, es necesario calcular la similitud entre el usuario para el que se quiere efectuar una recomendación y el resto de usuarios.

Aunque dicha similitud se puede calcular mediante diversas técnicas, considero que la técnica que mejor puede solventar la necesidad que se me presenta es Pearson correlation. El motivo es sencillo, es una técnica que tiene en cuenta la media de las valoraciones al realizar la comparación, algo que es especialmente importante en este caso dado que podríamos encontrarnos usuarios que valoran todas sus películas en un rango de [3.5,5] y otros que usuarios que valoran todas sus películas con puntuaciones más bajas.

Aunque existen diversas técnicas para calcular la correlación de Pearson, considero que en este caso en particular resulta mucho más interesante calcular la similitud únicamente entre las películas que ambos usuarios han valorado, pues de esta manera es posible medir con mucha más precisión cómo de similares son ambos perfiles. La fórmula para calcular dicha similitud es la siguiente:

$$\mathrm{Sim}(u,v) = \mathrm{Pearson}(u,v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}}$$

Figura 5.10.- Cálculo Pearson Correlation

- **Predicción valoraciones**

  Tras haber calculado la similitud entre todos los usuarios y el usuario al que queremos recomendar, llega el momento de predecir la puntuación que dicho usuario podría asignar a todas las películas valoradas por los usuarios más parecidos a él.

  Con el fin de reducir de forma significativa el coste computacional de las operaciones matemáticas que es necesario realizar, he tomado la decisión de tener en cuenta única y exclusivamente los 10 usuarios más parecidos.

  Para calcular esa predicción de las valoraciones, he decidido utilizar la siguiente fórmula, que tiene en cuenta la valoración media del usuario y del resto de usuarios a la hora de realizar la predicción.

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u,v) \cdot s_{vj}}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u,v)|} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u,v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u,v)|}$$

Figura 5.11.- Función de predicción

  Aunque la fórmula no lo refleje, las películas que hayan sido valoradas por un porcentaje reducido de los usuarios más parecidos no serán tenidas en cuentas. He decidido anteponer la robustez del algoritmo con respecto a las particularidades.

Una vez definidas las 3 partes que forman el algoritmo y su implementación, llega el momento de poner un ejemplo práctico que facilite la comprensión del mismo. Para simplificarlo lo máximo posible, tan solo se tendrán en cuenta 3 usuarios que hayan valorado 5 películas entre todos y que, además, tengan 3 valoraciones en común.

- **Construcción matriz de valoraciones**

  Supongamos que el usuario al que queremos efectuar una recomendación es el usuario 1, obtenemos la información de los usuarios con más valoraciones en común.

Acto seguido, construimos la siguiente matriz de valoraciones, colocando un 0 en las columnas de las películas no valoradas por el usuario.

User/Película	A	B	C	D	E
U1	3	5	2	0	0
U2	3.5	4	3	4.5	2
U3	5	2.5	5	2.5	5

Figura 5.12.- Matriz valoraciones generada

- **Cálculo similitud entre usuarios**

Para calcular la similitud entre el usuario 1 y el resto de usuarios que se pueden encontrar en la matriz de valoraciones se utiliza la siguiente implementación de Pearson Correlation.

$$\mathrm{Sim}(u,v) = \mathrm{Pearson}(u,v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}}$$

Figura 5.13.- Cálculo Pearson Correlation

Un ejemplo del cálculo de la similitud entre el usuario 1 y el usuario 2 es el siguiente.

$$Pearson(U1,U2) = \frac{(3-3.33)*(3.5-3.5)+(5-3.33)*(4-3.5)+(2-3.33)*(3-3.5)}{\sqrt{(3-3.33)^2+(5-3.33)^2+(2-3.33)^2}*\sqrt{(3.5-3.5)^2+(4-3.5)^2+(3-3.5)^2}} = 0{,}981$$

Figura 5.14.- Calculo similitud U1 y U2

Tras realizar el cálculo de las dos correlaciones de Pearson, la tabla resultante es la siguiente.

User/Película	A	B	C	D	E	Pearson(U1,i)
U1	3	5	2	0	0	
U2	3,5	4	3	4,5	2	0,981
U3	5	2,5	5	2,5	5	-0,944

Figura 5.15.- Pearson Correlation calculada

Cuanto mayor sea el valor de Pearson, mayor es la similitud entre los usuarios. Como ya podíamos observar a simple vista, el usuario 2 es muy parecido al usuario 1, mientras que el usuario 3 es muy diferente.

- **Predicción valoraciones**

Ahora que ya hemos calculado la similitud de Pearson para cada usuario, llega el momento de realizar la predicción de la puntuación de las películas no valoradas.

Antes de realizar la predicción, es necesario mencionar que solo se tendrán en cuenta las valoraciones de los usuarios más parecidos al usuario. Como este ejemplo es muy sencillo, tan sólo tendré en cuenta las valoraciones del usuario 2. Sin embargo, en la aplicación existen un número mucho mayor de usuarios, lo que hace que, a la hora de efectuar predicciones, solo haya que tener en cuenta las valoraciones de aquellos 10 usuarios más parecidos al que se desea recomendar.

Como recordatorio, la fórmula que se utiliza para realizar la predicción es la siguiente:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u,v) \cdot s_{vj}}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u,v)|} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathrm{Sim}(u,v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\mathrm{Sim}(u,v)|}$$

Figura 5.16.- Función de predicción

Un ejemplo de aplicación de la fórmula a la predicción de la puntuación para la película D es la siguiente:

$$Preddición(U1, D) = 3.33 + \frac{(4.5 - 3.5) * 0.981}{0.981} = 4.33$$

Figura 5.17.- Predicción puntuación D

Por otra parte, la aplicación de la fórmula para la predicción de la puntuación para la película E es:

$$Preddición(U1, E) = 3.33 + \frac{(2 - 3.5) * 0.981}{0.981} = 1.83$$

Figura 5.18.- Predicción puntuación E

Como podemos observar, en base a los resultados obtenidos, es altamente probable que la película D sea del agrado del usuario 1. En lo relativo a la película E, las probabilidades se reducen muchísimo.

En base a esto, el sistema recomendaría al usuario 1 la película D, que es la que ha obtenido una predicción de valoración mayor.

Por último, cabe destacar que existe la posibilidad de que alguna de las valoraciones que se realicen se encuentren en un rango diferente al [0,5], rango en el que se encuentran todas las valoraciones existentes en la base de datos. Si eso pasara, su valor se cambiaría a 0 o a 5 dependiendo de si han excedido la cota superior o inferior.

Como último paso ya del algoritmo, en vistas a una mejor integración con el algoritmo híbrido, se dividen las predicciones realizadas para cada película entre 5, logrando así que dicha predicción se encuentre en el rango [0,1], mismo rango que las predicciones efectuadas por el algoritmo basado en contenido.

C. ALGORITMO HÍBRIDO

Aunque existen diversos algoritmos de recomendación híbridos, cada uno con sus pros y sus contras, para un sistema de recomendación de películas el que mejor funciona con diferencia es el algoritmo de recomendación híbrido paralelo ponderado, que tiene el siguiente esquema:

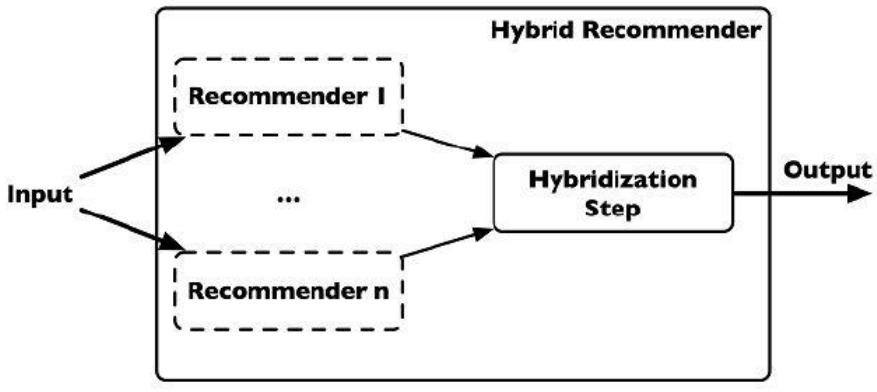

Figura 5.19.- Esquema algoritmo paralelo ponderado

De forma general, y al igual que los 2 algoritmos anteriores, este algoritmo también puede dividirse en 3 etapas perfectamente diferenciadas.

- **Unión de las salidas de ambos algoritmos**

  Como la salida de ambos algoritmos es una lista que contiene los ids de las películas y la valoración estimada, es necesario convertir este par de listas en una matriz en la que se encuentre, para cada película, la puntuación estimada por cada uno de los algoritmos de recomendación.

  En caso de que alguna película no esté presente en uno de los dos dataframes que forman la salida de los algoritmos, se pondrá un 0 en su lugar.

  Con esta técnica conseguiremos una matriz que tendrá tantas filas como películas diferentes existan entre ambas salidas y dos columnas, pudiéndose encontrar en la primera la puntuación estimada por el algoritmo basado en contenido y en la segunda la puntuación estimada por el algoritmo de filtrado colaborativo.

- **Cálculo de la suma ponderada de puntuaciones**

  Habiendo fusionado ya las salidas de ambos algoritmos, llega el momento de calcular la puntuación final de cada película, lo cual se realiza con la siguiente fórmula:

$$\hat{r}_{uj} = \sum_{i=1}^{q} \alpha_i \hat{r}_{uj}^{i}$$

Figura 5.10.- Cálculo hibridación

  A la hora de establecer las ponderaciones para cada uno de los algoritmos de recomendación, creo que lo más justo dada la más que probada efectividad de ambos es darle algo más de peso al algoritmo de CF con respecto al algoritmo CB. Por ello, he decidido establecer α_{CB} como 0,4 y α_{CF} como 0,6.

