PROFESORA:

Valentina Grajales

INTEGRANTES (GRUPO 1):

Diego Felipe Carvajal Lombo (201911910)
Brenda Catalina Barahona Pinilla (201812721)
Sergio Julian Zona Moreno (201914936)

Tabla de contenido

1 Introducción 1
2 Conocimiento del dataset de trabajo 1
3 Construcción de grafo de conocimiento 2
4 Construcción del sistema de recomendación. 4
5 Sintonización y evaluación del modelo 5
6 Construya una aplicación Web sencilla 6
7 Análisis de resultados 9
8 Bibliografía y referencias 10

1 Introducción

En este documento se presenta solución al taller 3 del curso Sistemas de recomendación. Implementamos un sistema de recomendación de películas usando la librería *surprise* para *Python*, un backend desarrollado en *FastAPI* y *Neo4j* (para la construcción de grafos), y un frontend desarrollo en *React + Material Design.* Además, utilizamos la base de datos *TMDB* para poder *aumentar* nuestro conjunto de datos y realizar un enriquecimiento semántico. Finalmente, implementamos *Linked Open Data* para el filtrado ontológico con *DBPedia*.

Link de la página web: https://taller3-sr-202310.onrender.com

2 Conocimiento del dataset de trabajo

a. Ubique los datos en http://grouplens.org/datasets/movielens/, siguiendo el enlace http://grouplens.org/datasets/movielens/latest/ de septiembre de 2018, full dataset de 27 millones de ratings

Al analizar el conjunto total de datos, por temas límites computaciones, optamos por trabajar con un extracto de los mismos. Particularmente, trabajamos con una muestra de 100.000 reviews y 10.000 películas.

b. Estudie el formato de los datos, en particular identifique la manera de identificar qué información refleja la interacción entre usuarios e ítems.

Para poder hacer un entendimiento de los datos, en primer lugar, se descargaron los 4 archivos CSV proporcionados realizar un análisis de los mimos. Encontramos las siguientes columnas relevantes:

- **Movies:** movieId, title, genres.
- **Links:** movieId, tmdbId
- **Ratings:** userId, movieId, rating

Decidimos no hacer uso del conjunto "Tag" debido a que no consideramos esta información relevante para el modelo que vamos a plantear más adelante. Además, como vamos a usar la base de datos de *TMDB*, también nos pareció innecesario el valor 'imdbId' de la tabla Links.

Con estos 3 archivos, podemos hacer las siguientes relaciones:

- Utilizar la tabla de Links, para poder relacionar cada película con el ID de esta misma en la base de datos de TMDB y así poder extraer más información de esta.
- La tabla de Ratings conecta usuarios con películas, y permite otorgarles un puntaje numérico de 1 a 5, que es la base de todas las recomendaciones que vamos a realizar.

3 Construcción de grafo de conocimiento

a. ***Explore procesos de enriquecimiento ontológico sobre la información disponible sobre las películas y sobre atributos que encuentre descritos en dbpedia. Construya su propio grafo de información usando conceptos de Linked Open Data. El modelo debe incluir como mínimo los actores que participan en la película y los directores, sin embargo, se espera que explore atributos que permitan incrementar la serendipia de las recomendaciones. Verifique cómo la información disponible en http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/ le puede ayudar. Otra fuente que puede ser útil es la descarga de información del sitio web IMDB.org para uso no comercial (https://www.imdb.com/interfaces/). Se sugiere el uso de la base de datos neo4j para representar el grafo de conocimiento.***

Para poder hacer el proceso de enriquecimiento ontológico, se utilizó la base de datos *TMDB*. Esta nos proporciona un *API Key* con el cual podemos hacer consultas *HTTP* y así extraer diferente tipo de información de cada una de las películas. Además, se usó el wrapper *TMDBSimple*, el cual nos genera un conjunto de funciones para poder acceder a toda la información de las películas de una manera sencilla.

Gracias a este wrapper, se pudo generar una nueva tabla *CSV* de las películas extendida donde en cada película se le añadió:

- **Actores**: reparto de la película
- **Director**: director de la película
- **KeyWords**: conjunto de palabras claves de la película, más relevantes

- **SimilarMovies**: conjunto de películas similares dadas por el DB
- **WatchProviders**: plataformas de streaming donde se puede encontrar la película.

Con esto, podemos generar un grafo de conocimiento donde podemos relacionar las películas con cada uno de los atributos mencionados previamente y así generar un proceso de enriquecimiento ontológico. A continuación, se muestra un ejemplo de las relaciones de las películas con sus actores y director. Se aplica la misma lógica para los de más atributos.

