Práctica 2

Sistemas Inteligentes

Contenido

[1. Resumen: 3](#_Toc72603562)

[2. Introducción: 3](#_Toc72603563)

[3. Descripción de la base de datos (DB): 3](#_Toc72603564)

[3.1 Movies: 3](#_Toc72603565)

[3.2 Ratings: 3](#_Toc72603566)

[3.3 Links: 3](#_Toc72603567)

[3.4 Tags: 4](#_Toc72603568)

[4. Carga de datos a la BD: CSV -> BD 5](#_Toc72603569)

[5. Recomendaciones: 6](#_Toc72603570)

[5.1 Métodos implementados 6](#_Toc72603571)

[5.2 Descripción y justificación de las funciones 6](#_Toc72603572)

[5.3 Seudocódigos 6](#_Toc72603573)

[6. Resultados y Conclusiones 6](#_Toc72603574)

[7. Manual de Instalación y dependencias 6](#_Toc72603575)

[8. Manual de Usuario 6](#_Toc72603576)

[9. Bibliografía: 6](#_Toc72603577)

# 1. Resumen:

En esta práctica se plantea el problema de realizar un recomendador de películas, donde a partir de datos que se nos aportan (formato csv), vamos a introducirlos en nuestra base de datos para poder ir accediendo a ellos y así poder realizar las diferentes consultas que sean necesarias para poder realizar el recomendador.

Este sistema de recomendación de películas se basa en dos partes, la primera parte consiste en recomendar a un usuario determinado un número de películas que el elija en función de los parámetros que rellene él. Estas recomendaciones saldrán representadas como el nombre de la película y el ID de la misma en la base de datos. La segunda parte consiste en a partir de un usuario y una película que no haya visto, predecir qué valoración le daría el usuario a esta misma. Estos dos procesos que acabamos de explicar se van a realizar en la relación item-item

# 2. Introducción:

Esta práctica ha surgido a partir de las sesiones realizadas en clase donde usamos el recomendador de películas de <http://movielens.org> donde probamos cómo funcionaba el sistemas y comprobamos como iban cambiando las recomendaciones según íbamos valorando más películas, hasta llegar a un total de 50 películas, donde pudimos comprobar que las recomendaciones en ese momento eran totalmente diferentes a las recomendaciones que se nos daban cuando accedimos a las página web.

Esta actividad fue el primer contacto que tuvimos con un recomendador de películas que solo habíamos visto de manera teórica hasta el momento. Una vez vimos como funcionaba se nos propuso realizar un recomendador de películas donde podremos recomendar películas a un usuario y podremos predecir la valoración que le dará el usuario a una película que él elija.

Para poder llevar a cabo este proceso de recomendaciones, partimos de una serie de datos que se nos aportan con el enunciado donde estos datos nos dan toda la información necesaria para poder realizar las recomendaciones.

Los archivos proporcionados son los siguientes:

Texto

Descripción generada automáticamente

Estos cuatro ficheros nos aportan toda la información necesaria para poder llevar a cabo la tarea. Estos datos se guardan en una base de datos creada por nosotros, y a partir de esta misma podemos empezar a acceder a los datos por consultas a la base de datos.

Este recomendador ha de ser capaz de recomendar x número de películas a un usuario determinado, siendo x un dato que puede modificar el usuario y también tiene que poder predecir la valoración que le dará el usuario a esa película sin haberla visto antes. En la parte de predicción el usuario puede elegir el usuario que quiere y cualquier de las películas que no ha visto, una vez haya introducido los datos, se podrá pasar al proceso de predicción de la valoración.

# 3. Descripción de la base de datos (DB):

Para está practica que hemos realizado hemos hecho uso de SQLite3 para poder mantener y alojar nuestra base de datos donde tenemos todos los datos necesarios para poder realizar la práctica al completo.

Lo primero que debemos hacer es crear la base de datos que vamos a usar para almacenar todos los datos necesarios para llevar a cabo esta práctica. La base de datos la creamos a mano dentro de SQLite, la cual ha sido llamada movies.db.

Una vez tenemos la base de datos tenemos que empezar a ver tabla vamos a tener que crear y que datos van a tener cada tabla. Esto lo vemos principalmente viendo los archivos **.csv** que se nos aportan en el enunciado de la práctica. Una vez descargados estos archivos y revisados, viendo los datos que hay en cada uno y que significa cada uno de ellos empezamos a realizar la planificación de las tablas de la base de datos.

En total vamos a crear 4 tablas siendo estas las siguientes: Movies, Ratings, Tags, Links.

Vamos a explicar cada atributo de la tabla y porque hemos decidido tener 4 tablas en total.