- **Ordenación en base a la puntuación estimada**

  Tras haber calculado la puntuación para cada una de las películas, aunque pueda parecer algo tribial, es necesario ordenar las películas de modo que, a la hora de devolver las n primeras, se devuelvan aquellas películas con una mayor puntuación.

Tras haber definido las 3 partes que forman el algoritmo, llega el momento de poner un ejemplo concreto en el que se facilite la comprensión del mismo. Para que el ejemplo sea lo más sencillo posible, se partirá de dos salidas de 5 y 3 películas cada una.

- **Unión de las salidas de ambos algoritmos**

  Partimos de la siguiente salida de los dos algoritmos:

Salida Algoritmo CB		Salida Algoritmo CF	
ID película	Puntuación	ID película	Puntuación
12	0,5	25	1
312	0,33	79	0,75
4231	0,75	4231	0,9
25	0,85		
79	0,4		

Figura 5.21.- Salida algoritmos

  Como se puede apreciar, algunas de las películas existentes en la salida del algoritmo CB no están presentes en las salidas del algoritmo CF, lo que hace que a la hora de fusionar ambas salidas sea necesario poner un 0 en aquellas puntuaciones inexistentes. El dataframe resultante de la unión de ambas salidas es el siguiente:

Matriz resultante		
ID película	Puntuación CB	Puntuación CF
12	0,5	0
312	0,33	0
4231	0,75	0,9
25	0,85	1
79	0,4	0,75

Figura 5.22.- Matriz resultante

- **Cálculo de la suma ponderada de puntuaciones**

  Tras haber generado la matriz resultante de la unión de ambas salidas, llega el momento de calcular la suma ponderada de ambas puntuaciones lo cual, como recordatorio, se calcula de la siguiente manera:

$$\hat{r}_{uj} = \sum_{i=1}^{q} \alpha_i \hat{r}_{uj}^{i}$$

Figura 5.23.- Cálculo hibridación

Los cálculos que se efectuarían serían, por tanto, los siguientes:

$\widehat{r_{12}} = 0.4 * 0.5 + 0.6 * 0 = 0{,}2$

$\widehat{r_{312}} = 0.4 * 0.33 + 0.6 * 0 = 0{,}132$

$\widehat{r_{4231}} = 0.4 * 0.75 + 0.6 * 0{,}9 = 0{,}84$

$\widehat{r_{25}} = 0.4 * 0.85 + 0.6 * 1 = 0{,}94$

$\widehat{r_{79}} = 0.4 * 0.4 + 0.6 * 0{,}75 = 0{,}61$

Figura 5.24.- Cálculos efectuados hibridación

Siendo la matriz resultante la siguiente:

Matriz resultante			
ID película	**Puntuación CB**	**Puntuación CF**	**Suma Ponderada**
12	0,5	0	0,2
312	0,33	0	0,132
4231	0,75	0,9	0,84
25	0,85	1	0,94
79	0,4	0,75	0,61

Figura 5.25.- Matriz ponderada resultante

- **Ordenación en base a la puntuación estimada**

Habiendo calculado ya la suma ponderada, solo nos queda ordenar la matriz en base al valor de dicha columna en orden descendente, de modo que las películas con mayor suma ponderada sean las que se encuentren en las primeras filas de la matriz. En este ejemplo concreto, la matriz final es la siguiente:

Matriz resultante			
ID película	Puntuación CB	Puntuación CF	Suma Ponderada
25	0,85	1	0,94
4231	0,75	0,9	0,84
79	0,4	0,75	0,61
12	0,5	0	0,2
312	0,33	0	0,132

Figura 5.26.- Matriz ordenada

De esta matriz se seleccionarían las n primeras películas las cuales, tras consultar a la base la información necesaria que se debe mostrar en la aplicación, serían devueltas al frontend.

Con esto doy por terminada la explicación de los tres algoritmos de recomendación. Como ya comenté en el apartado anterior, el usuario puede seleccionar qué algoritmo de recomendación quiere que se utilice. Sin embargo, como se demostrará en el siguiente capítulo, el algoritmo que presenta una mayor robustez y fiabilidad es el híbrido, con bastante diferencia con respecto a los demás.

6. ANÁLISIS DE RESULTADOS

A la hora de analizar el comportamiento del sistema de recomendación considero que, dado que está formado por tres algoritmos, es necesario analizar los resultados arrojados por cada uno de ellos por separado. Todo esto teniendo en cuenta que la salida del algoritmo híbrido no deja de ser una suma ponderada de los otros dos algoritmos, lo que quiere decir que su efectividad es parcialmente dependiente de la efectividad de los otros dos algoritmos por separado.

El orden que he decidido seguir al analizar los resultados de los tres algoritmos es el mismo seguido en el capítulo anterior, es decir, algoritmo basado en contenido, algoritmo de filtrado colaborativo y algoritmo híbrido.

A. ALGORITMO BASADO EN CONTENIDO

Para probar la efectividad de este algoritmo, he decidido diseñar una serie de pruebas variadas que sean capaces de reflejar tanto sus fortalezas como sus debilidades.

Para la primera prueba, vamos a partir de un supuesto bastante general, un usuario al que le gusta en exclusiva un tipo concreto de películas. En este caso, he seleccionado las películas que pertenecen al género *Acción*, aunque no exclusivamente, también pertenecen a otra serie de géneros.

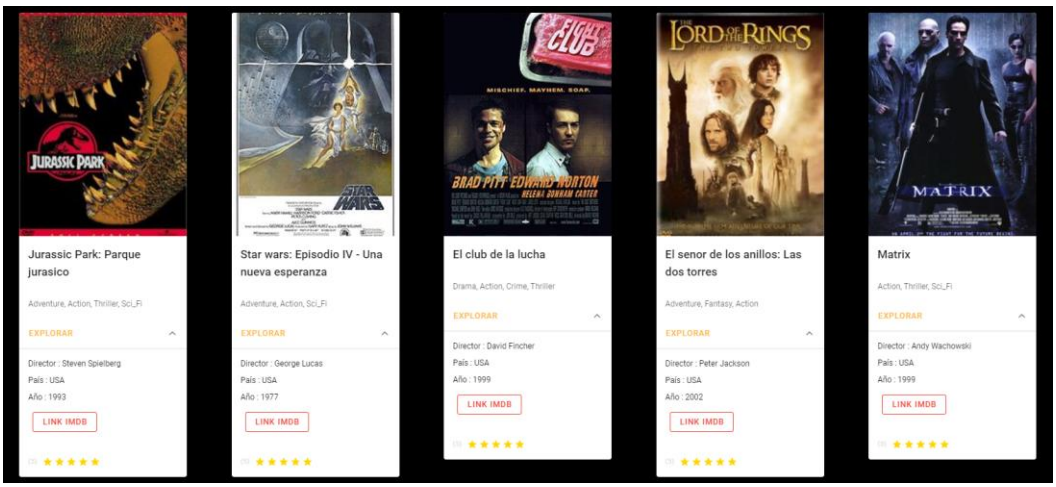

Figura 6.1.- Perfil Prueba 1 CB

Cabe destacar que, aunque los géneros de las películas son relativamente comunes (siempre con *Action* presente), los directores son totalmente diferentes, no hay dos películas dirigidas por el mismo director. Esto es especialmente relevante dado el algoritmo ponderado que he

implementado, el cual ya fue explicado en el anterior capítulo y que, como recuerdo, da más valor a los directores que al resto de características.

Si probamos a obtener una recomendación basada en contenido para un perfil que ha realizado esas valoraciones nos encontramos lo siguiente:

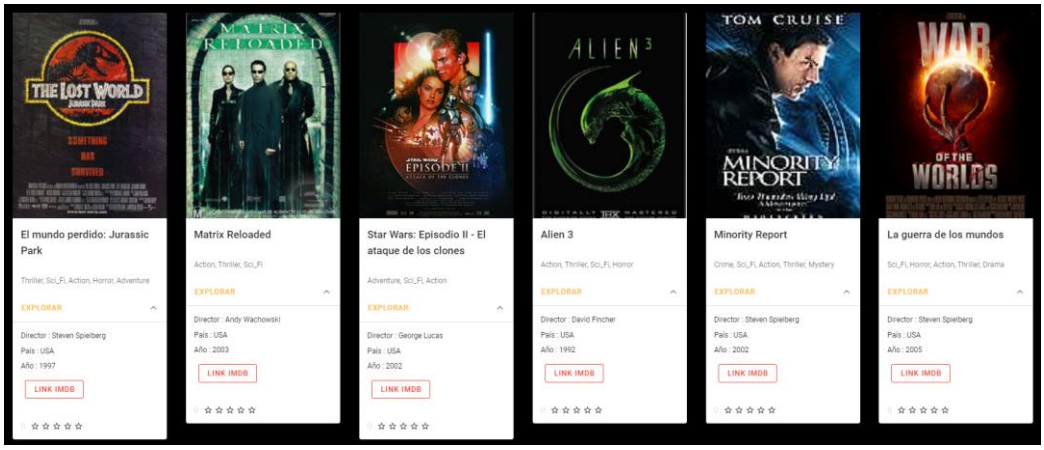

Figura 6.2.- Primera salida CB Perfil 1

Como se puede observar, todas las películas recomendadas pertenecen al género *Acción*, además de a otra serie de géneros también presentes en las películas valoradas. Cabe destacar también que los directores también coinciden totalmente.

Por último, veo necesario remarcar que las 3 primeras películas valoradas comparten saga con algunas de las películas valoradas previamente, *Jurassic Park*, *Matrix* y *Star Wars*. No es que esto deba considerarse como la prueba definitiva de la efectividad del algoritmo ni mucho menos, pero desde luego sí que sirve como indicativo de que la salida del algoritmo sigue una lógica perfectamente interpretable por el ser humano.

En vistas a agregar un poco más de información al perfil de usuario, he valorado algunas películas más, predominando 2 directores, Steven Spielberg y Peter Jackson. Para ser exactos, he valorado dos películas de cada uno, de modo que puedan encontrarse entre las películas valoradas del usuario 3 de Steven Spielberg, 3 de Peter Jackson y 1 de los demás.