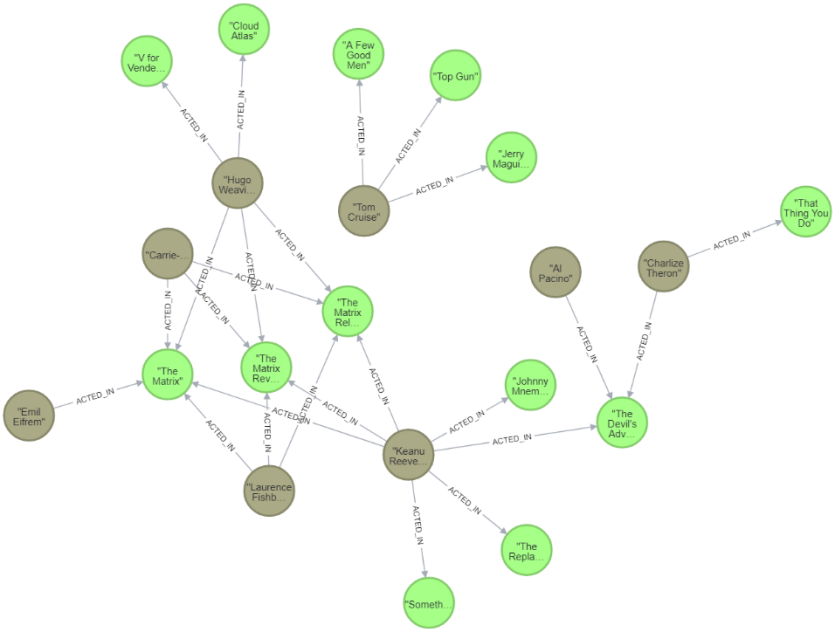

Imagen. Grafo de relación entre películas y actores.

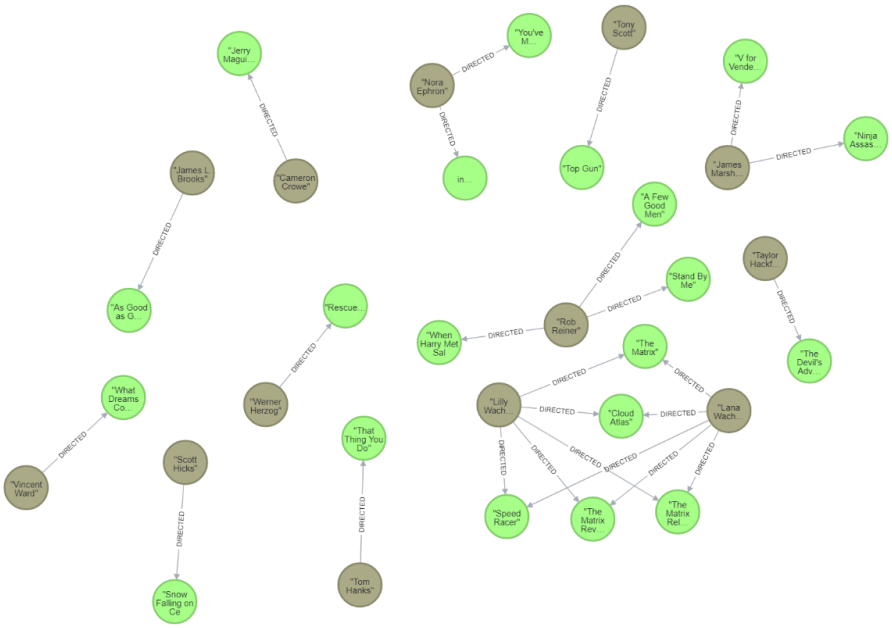

Imagen. Grafo de relación entre películas y director.

4 Construcción del sistema de recomendación.

a) Se desea implementar un sistema de recomendación de películas utilizando la información del grafo de conocimiento y filtrado colaborativo. El sistema debe, además de proveer recomendaciones para cualquier usuario presente en el dataset, proveer recomendaciones para un usuario nuevo.

Dentro de la página web el usuario, posterior a reproducir una película, podrá visualizar en la sección: "Porque viste {nombre_película} te recomendamos ver…", el *top N* de recomendaciones.

b) El sistema desarrollado debe ser un modelo híbrido, que combine filtrado colaborativo (puede ser mediante grafos también) y filtraje ontológico, aplicando una estrategia de hibridación que sea apropiada para el objetivo planteado.

Dentro de la solución planteada (como se describe en el inciso 3) se implementó adecuadamente el modelo híbrido solicitado.

c) Describa formalmente su modelo de recomendación (Framework, arquitectura del sistema híbrido y la de sus componentes).