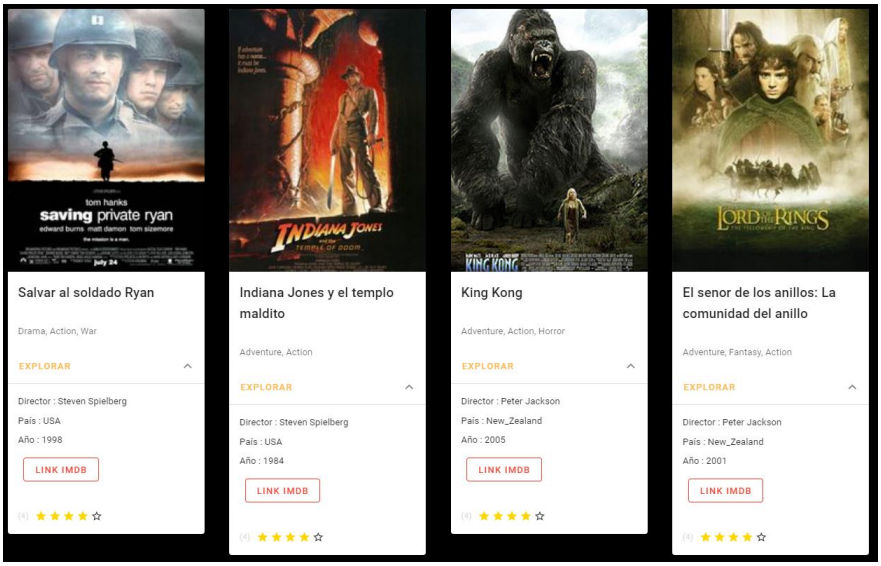

Figura 6.3.- Películas añadidas al perfil 1

Si ahora solicitamos una recomendación al sistema, la lógica parece indicarnos que, dado el doble valor que tiene el director al efectuar una recomendación, nos encontremos principalmente películas tanto de Spielberg como de Jackson.

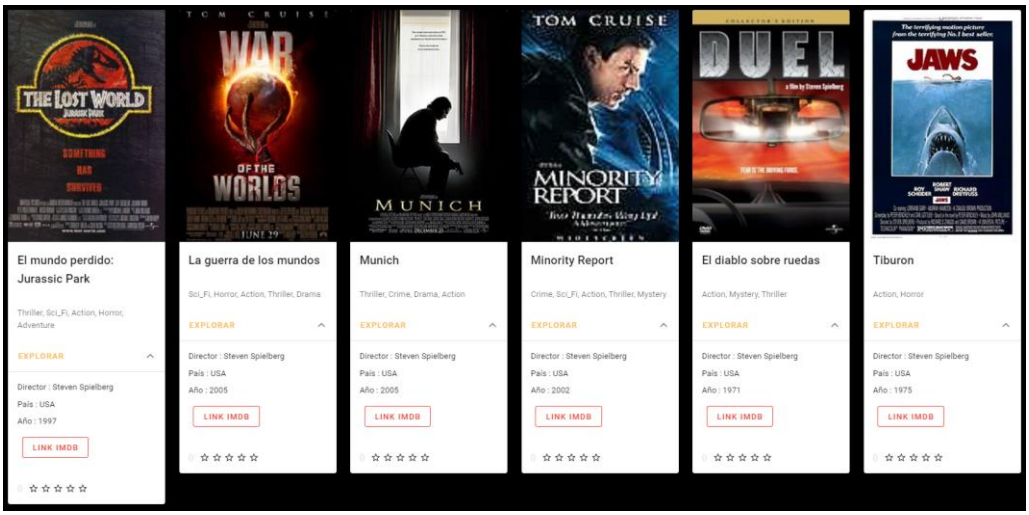

Figura 6.4.- Salida 2 CB perfil 1

Como podemos observar, todas y cada una de las primeras películas recomendadas están dirigidas por Spielberg, no está presente ninguna película de Jackson. Para analizar bien por qué pasa esto, es necesario realizar una serie de comprobaciones.

Primero de todo, es necesario observar el perfil generado por el algoritmo, que tras normalizar y adaptar en base a la frecuencia es el siguiente:

{'Adventure': 0.644, 'Action': 0.956, 'Thriller': 0.333, 'Sci_Fi': 0.333, 'steven_spielberg': 0.578,
'USA': 0.733, 'george_lucas': 0.222, 'Drama': 0.2, 'Crime': 0.111, 'david_fincher': 0.222,
'Fantasy': 0.222, 'peter_jackson': 0.667, 'andy_wachowski': 0.222, 'War': 0.089,
'New_Zealand': 0.222, 'Horror': 0.111}

Figura 6.5.- Perfil generado

En el perfil, se puede comprobar que Acción tiene una puntuación extremadamente elevada (algo normal dado que todas las películas están increíblemente bien valoradas y todas pertenecen a este género). En cuanto a las demás características elevadas, nos encontramos Aventura, Steven Spielberg, Peter Jackson y USA.

Para poder efectuar un análisis sobre por qué las películas de Peter Jackson no se recomiendan, es necesario visualizar sus características. Con este fin, he realizado una consulta Cypher que me muestre las características de las películas de Peter Jackson que no hayan sido valoradas por el usuario, obteniendo los siguientes resultados:

"title"	"genres"	"country"
"El delirante mundo de los Feebles"	["Musical","Comedy","Animation"]	"New_Zealand"
"Mal gusto"	["Horror","Sci_Fi","Comedy"]	"New_Zealand"
"Braindead: tu madre se ha comido a mi perro"	["Horror","Comedy","Fantasy"]	"New_Zealand"
"El senor de los anillos: El retorno del rey"	["Adventure","Children","Fantasy","Animation"]	"USA"
"Criaturas celestiales"	["Crime","Drama","Romance","Thriller","Fantasy"]	"New_Zealand"
"Agarrame esos fantasmas"	["Comedy","Thriller","Horror"]	"New_Zealand"

Figura 6.6.- Películas de Peter Jackson sin valorar

Como se puede observar, el resto de películas de Peter Jackson no concuerdan prácticamente en nada con el perfil del usuario. Tienen Nueva Zelanda como país de origen en lugar de USA y sus géneros son totalmente opuestos a los que caracterizan mayoritariamente los gustos del usuario. La única que realmente sí que podría ser recomendada es *El señor de los anillos*. Sin embargo, no tener Acción entre sus géneros es un lastre demasiado grande que le impide estar entre las primeras recomendaciones.

Con estas dos pruebas realizadas ya se puede comentar tanto los aspectos positivos como los negativos del algoritmo.

Como principal punto positivo, cabe destacar que las recomendaciones efectuadas son muy certeras. Todas las películas recomendadas son extremadamente parecidas a las valoradas previamente. En este sentido, no cabe duda de que el algoritmo se comporta como cualquier otro algoritmo de recomendación basado en contenido debería comportarse. Recomienda los

ítems más parecidos a aquellos con los que se ha interactuado previamente. Por expresarlo de una manera sencilla, se podría decir que se basa en la premisa "Si algo funciona, no lo cambies".

Otro punto positivo a destacar es que recomienda las películas basándose únicamente en sus características, no es relevante en ningún caso la popularidad ni el número de valoraciones recibidas. De este modo, si se añadiera una película de Peter Jackson que tuviera como géneros Acción y Aventura y estuviera rodada en USA, con total seguridad sería la primera película recomendada, sin importar como ya digo que no exista ninguna valoración en la base de datos para esa película.

En cuanto a los aspectos negativos del algoritmo, es necesario destacar el más importante de todos, la variedad de las recomendaciones. Debido al diseño conceptual de este tipo de algoritmos, siempre que se implementa uno, el desarrollador se encuentra con un "problema" característico, las recomendaciones son siempre extremadamente similares a los artículos con los que se ha interactuado en el pasado.

En este algoritmo, que desde luego no es una excepción, se puede apreciar a la perfección este problema. Si observamos la última recomendación, nos damos cuenta de que todas las películas pertenecen al género Acción, están dirigidas por Spielberg y rodadas en USA. Seguramente, parece intuitivo pensar que el algoritmo no funciona como debería. Sin embargo, esto no es realmente así. Mediante diversas técnicas que ya se comentaron en el capítulo anterior, se modela un perfil que representa los gustos del usuario. En base a dicho perfil, se recomiendan las películas más parecidas. En el caso de que un perfil esté perfectamente definido, es decir, el usuario siempre valore un tipo específico de contenido, siempre se efectuarán recomendaciones extremadamente parecidas. Y como digo, no es un error ni mucho menos, simplemente se debe a que la información extraída del usuario indica que ese contenido es el único que le gusta, con esta técnica no se realiza ninguna asunción, es pura transparencia y pura lógica.

B. ALGORITMO DE FILTRADO COLABORATIVO

A la hora de probar la efectividad de este tipo de algoritmos, es necesario destacar que no necesariamente producen salidas que puedan ser interpretadas como lógicas a simple vista.

Por poner un ejemplo que no tiene que ver con las películas, pensemos en una tienda de ecommerce como Amazon. Partamos de un usuario que ha adquirido 3 películas en DVD. Basándonos en un sistema de recomendación basado en contenido, lo lógico sería recomendarle películas similares a las ya adquiridas. Sin embargo, en un sistema de recomendación de filtrado colaborativo, se le podrían recomendar cosas diversas que hayan comprado otros usuarios tras haber adquirido las películas, ya sean reproductores DVD, taladros o un smartwatch. Recomendaciones que, aunque a simple vista puedan parecer difícilmente comprensibles, siguen una lógica más que clara con respecto a las adquisiciones de usuarios que se comportan en mayor o menor medida como el usuario inicial.

Tras este breve inciso, puedo comenzar ya a probar la efectividad del algoritmo, la cual realizaré en varios pasos diferentes.

Como punto de entrada, partiré de un usuario que ha realizado las siguientes valoraciones:

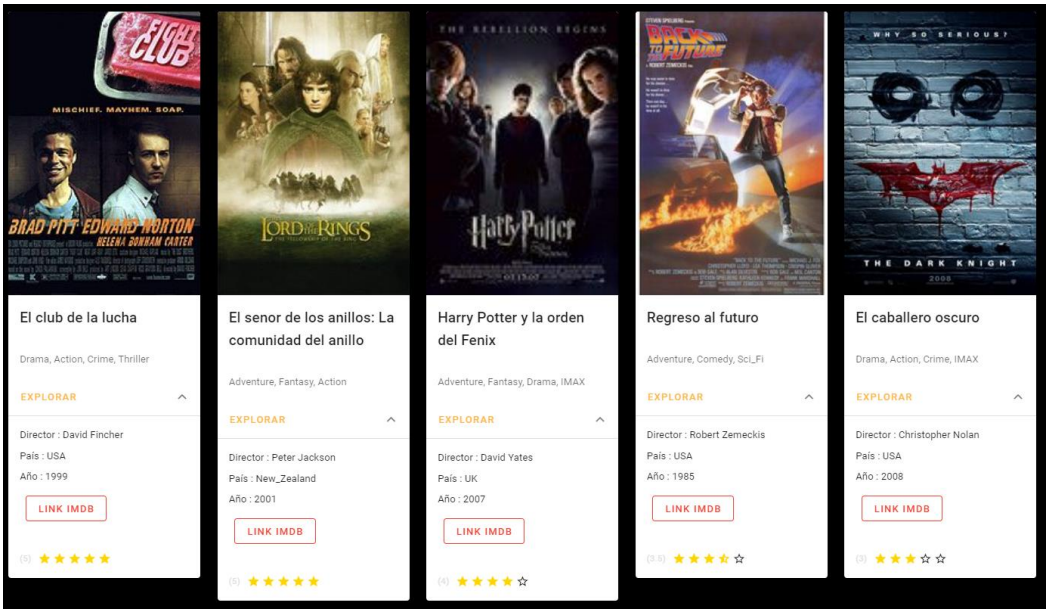

Figura 6.7.- Valoraciones usuario 1 CF

Como se puede observar, las películas valoradas tienen más o menos unos géneros parecidos, lo que a simple vista nos indica que las películas recomendadas deberían ser medianamente parecidas.

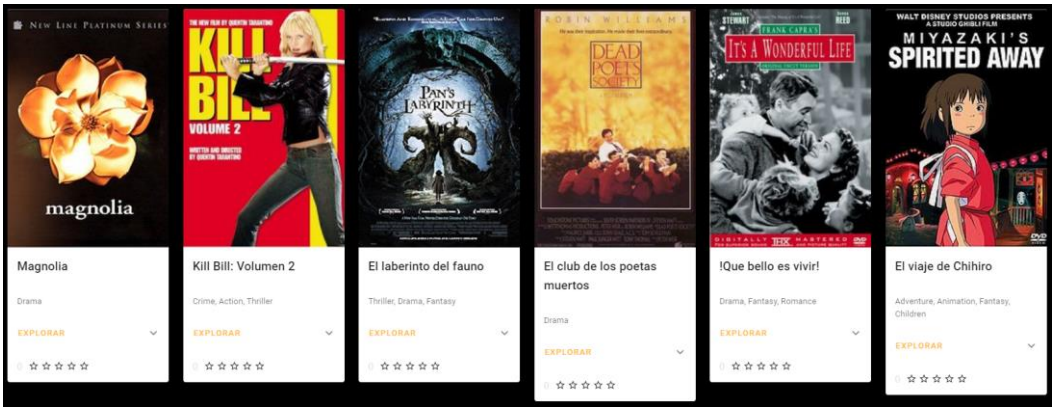

Figura 6.8.- Primera recomendación CF

Rápidamente observamos que algunas de las recomendaciones recibidas son un tanto extrañas. Magnolia (aunque por la imagen no lo parezca) es un conocido drama estadounidense en el que actúan actores realmente conocidos como Tom Cruise y Julianne Moore. No es una película muy conocida, pero es sin duda una buena recomendación. Los problemas vienen con *¡Qué bello es vivir!* y, especialmente, con *El viaje de Chihiro*. A simple vista, estas películas no tienen absolutamente nada que ver con las películas valoradas previamente por nuestro usuario de pruebas.

¿Qué está pasando por tanto? La respuesta es sencilla, hay demasiados pocos datos sobre el usuario. Cuanto mayor sea el número de películas valoradas, mayor será la efectividad del algoritmo. El motivo es bastante lógico, y está estrechamente ligado a la capacidad computacional del servidor de aplicación sobre el que corra el backend. Me explico, en una base de datos que consta de 9.000 películas y más de 800.000 valoraciones, ¿cuántas personas han valorado las mismas 5 películas? Para este ejemplo concreto, casi 500. Por desgracia, computacionalmente no es asumible calcular la similitud con esas 500 personas. Hablando de este algoritmo en concreto, tras una serie de pruebas, decidí limitar a 100 ese número. Un número más que decente, pero desde luego no suficiente (apenas representa en este caso el 20% de los usuarios con las mismas valoraciones).

Si ahora pensamos en un perfil compuesto por más películas, unas 10, parece lógico pensar que el número de usuarios parecidos al usuario se reducirá significativamente. Lo mismo pasará si valoramos 15 películas en lugar de 10, o 20 en lugar de 15. Cuantas más películas se valoren, más sencillo será encontrar los usuarios más parecidos y, por tanto, mayor será la efectividad del algoritmo.

Para intentar demostrar esta premisa, voy a añadir otras 5 valoraciones, esta vez no solo valoraciones positivas, también algunas negativas.

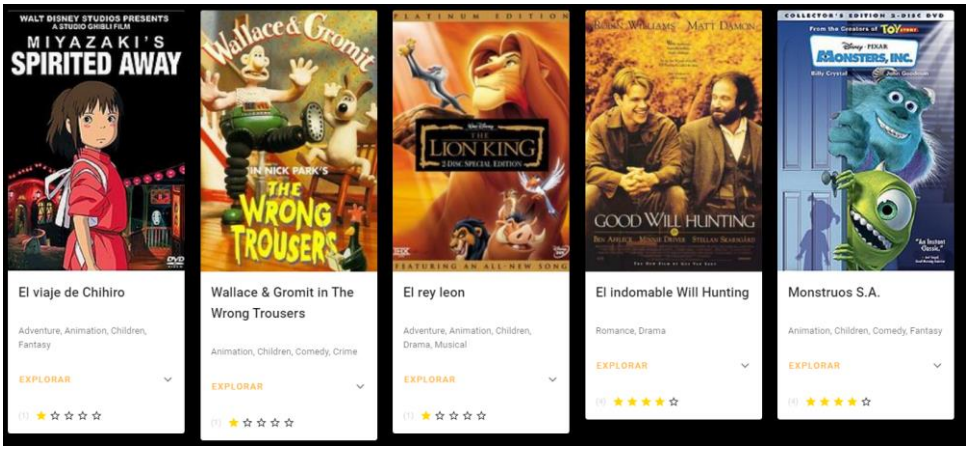

Figura 6.9.- Películas añadidas al perfil

Como podemos comprobar, las películas añadidas son principalmente películas de animación, todas ellas con mala puntuación a excepción de *Monstruos S.A*. También se ha añadido *El indomable Will Hunting* con una puntuación de 5 estrellas, todo un clásico.

Si ahora solicitamos una recomendación al sistema, por el mero hecho de haber doblado el número de películas valoradas, se debería incrementar notablemente la efectividad de la recomendación en cuestión.

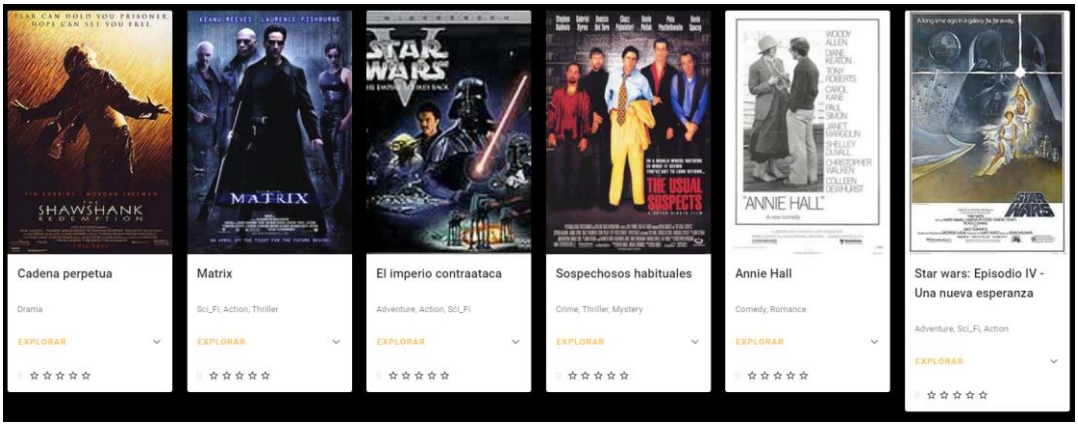

Figura 6.10.- Recomendación CF tras añadir valoraciones

La salida del algoritmo, como puede apreciarse en la figura superior, ha mejorado notablemente. Ya no nos encontramos películas extrañas como *El viaje de Chihiro,* ahora todas las películas recomendadas tienen a simple vista una pinta mucho mejor.

Habiendo demostrado que el algoritmo funciona cada vez mejor a medida que se van introduciendo datos en el sistema, es decir, valorando películas, es el momento de realizar la prueba de fuego del sistema y probar cómo se comporta con las sagas.

La lógica humana nos dice que, si a una persona le han gustado las dos primeras películas de una saga, es altamente probable que, como mínimo, vaya a ver el resto de películas de la saga. Aun así, no quiere decir que el resto de películas le vayan a gustar, hay una gran cantidad de factores que influyen en eso.