Se construyó un modelo híbrido tipo Pipeline que contiene dos submodelos de sistema de recomendación. El primero, corresponde a un Sistema de recomendación por filtrado ontológico, que obtiene información de DBPedia mediante queries con *SPARQL*. Este permite filtrar en primera instancia aquellas películas que presenten alguna relación con la película que reprodujo el usuario. Posteriormente, el segundo sistema de recomendación implementó *KNN with Z-Score*, y es un filtrado colaborativo *item-item*. El output termina siendo un *top N* de películas, esto incluye su ranking y la "probabilidad de que te guste la película"

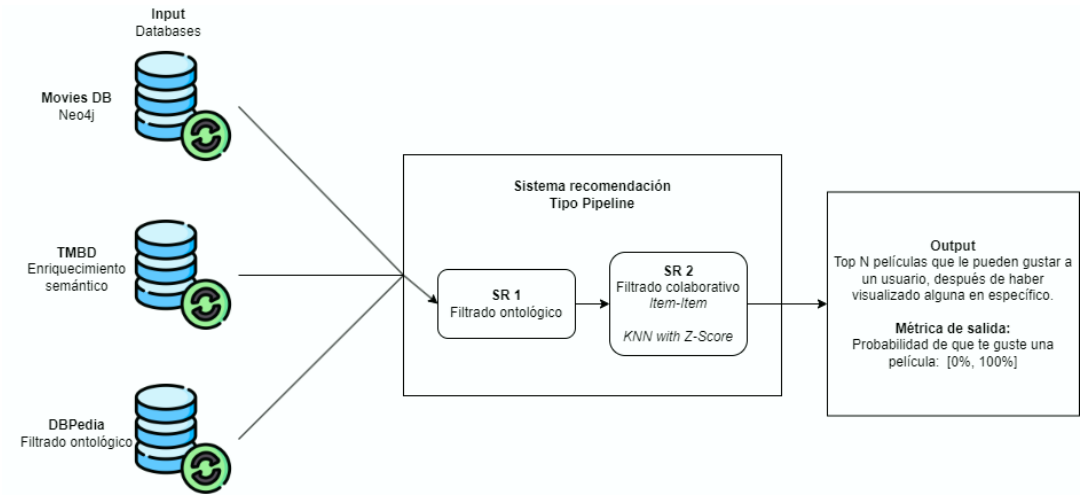

Imagen. Arquitectura del sistema de recomendación.

5 Sintonización y evaluación del modelo

a. Establezca un esquema de experimentación y de evaluación del modelo de recomendación propuesto, que permita identificar qué tan bien se cumple el objetivo de recomendación propuesto y, en particular, cómo se tienen en cuenta los aspectos de enriquecimiento semántico.

Para el esquema de experimentación y de evaluación del modelo de recomendación se hizo uso de las métricas estadísticas de RMSE y MAE. El RMSE se calcula tomando la raíz cuadrada de la media de los errores al cuadrado entre los calores reales y las predicciones del modelo. Básicamente, el RMSE mide qué tan cerca están las predicciones del modelo de los valores reales. Cuanto menor sea el valor del RMSE, mejor será la precisión del modelo. Esta métrica es útil porque penaliza los errores más grandes de manera más significativa que los errores pequeños.

El MAE se calcula tomando la media de los errores absolutos entre los valores reales y las predicciones del modelo. El MAE mide el promedio de las diferencias absolutas entre las predicciones y los valores reales. Al igual que el RMSE, el MAE también es una medida de qué tan cerca están las predicciones del modelo de los valores reales.

Para optimizar estas métricas se implementó una selección por hiperparámetros al modelo de filtrado colaborativo. Con esto, podíamos hacer diferentes búsquedas simultáneas de la mejor "configuración" de cada uno del modelo para los datos obtenidos. Repitiendo este proceso indefinidamente hasta mejorar sustancialmente los resultados de este.

```
from surprise.model_selection import GridSearchCV

param_grid = {'k': [10, 20, 30, 40, 50],
              'sim_options': {'name': ['pearson','pearson_baseline'], # pearson, pearson_baseline
                              'user_based': [False],
                              'verbose': [False],
                             },
              'verbose': [False]
             }
# Set up the grid search
gs = GridSearchCV(KNNWithZScore, param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=-1)

# Fit the grid search on the dataset
gs.fit(data)

# Print the best score and parameters
print('Best score:', gs.best_score['rmse'])
print('Best parameters:', gs.best_params['rmse'])
```

```
Best score: 0.8845480673906128
Best parameters: {'k': 40, 'sim_options': {'name': 'pearson_baseline', 'user_based': False, 'verbose': False}, 'verbose': False}
```

Imagen. Medición del modelo con hiperparámetros.