Para comprobar cómo se comporta el algoritmo ante las sagas incompletas, he realizado las siguientes valoraciones:

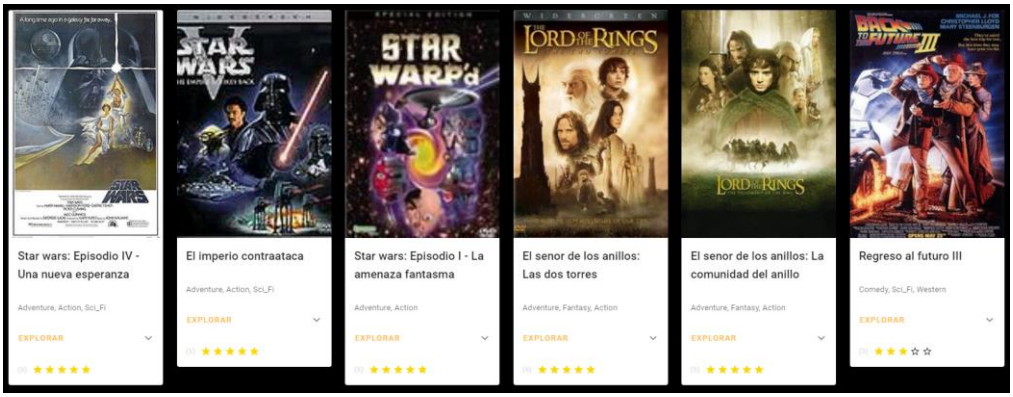

Figura 6.11.- Valoraciones CF Perfil sagas

Como puede apreciarse, se han valorado 3 películas de la saga *Star Wars* y 2 películas de la saga *El señor de los anillos*. Aunque las películas de *Star Wars* valoradas no son las tres primeras de la saga, el proceso de prueba es totalmente válido dado que no es relevante el orden de las películas a la hora de realizar una recomendación. Es un sistema inteligente, sí, pero no tanto.

Si solicitamos una recomendación al sistema nos encontramos lo siguiente:

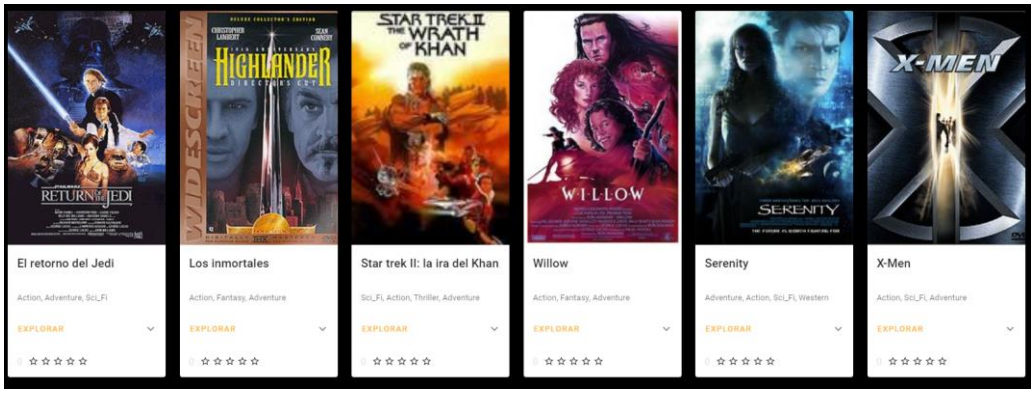

Figura 6.12.- Salida CF perfil sagas

Como podemos comprobar, entre las películas recomendadas podemos encontrar una película de la saga *Star Wars*. Sin embargo, no hay ni rastro de la tercera película de la saga *El señor de los anillos*. Para entender por qué pasa esto, es necesario realizar un par de comprobaciones sobre la base de datos.

Para empezar, es necesario consultar en la base de datos el número de valoraciones que tiene cada película de la saga.

"movie"	"num_ratings"
"El senor de los anillos: El retorno del rey"	225
"El senor de los anillos: La comunidad del anillo"	1582
"El senor de los anillos: Las dos torres"	1535

Figura 6.13.- Número valoraciones ESDLA

Los resultados arrojados por la consulta son perfectamente claros. Las dos primeras películas de la saga tienen un número de valoraciones realmente elevado. Sin embargo, la tercera película de la saga tiene un número de valoraciones muy reducido. Esto hace que las probabilidades de que dicha película sea recomendada sean también bastante reducidas. Por cada usuario que ha valorado las 3 películas de la saga (un total de 208), existen 6 usuarios que sólo han valorado las dos primeras (un total de 1248). Por ello es bastante probable que de entre los 10 usuarios más parecidos que son los que finalmente se utilizan para realizar las predicciones de las puntuaciones de cada película, no existan más de 2 o 3 en los casos más optimistas que hayan valorado la tercera película de la saga.

user_id	
51954	NaN
70331	NaN
64500	NaN
64063	4.0
66149	5.0
52481	NaN
67189	3.0
57475	NaN
53045	NaN
63198	NaN

Figura 6.14.- Valoraciones top 10

En la figura superior podemos comprobar que, en efecto, tan sólo 3 usuarios entre los 10 más parecidos han valorado *El señor de los anillos: El retorno del rey*. Además, las valoraciones de dichos usuarios tampoco son demasiado elevadas.

Con este par de comprobaciones queda perfectamente demostrado por qué no siempre se recomiendan las películas de sagas empezadas.

A modo de conclusión sobre el algoritmo, creo necesario destacar los puntos tanto positivos como negativos del mismo.

En relación a los aspectos positivos, cabe destacar que el comportamiento del algoritmo es bastante robusto, mejorando de forma significativa a medida que se va aumentando el número de valoraciones efectuadas.

Con respecto a los aspectos negativos, es necesario resaltar un problema común a casi todos los algoritmos de filtrado colaborativo. Al contrario que los algoritmos basados en contenido, estos algoritmos se basan en las relaciones entre usuarios e ítems para realizar recomendaciones, lo que quiere decir que los ítems con un número reducido de valoraciones rara vez son recomendados. Utilizando un vocabulario más técnico, y más exacto también, se podría afirmar que sufren del fenómeno denominado cold start, que surge cuando un modelo es incapaz de extraer inferencias sobre objetos o usuarios para los cuales no ha conseguido reunir suficiente información.

C. ALGORITMO HÍBRIDO

A la hora de probar el algoritmo híbrido, hay que tener en cuenta algunos factores que, en caso de desconocer, podrían alterar la percepción sobre el funcionamiento del algoritmo.

Para empezar, es necesario remarcar los pesos seleccionados para calcular la suma ponderada. Para el algoritmo de recomendación basado en contenido he seleccionado un peso de 0,4 mientras que para el de filtrado colaborativo he seleccionado uno de 0,6. ¿Qué quiere decir esto? Es muy sencillo, para cada película, la salida obtenida por el primer algoritmo se multiplica por 0,4 y la obtenida por el segundo se multiplica por 0,6. Posteriormente, ambas puntuaciones se suman obteniendo una puntuación final. Esto quiere decir que, al analizar posibles resultados, no se puede asumir que ambas salidas tienen el mismo peso.

Sumado a esto, también es necesario tener en cuenta otra cosa. Aunque el número de películas valoradas sea escaso, a poca variedad de géneros, directores y países que haya entre ellas, las puntuaciones generadas mediante un algoritmo basado en contenido son por lo general bastante pequeñas, pasando rara vez de una similitud del 50%(0,5 sobre 1 en este caso). Por otra parte, dado que actualmente existen un total de 2.135 usuarios que, de media, han valorado un total de 357 películas cada uno, es altamente probable encontrar al menos 5 usuarios que tengan valoraciones muy parecidas a las nuestras (todo esto teniendo en cuenta que el conjunto de películas valoradas en el conjunto de pruebas es reducido). Al encontrar esos usuarios con valoraciones muy parecidas, la correlación de Pearson calculada será bastante elevada (superior a 0,7), lo que con total seguridad hará que las predicciones estimadas para algunas películas estén cerca de 1.

Como se puede comprobar, por el mero funcionamiento de ambos algoritmos y como es muchísimo más sencillo que una película obtenga una puntuación elevada en un algoritmo que en otro, lo más probable es que las películas recomendadas por el algoritmo híbrido sean en su mayoría las películas mejor valoradas por el algoritmo de filtrado colaborativo.

La pregunta que parece inminente hacerse es, por tanto, ¿qué aporta el algoritmo basado en contenido entonces? La respuesta es sencilla, añadir valor a la salida efectuada por el algoritmo de filtrado colaborativo. Supongamos que un total de 20 películas tienen una puntuación muy elevada en el proceso de filtrado colaborativo, para que el ejemplo sea más sencillo, supongamos que todas han recibido una puntuación de 1. Si efectuamos la recomendación basándonos solo en esa salida, al tener todas la misma puntuación, las 5 películas que se recomienden serán 5 “aleatorias” de esas 20. Sin embargo, si le sumamos la salida del algoritmo basado encontramos que, por ejemplo, las películas de Drama y Crimen

tienen una puntuación de media de 0,5 mientras que las de Acción tienen una puntuación media de 0,25. Con esto lo que vamos a conseguir es que, con total seguridad, entre esas 5 películas recomendadas existan muchas más películas pertenecientes a los géneros Drama y Crimen que, por ejemplo, al género Acción.

Habiendo comentado ya esta serie consideraciones de obligado conocimiento para la correcta evaluación del algoritmo, llega el momento de proceder a la evaluación en cuestión. Para ello, vamos a partir del siguiente perfil de usuario:

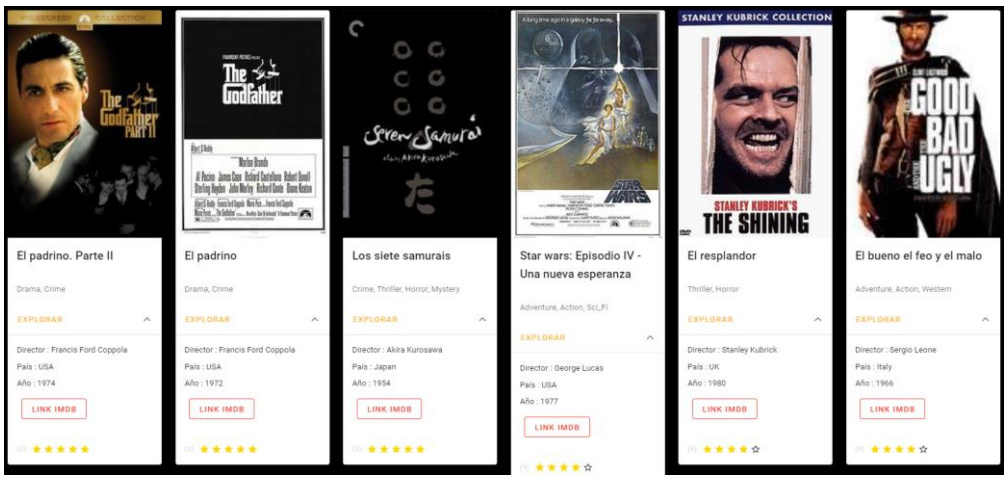

Figura 6.15.- Perfil usuario híbrido

Como se puede apreciar, nuestro usuario es todo un amante del cine clásico, Francis Ford Coppola, Stanley Kubrick, Sergio Leone... Auténticos maestros del cine de los años 70. A simple vista, el perfil de usuario está bastante definido. Si solicitamos por tanto una recomendación basado en contenido al sistema, obtenemos el siguiente resultado:

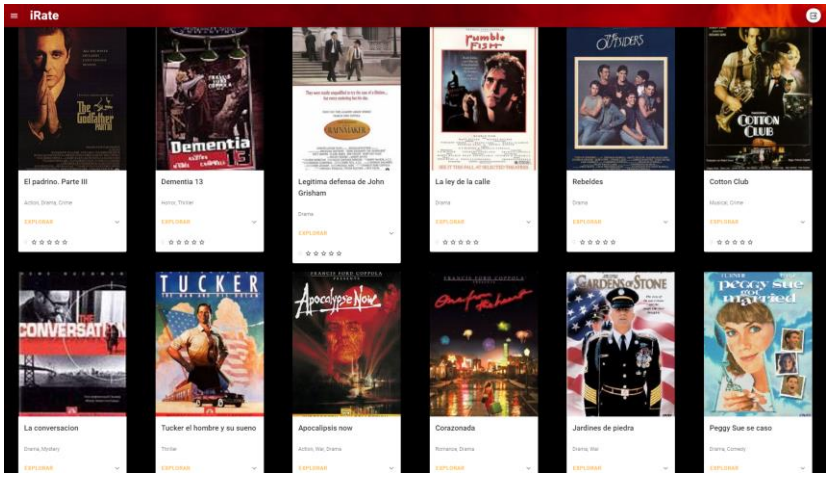

Figura 6.16.- Recomendación Basada en contenido

A simple vista ya se puede apreciar que todas las películas recomendadas son extremadamente parecidas a las valoradas con anterioridad tanto en temática como en directores, así como también lo son en la fecha de lanzamiento. En el apartado del algoritmo basado en contenido ya comenté los problemas que entraña esto, pero a modo de resumen cabe destacar un par de factores. Primero, la variedad de las recomendaciones, que es prácticamente nula. Si se pretende descubrir nuevos géneros, estilos o directores desde luego este algoritmo no nos lo va a poner fácil. Segundo, un problema innato a todo lo "antiguo". El contenido creado en el pasado, que en muchas ocasiones es magnífico, siempre será limitado. Hay lo que hay, y es lo que siempre habrá. Este factor, aunque parezca una nimiedad, es realmente importante para un sistema de recomendación de películas dado que llegará un momento en el que un usuario ya haya valorado las mejores películas de una época pasada, y si no existe un método que vaya introduciendo películas nuevas al usuario, los resultados de las recomendaciones pueden ser catastróficos.

Si comprobamos ahora cuáles son las películas recomendadas por el algoritmo de filtrado colaborativo, nos encontramos lo siguiente:

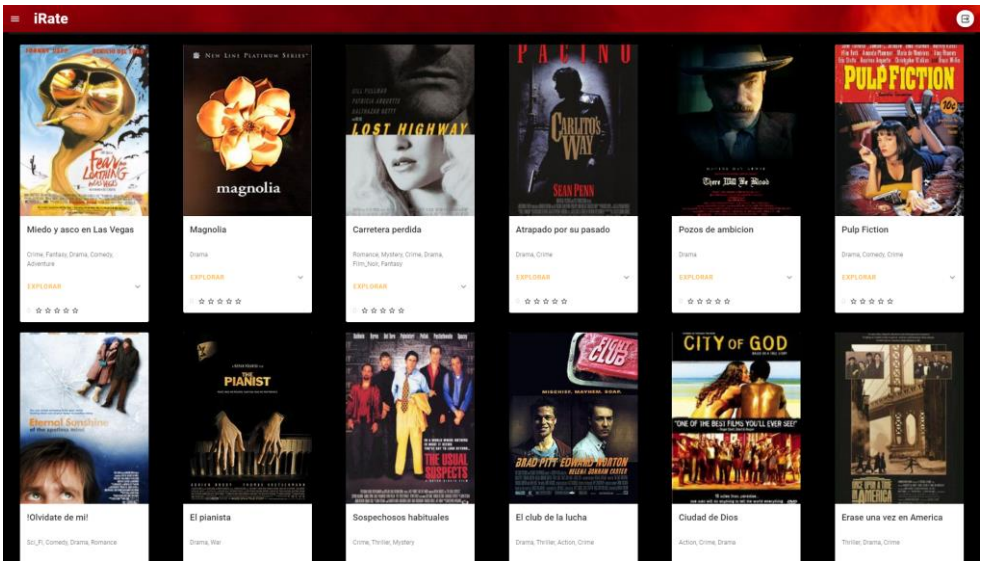

Figura 6.17.- Recomendación de filtrado colaborativo

Como se puede observar en la figura superior, si bien entre las películas recomendadas pueden encontrarse grandes obras como *Pulp Fiction*, *El club de la lucha* o *El pianista*, se echan en falta películas más antiguas. De hecho, a excepción de *Érase una vez en América*, todas las películas fueron estrenadas alrededor del año 1995 y el 2005. Si queremos encontrar algún clásico similar a los valorados con anterioridad por el usuario, debemos bajar unas

cuantas posiciones, encontrando entre las películas recomendadas algunos títulos como *Los siete magníficos* o *La naranja mecánica*.

El motivo por el que ocurre esto es bastante sencillo. Para empezar, muchas películas antiguas son bastante poco conocidas, lo que hace que el número de valoraciones en la base de datos sea menor que el de otras películas actuales y que a su vez implica que grandes películas no sean recomendadas debido al problema comentado anteriormente denominado cold start (arranque en frío). Sumado a esto, cabe destacar que gran parte del público joven no valora del todo este tipo de cine, lo que hace que sus valoraciones por norma general sean ligeramente inferiores a otras obras de igual calidad actuales.

Por último, queda probar el algoritmo híbrido, que es el realmente importante para el correcto funcionamiento de la aplicación. Los resultados que nos encontramos son los siguientes:

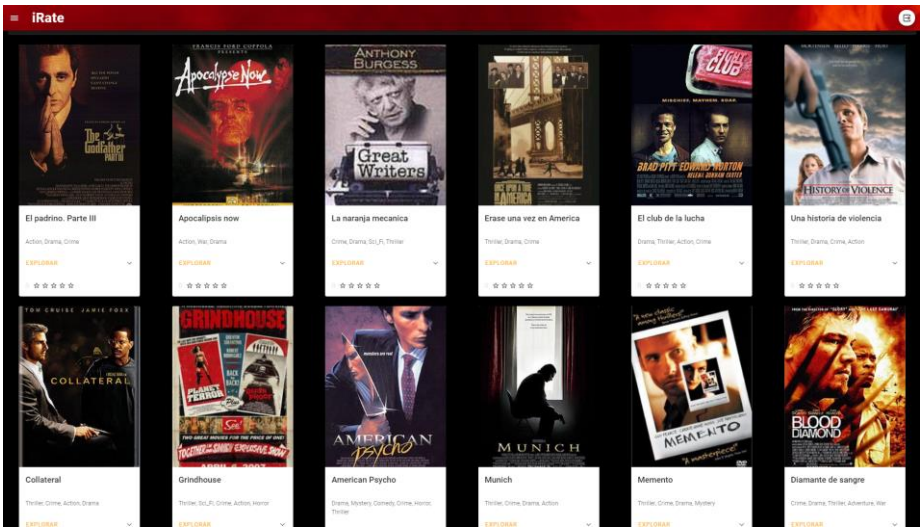

Figura 6.18.- Recomendación películas Híbrido

La salida del sistema, como podemos comprobar, difiere bastante de la salida producida por los dos algoritmos anteriores. En la salida del primer algoritmo solo nos encontrábamos películas antiguas con temáticas bastante parecidas. En la del segundo, nos encontrábamos con películas exclusivamente modernas, que poco tenían que ver con las películas valoradas previamente por el usuario. En este, por el contrario, nos encontramos una mezcla de ambos algoritmos.

Por un lado, en las primeras posiciones (que corresponden con las películas más recomendadas), nos encontramos películas antiguas similares a las valoradas por el usuario.