Para tener en cuenta los valores del enriquecimiento semántico, se hace un filtrado previo de las películas a evaluar por el modelo, usando únicamente las que nuestro filtro considera que tienen atributos parecidos o iguales, es decir, mismo reparto o director. Teniendo esto en cuenta, cada vez que se realizó un filtrado en la base de datos DBPedia se revisaban las coincidencias descritas previamente.

b. Además de las métricas que considere relevantes para su objetivo de recomendación propuesto, la evaluación debe realizarse utilizando las medidas de

ranking con listas de 5 y 10 items recomendados: P@5, P@10. Describa formalmente su manera de determinar si un elemento es relevante para un usuario.

Debido a que nos basamos en un sistema de recomendación basado en eventos, es decir, recomendamos ítems (películas) dependiendo del ítem con el que el usuario interactúa (visualiza). La forma en la que determinamos si un elemento es relevante para el usuario es en 2 partes.

En primer lugar, la similitud que tiene los ítems recomendados con el ítem con el que el usuario interactuó. Esto lo podemos hacer gracias al filtrado que hacemos con la información que se obtuvo de TMDB para obtener las respuestas de Dbpedia.

En segundo lugar, pero más importante, luego de obtener el subconjunto de películas parecidas a la que el usuario interactuó recientemente. Se aplica el modelo de filtro colaborativo, en donde se ordenan las películas de mayor a menor por el rating que nuestro modelo predice. De esta manera, podemos obtener un TOP 5/10 de ítems que al usuario le van a gustar.

6 Construya una aplicación Web sencilla

a) La aplicación debe permitir interactuar adecuadamente con la información, de manera que constituya una aplicación Web interesante para un usuario final. La aplicación debe permitir visualizar y evaluar la recomendación construida, con información suficiente para entender clara y eficientemente los resultados.

Para la construcción de la aplicación utilizamos la librería de JavaScript, React. Donde separamos la interfaz del usuario por componentes. Para el desarrollo de esta aplicación, se separaron las carpetas de la siguiente manera:

- Pages: Se encuentran las páginas que se utilizan en la aplicación, esta a su vez está separada por una subcarpeta “Componentes”, donde están los componentes que componen la vista de la página en React, cada uno de estos con su correspondiente archivo de diseño css. Dentro de estas carpetas se encuentran las subcarpetas: “home”, “login”, “navbar”, “recomendation”
- Routes: Se encuentra el archivo donde se especifican las rutas que se manejan en la página web.
- Services: En esta carpeta se encuentran los archivos correspondientes para realizar la conexión con los endpoints del back y obtener la información de los lugares, calificaciones, comentarios, entre otros.

Adicionalmente, al ser una página web sencilla, nos enfocamos en realizar las funcionales más importantes para mostrar e interactuar con el sistema de recomendación. Con lo anterior, se tienen las siguientes funcionalidades en la página web:

- Inicio y cierre de la sesión del usuario: en esta ocasión decidimos realizar un login para los usuarios existentes. En caso de que el id del usuario no exista, se creará un nuevo usuario de manera automática. Por otro lado, el botón el Log Out se encuentra en la parte derecha superior del navbar cuando el usuario inicia sesión.

Imagen. LogIn del la aplicación.

- Listado de películas junto con sus detalles: Cuando el usuario inicia sesión, la primera página que verá será el *home* con la lista de películas, para cada una de estas se muestra el título, el director, los actores, las palabras clave y en qué plataformas se puede ver.

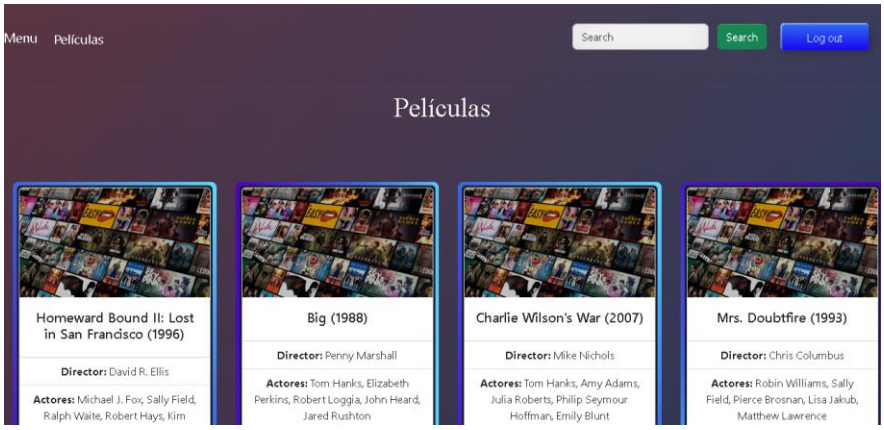

Imagen. Lista de películas.