En la primera posición se encuentra *El padrino. Parte III* película que, si nos fijamos, es la que le falta al usuario para terminar la saga entera. En la segunda posición, se encuentra *Apocalypse Now*, considerada por muchos la mejor película de Francis Ford Coppola tras la saga *El padrino*. En las siguientes posiciones, podemos ver *La naranja mecánica* y *Érase una vez en América*, obras magnas de Kubrick y Leone respectivamente. A continuación, nos empezamos a encontrar películas mucho más modernas que las anteriores, con temáticas por supuesto bastante relacionadas con los gustos del usuario.

A modo de conclusión, cabe destacar que este algoritmo consigue reducir las debilidades de los otros dos algoritmos, resultando por tanto un algoritmo mucho más robusto y eficiente. El problema de la falta de variedad propio de los algoritmos basados en contenido se ve resuelto por la influencia de la salida del algoritmo de filtrado colaborativo. A su vez, el problema de éste último para recomendar películas con pocas valoraciones se ve parcialmente solventado por la influencia de la salida del algoritmo basado en contenido.

Aun así, cabe destacar que el algoritmo no es ni mucho menos perfecto. En relación a la importancia que tienen las salidas de los dos algoritmos, como ya comenté al principio del capítulo, es significativamente más importante la salida del segundo algoritmo, el de filtrado colaborativo. Esto debe ser así dado que es de lejos mucho más consistente que el basado en contenido. El de filtrado, se comporta mejor para el 90% de usuarios, aquellos que no tienen unos gustos extremadamente fijos. El basado en contenido, al contrario, se comporta mejor para el 10% de los usuarios, aquellos que tienen unos gustos fijos y que no cambian bajo ningún concepto. Este es el motivo por el que he decidido asignarle algo más de peso al algoritmo de filtrado colaborativo a la hora de realizar la hibridación y precisamente lo que hace que el algoritmo híbrido resultante sea ligeramente vulnerable ante el mismo problema que dicho algoritmo, el arranque en frío.

7. ANÁLISIS CRÍTICO

Como en casi cualquier proyecto realizado, es prácticamente obligatorio realizar un análisis crítico exhaustivo que permita observar de manera global cuál es la situación del proyecto. De este modo, las estrategias de marketing que se planteen tendrán una efectividad mucho mayor.

Aunque existen diversas técnicas que permiten realizar un análisis crítico para una determinada empresa o proyecto, uno de los más simples y seguramente el más visual de todos es el análisis DAFO.

Dicho análisis, cuyas siglas corresponden a las 4 áreas diferentes a analizar (Debilidades, Amenazas, Fortalezas y Oportunidades), es una herramienta de estudio que permite estudiar la situación de una determinada empresa o proyecto analizando sus características internas (Debilidades y Fortalezas) y sus características externas (Amenazas y Oportunidades) y presentándolas en forma de matriz cuadrada.

En el caso concreto de este proyecto, la matriz DAFO obtenida es la siguiente:

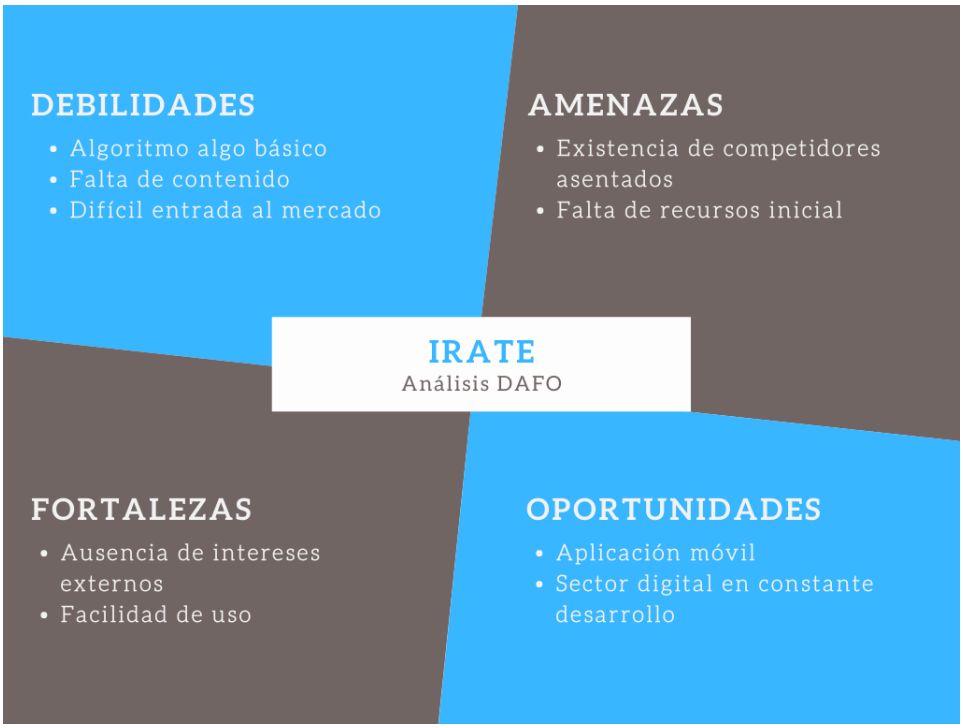

Figura 7.1.- Análisis DAFO

Aunque en la propia matriz se pueden comprender de forma rápida los puntos reseñables de las 4 áreas, considero apropiado explicar cada uno de ellos de forma un poco más extendida, aclarando aquellos conceptos que difícilmente pueden ser desarrollados en una matriz.

A. DEBILIDADES

- **Algoritmo algo básico**: Aunque ya se haya comprobado que el algoritmo tiene un comportamiento bastante robusto, hay una gran cantidad de aspectos que podrían añadirse y que mejorarían enormemente su rendimiento, tales como: Incorporación de algoritmo basado en contexto, incorporación de características como sinopsis, actores o año al algoritmo basado en contenido.
- **Falta de contenido**: Si bien puede parecer que 9.000 películas es un número más que aceptable, se evidencia claramente la falta de series en el catálogo de la aplicación. Hoy en día, este tipo de contenido es aún más consumido que las películas, de modo que carecer de él supone una gran debilidad
- **Difícil entrada al mercado**: Aunque en la era en la que vivimos pueda parecer que resulta muy sencillo hacerse un lugar en Internet, lo cierto es que sin una buena campaña de marketing esto puede resultar un obstáculo gigantesco para cualquier proyecto de nueva creación.

B. AMENAZAS

- **Existencia de competidores asentados**: En el mercado se pueden encontrar empresas como IMDb, que tienen una base de datos de películas y valoraciones gigantesca sobre la que han implementado un sistema de recomendación.
- **Falta de recursos inicial**: Como en casi todos los negocios, el comienzo puede resultar bastante duro, pues es necesario alquilar servidores con buena potencia para que sean capaces de realizar todos los cálculos asociados a las recomendaciones en un periodo de tiempo corto. Y esto, sin dinero, es muy complicado conseguirlo.

C. FORTALEZAS

- **Ausencia de intereses externos**: Como es lógico, los sistemas de recomendación que utilizan las plataformas de streaming están influenciados,

lo que hace que por norma general recomienden en mayor medida el contenido que esas propias plataformas han producido. Esta aplicación, por el contrario, es totalmente transparente y no se encuentra influenciada por ningún factor externo.

- **Facilidad de uso:** A la hora de diseñar la interfaz de la aplicación, se ha tenido mucho en cuenta la accesibilidad de los usuarios, de modo que cualquier persona independientemente de sus conocimientos informáticos pueda utilizarla sin ningún tipo de problema.

D. OPORTUNIDADES

- **Aplicación móvil**: Aunque la aplicación esté diseñada para ser utilizada en un ordenador, también puede utilizarse sin ningún tipo de problema en un dispositivo portable, lo que hace que un usuario pueda valorar películas y obtener recomendaciones con un simple clic en su teléfono móvil.
- **Sector digital en constante desarrollo**: En la actualidad, cada vez son más las personas que utilizan dispositivos electrónicos para realizar todo tipo de acciones. Por ello, el número de potenciales consumidores es gigantesco

8. LÍNEAS DE FUTURO

Debido al reducido periodo de tiempo en el que he tenido que desarrollar la aplicación, no he sido capaz de abarcar todos los campos que inicialmente me habría gustado. En un principio, siendo un completo principiante en la materia, se me pasaron un montón de cosas por alto, pero tras todo el tiempo que le he dedicado al proyecto, un montón de mejoras y de oportunidades de negocio pasan ahora mismo por mi mente.

Para empezar, es necesario desarrollar una aplicación móvil. Aunque una aplicación web esté diseñada para ser utilizada tanto en dispositivos móviles como en ordenadores, siempre resulta mucho más cómodo utilizar una aplicación nativa. En la práctica, las aplicaciones web visualizadas desde Smartphone acaban teniendo una apariencia y una usabilidad bastante inferior a la misma aplicación visualizada desde una pantalla de ordenador.

El sector móvil es un sector en constante desarrollo y evolución. No es ni mucho menos una casualidad que casi todas las empresas por diversos que puedan ser los negocios a los que se dediquen estén totalmente empeñadas en que te descargues su aplicación. Desde bancos hasta tiendas, todas estas empresas han detectado que parte de su futuro se encuentra estrechamente ligado a las aplicaciones móviles y, como es lógico, han actuado en consecuencia.

Por estas razones considero que es totalmente prioritario desarrollar una aplicación móvil a la par que la aplicación web. Si quieres entrar en el mercado en condiciones, sin duda debes hacerlo con estas dos aplicaciones de la mano.

Otra línea de futuro que sin duda tendría que ser explorada es el aumento de tamaño de la base de datos. Al partir de una base de datos generada por un tercero, siempre echaremos en falta un montón de información. En mi caso, sin ir más lejos, echo en falta que la base de datos no tenga información alguna sobre series. Al igual que también me gustaría que contuviera películas más novedosas, más información sobre el reparto, una pequeña descripción de la película, keywords…

Desde luego, parece demasiado optimista pensar que alguien va a aparecer por arte de magia en Kaggle y publicar una base de datos de estas características. Por ello, lo más efectivo

siempre será crear nuestra propia base de datos. ¿Y cómo podemos hacer eso? La respuesta es sencilla, scrapeando.