Imagen. Detalle de una película.

- Recomendación de películas según la película vista: Cuando el usuario da clic a una de las películas significa que la vio, por lo que, en la página se despliega la vista de las recomendaciones de películas al usuario. Estas películas tienen la misma información que la lista anterior, y adicionalmente, tienen **la probabilidad de que le guste al usuario.**

Imagen. Lista de películas recomendadas.

Imagen. Descripción de Películas recomendadas.

Finalmente, también nos apoyamos de otras librerías para facilitar la construcción de los componentes, estas fueron “react-bootstrap” y “Material UI”.

7 Análisis de resultados

Elabore un informe en el que describa el modelo desarrollado, los parámetros de sintonización, sus resultados de experimentación, los resultados de evaluación y concluya sobre los resultados obtenidos. El informe debe tener una extensión máxima de 12 páginas y debe incluir todos los aspectos logrados y no logrados del trabajo, así como información suficiente para la evaluación integral de lo realizado. Debe incluir, al inicio del informe, el enlace al sitio Web de resultados.

En la experimentación offline, luego de construir el modelo híbrido con el filtrado ontológico y filtro colaborativo, fue posible realizar recomendaciones al usuario de películas que nunca ha reseñado. Al contrastar estos datos con el ítem con el que el usuario interactuó (con el que se obtienen las películas parecidas de Dbpedia), podemos ver similitudes.

A continuación, presentamos una lista con el TOP de películas recomendadas para el usuario 5 que interactuó con la película Just Cause (1995).

```
Predicting movies...
User: 5

Movie title: Just Cause (1995)
Director: Arne Glimcher
Actors: ['Sean Connery', 'Laurence Fishburne', 'Blair Underwood', 'Kate Capshaw', 'Ruby Dee']

Retrieving data from DBPedia...
```

[137]:

	movieId	prediction	percentage	tmdbId	title	actors	director	keyWords	similarMovies	watchProviders
0	86781	4.247920	84.96%	46738	Incendies (2010)	Lubna Azabal, Mélissa Désormeaux-Poulin, Maxim...	Denis Villeneuve	prison, middle east, rape, muslim, militia	Just Cause, Young Guns II, The Longest Yard, T...	NaN
1	168248	4.156063	83.12%	324552	John Wick: Chapter Two (2017)	Keanu Reeves, Common, Ian McShane, Laurence Fi...	Chad Stahelski	italy, gun, roof, hitman, secret organization	Only You, Young Guns II, Eraser, 11:14, The Av...	NaN
2	1303	4.118552	82.37%	983	Man Who Would Be King, The (1975)	Sean Connery, Michael Caine, Christopher Plumm...	John Huston	robbery, journalist, gold, cheating, treasure	Tristan & Isolde, Only You, Harlem Nights, The...	NaN
3	2181	4.114334	82.29%	506	Marnie (1964)	Tippi Hedren, Sean Connery, Martin Gabel, Dian...	Alfred Hitchcock	philadelphia, pennsylvania, post traumatic str...	Tristan & Isolde, 10, Police Story, Tales from...	Netflix, Netflix basic with Ads
4	3424	4.077714	81.55%	925	Do the Right Thing (1989)	Danny Aiello, Ossie Davis, Ruby Dee, Richard E...	Spike Lee	new york city, black people, italian american,...	1:1 Thierry Henry, 10, Just Cause, To Wong Foo...	NaN
5	2194	4.052778	81.06%	117	Untouchables, The (1987)	Kevin Costner, Sean Connery, Charles Martin Sm...	Brian De Palma	chicago, illinois, prohibition era, gangster, ...	Police Story, Hudson Hawk, Donnie Brasco, Ferr...	fuboTV, Paramount Plus, Paramount Plus Apple T...

Imagen. Top películas para usuario y película específico.

También se puede concluir que, gracias a las métricas de evaluación, las recomendaciones que hace nuestro modelo híbrido están muy bien rankeadas, gracia a que estas métricas dan valores bajos (0,84 aproximadamente). Por lo que se puede tener un gran nivel de confiabilidad en el algoritmo realizado.

8 Bibliografía y referencias

[1] Celiao. (n.d.). GitHub - celiao/tmdbsimple: *A wrapper for The Movie Database API v3*. GitHub. https://github.com/celiao/tmdbsimple

[2] The Movie Database (TMDB). (n.d.). *The Movie Database.* https://www.themoviedb.org/?language=es