Mediante minería de datos (comúnmente denominada web scraping) somos capaces de explorar una página web como IMDb de forma automática y descargar toda esa información que podamos necesitar: películas, géneros, directores, reparto, palabras clave, descripciones, valoraciones… Absolutamente toda la información que podamos necesitar está disponible en internet a día de hoy. Si quieres construir una buena base de datos, tan solo tienes que aprender web scraping, el resto es pan comido.

Por último, cabe destacar otra línea de futuro que, si bien no es tan relevante como las dos anteriores, puede ayudar a diferenciar la aplicación en el mercado con respecto a sus competidoras.

Hay que tener una cosa muy clara con respecto a la aplicación. No es ni será nunca una aplicación de visualización de contenido. En base a esto, se presenta el siguiente problema. ¿Por qué un determinado usuario iba a molestarse en valorar películas en la aplicación si luego no va a ser capaz de encontrar donde visualizar todo el contenido recomendado? Es algo que me preocupa mucho y que puede llegar a suponer un problema crítico a la aplicación.

Para solventarlo, he pensado en lo siguiente: mostrarles dónde pueden ver el contenido que se les recomienda. Me explico. Mediante web scraping, podemos obtener todo el contenido disponible de cada una de las plataformas de streaming existentes. Si almacenamos esa información en la base de datos, podemos mostrar al usuario en qué plataformas está disponible el contenido que se le recomienda.

De esta sencilla manera, somos capaces de encontrar algo que nos diferencia del resto de competidoras, algo que nos hace únicos. Además, también somos capaces de generar valor a las plataformas de streaming. No sería algo generalizado ni mucho menos, pero en la práctica podría acabar ocurriendo que un determinado usuario acabara suscribiéndose a una determinada plataforma de streaming para ver una determinada película o serie e incluso quién sabe si también quedándose para siempre. Eso ya dependería de la plataforma en cuestión, nosotros somos y siempre seremos meros intermediarios.

9. LECCIONES APRENDIDAS

He de reconocer que esta aplicación ha supuesto todo un reto para mí, pero es precisamente por eso por lo que he aprendido cosas que ni siquiera me imaginaba al matricularme en la asignatura.

Lo primero que aprendí fueron las bases de datos orientadas a grafos de conocimiento. He de reconocerlo, antes de empezar la asignatura desconocía por completo su existencia. Jamás había oído hablar de ellas. Sin embargo, al tener que aprender su funcionamiento para el desarrollo de la asignatura, descubrí un mundo súper interesante que nos permite hacer cosas que, con bases de datos relacionales, nos supondrían un auténtico quebradero de cabeza. Recomendaciones en tiempo real, detección de fraude… Son sólo algunas de las múltiples aplicaciones que se pueden realizar con este tipo de bases de datos.

Acto seguido, tuve que aprender a interactuar con una base de datos orientada a grafos, Neo4j para ser más concreto. Al igual que las bases de datos relacionales o las bases de datos orientadas a documentos, este tipo de bases de datos también tienen su propio lenguaje mediante el que realizar consultas. El lenguaje de Neo4j, que se está convirtiendo en un Standard en el sector, se denomina Cypher. Este lenguaje, que se nota que está influido por SQL, es extremadamente sencillo de aprender si conoces las bases de SQL. Es cierto que la sintaxis es algo diferente, pero la lógica sobre la que opera es prácticamente idéntica a SQL.

Para aprender Cypher, decidí realizar un curso en la academia de Neo4j [3]. Un curso totalmente gratuito en el que se puede aprender de forma perfecta todo lo que un desarrollador pudiera necesitar para comenzar un proyecto en el que se utilicen bases de datos Neo4j.

Tras haber aprendido todo lo necesario sobre bases de datos orientadas a grafos y Cypher, llegó el momento de ponerse manos a la obra con los sistemas de recomendación. Cuando empecé el proyecto, tenía algo de idea sobre cómo funcionaban estos sistemas, pero al ponerme a desarrollar de verdad me di cuenta de que lo que realmente sabía y nada no estaban demasiado lejos.

El mundo de los sistemas de recomendación es un mundo extremadamente complejo, especialmente si, como yo, te gusta entender la lógica matemática que respalda el funcionamiento de las cosas.

En un principio, en vistas a ir observando qué tipos de sistemas de recomendación se podían desarrollar utilizando Neo4j, me leí los capítulos 4, 5 y 7 del libro *Graph-Powered Machine Learning* [26]. Este libro me sirvió como iniciación en los sistemas de recomendaciones, pudiendo aprender con ejemplos la lógica que deben seguir este tipo de sistemas. Sin embargo, a medida que fui profundizando un poco en el libro, me di cuenta de que los algoritmos desarrollados difícilmente iban a poder ser escalados a una base de datos de tamaño grande. El motivo es muy sencillo. Las bases de datos, y especialmente sus lenguajes de consulta, han sido diseñado para efectuar inserciones, borrados y búsquedas. Esto quiere decir que en ninguno de los casos han sido diseñados para realizar operaciones matemáticas por simples que sean.

Los algoritmos que en el libro se desarrollaban, podían funcionar con vectores de 10 características o con un número de relaciones insignificante. Sin embargo, ¿qué iba a pasar con 9.000 películas y 1.000.000 de relaciones? Lo más seguro es que los tiempos de espera fueran totalmente inaceptables, nadie quiere esperar 3 minutos para absolutamente nada cuando utiliza una aplicación.

Debido a esto, decidí buscar algún otro libro que explicara los sistemas de recomendación de forma teórica, permitiéndome así implementar mis algoritmos desde 0. De entre todos los libros que encontré, el que más me llamó la atención fue *Recommender Systems. The textbook* [27], que viene a ser algo así como La Biblia de los sistemas de recomendación. Leyendo este libro, que describe de forma perfecta el funcionamiento de todos los tipos de sistemas de recomendación existentes, conseguí comprender a la perfección todos esos conceptos que no se explican en los libros más técnicos y que, en mi opinión, son fundamentales a la hora diseñar un sistema de recomendación. El libro es un libro fundamentalmente matemático, lo que hace que sea estrictamente necesario poseer una cierta base matemática a la hora de poder comprender los problemas que se plantean.

En mi caso particular, como tenía bastante claro los 3 algoritmos que quería implementar, tan solo me leí los capítulos 2, 4 y 6, relativos a los algoritmos de filtrado colaborativo, basados en contenido e híbridos respectivamente.

Tras terminar la lectura de estos capítulos, se podría decir que ya sabía todo lo necesario sobre estos 3 tipos diferentes de algoritmos de recomendación y que, por tanto, los podía implementar sin ningún de tipo de problema. Decidí implementarlos en Python dado que estaba bastante familiarizado con el lenguaje y, además, creía que iba a resultar más eficiente a la hora de implementar todo lo relacionado con el análisis de datos.

Aun así, el lenguaje es lo de menos, lo verdaderamente importante es aprender cómo funcionan los algoritmos, Python no deja de ser una herramienta. Una muy completa, sí, pero una simple herramienta, al fin y al cabo.

10. BIBLIOGRAFÍA

[1] Á. G. Jiménez, «Cuaderno de trabajo,» [En línea]. Available: https://osf.io/7a6nq/?view_only=520ade80252e43e69fa9be484a3fa7a5.

[2] Á. G. Jiménez, «Repositorio Github proyecto,» [En línea]. Available: https://github.com/agonzj02/SIBI.

[3] «Neo4j GraphAcademy,» [En línea]. Available: https://neo4j.com/graphacademy/online-training/v4/00-intro-neo4j-about/.

[4] «API Javascript,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript.

[5] «API Vue,» [En línea]. Available: https://vuejs.org/.

[6] «API Vuetify,» [En línea]. Available: https://vuetifyjs.com/en/.

[7] «API Vue-router,» [En línea]. Available: https://router.vuejs.org/.

[8] «API Vuex,» [En línea]. Available: https://vuex.vuejs.org/.

[9] «API Vuex-persistedstate,» [En línea]. Available: https://www.npmjs.com/package/vuex-persistedstate.

[10] «API axios,» [En línea]. Available: https://www.npmjs.com/package/axios.

[11] «API nodejs,» [En línea]. Available: https://nodejs.org/es/.

[12] «API Flask,» [En línea]. Available: https://palletsprojects.com/p/flask/.

[13] «API Python,» [En línea]. Available: https://es.python.org/.

[14] «Vídeo Flask,» [En línea]. Available: https://www.youtube.com/watch?v=GMppyAPbLYk.

[15] «Código Flask,» [En línea]. Available: https://github.com/neo4j-examples/neo4j-movies-template/tree/master/flask-api.

[16] «API Pandas,» [En línea]. Available: https://pandas.pydata.org/.

[17] «API Numpy,» [En línea]. Available: https://numpy.org/.

[18] «API Neo4j Python,» [En línea]. Available: https://neo4j.com/developer/python/.

[19] «API Sklearn,» [En línea]. Available: https://scikit-learn.org/stable/.

[20] «API Scipy,» [En línea]. Available: https://www.scipy.org/.

[21] «Kaggle,» [En línea]. Available: https://www.kaggle.com/.

[22] «TMDB 5000 Movie Dataset,» [En línea]. Available: https://www.kaggle.com/tmdb/tmdb-movie-metadata?select=tmdb_5000_movies.csv.

[23] «The movies dataset,» [En línea]. Available: https://www.kaggle.com/rounakbanik/the-movies-dataset.

[24] «Movielens,» [En línea]. Available: https://grouplens.org/datasets/movielens/.

[25] «Extension Movielens,» [En línea]. Available: http://files.grouplens.org/datasets/hetrec2011/hetrec2011-movielens-readme.txt.

[26] A. Negro, Graph-Powered Machine Learning, Manning.

[27] C. C. Aggarwal, Recommender Systems. The textbook, Springer, 2016.