## 3.1 Movies:

Esta tabla, contiene todas las películas que hay, teniendo cada película 3 atributos, siendo estos los siguientes:

1. **movieId:** Este es el ID de la película, siendo la clave primaria de la tabla y siendo un identificador único para cada película. Es de tipo Integer.
2. **title:** Título de la película en cuestión, es de tipo Text.
3. **genres:** Este argumento indica el generó de la película en cuestión. Una película puede pertenecer a varios géneros al mismo tiempo. Es de tipo Text.

## 3.2 Ratings:

Esta tabla representa todas las valoraciones que han dado los usuarios a cada película que han visto, esta valoración va entre 0 y 5, siendo 5 la máxima valoración. Esta tabla tiene cuatro atributos siendo estos los siguientes:

1. **userId:** ID del usuario que ha realizado la valoración sobre la película. Es de tipo Interger.
2. **movieId:** ID de la película la cual el usuario le ha dado un rating. Es de tipo Interger.
3. **rating:** Rating o valoración que le ha dado el usuario a la película. La valoración puede ir desde 0 siendo la peor valoración hasta 5 siendo esta la máxima valoración. Es de tipo Integer.
4. **timestamp:** Momento en el que el usuario realizó la valoración de la película. Es de tipo Integer.

## 3.3 Links:

La tabla Links representa las películas y sus ID en las diferentes páginas en las que aparecen, las cuales son recomendadores de películas de donde se han extraídos los **csv** inciales. La tabla tiene los siguientes atributos:

1. movieId: ID de la película en cuestión, y es de tipo Integer.
2. imdbId: ID para películas usadas en [http://www.imdb.com](http://www.imdb.com/).

Ejemplo: Toy Story es tal que así <https://www.imdb.com/title/tt0114709/>

1. tmdbId: ID para películas usadas en [https://www.themoviedb.org](https://www.themoviedb.org/).

Ejemplo: Toy Story es tal que así <https://www.themoviedb.org/movie/862>.

## 3.4 Tags:

La tabla tags alberga todos los comentarios que los usuarios han realizado en las diferentes películas. La tabla por lo tanto tiene cuatro atributos, siendo estos los siguientes:

1. **userId:** ID del usuario que ha realizado el comentario sobre la película. Es de tipo Integer.
2. **movieId:** ID de la película la cual ha recibido el comentario por parte del usuario. Es de tipo Integer.
3. **tag:** Comentario que ha realizado el usuario sobre la película. Es de tipo Text.
4. **timestamp:** Es el atributo que guarda el momento exacto en el que el usuario realizo el comentario sobre la película. Es de tipo Integer.

Una vez tenemos todas las tablas descritas a la vez que sus atributos, pasamos a realizar el diagrama donde queda representada la base de datos, y queda tal que así:

Diagrama

Descripción generada automáticamente

Patrón de fondo

Descripción generada automáticamente

# 4. Carga de datos a la BD: CSV -> BD

En este apartado vamos a describir el proceso que hemos seguido para pasar todos los datos de los diferentes **.cvs** que se nos daban con el enunciado a la base de datos donde podrán ser accedidos mediante consultas a la base de datos.

Una vez tenemos la base de datos creada con todas las tablas y atributos correctos, empezamos a escribir el script el cual nos permitirá automáticamente rellenar las tablas con todos los datos. Este script hay que ejecutarlo cuatro veces, una vez por cada tabla con lo que vamos a rellenar las cuatro tablas con todos los datos.

Texto

Descripción generada automáticamente

Con la primera función es con la que insertamos los datos del csv a la tabla elegida. Como parámetros, la función tiene el fichero csv de donde vamos a sacar los datos, el nombre de la tabla donde queremos insertar este archivo, y el número de columnas que tiene el csv. Esta misma función llama a la siguiente función **insertar\_csv\_a\_tabla** donde vamos como parámetros le pasamos el nombre de la tabla donde queremos insertar, el numero de columnas que tiene la tabla y el numero de filas que hay en el csv para poder realizar un bucle donde vamos cogiendo los datos de cada fila para insertarlos con una query, una vez nos hayamos conectado a la base de datos.

Para rellenar todas las tablas, debemos ejecutar esta función con cada una de las tablas que vamos a usar, en este caso cuatro tablas. El código para la ejecución queda tal que así:

Texto

Descripción generada automáticamente

# 5. Recomendaciones:

## 5.1 Métodos implementados

## 5.2 Descripción y justificación de las funciones

## 5.3 Seudocódigos

# 6. Resultados y Conclusiones

# 7. Manual de Instalación y dependencias

# 8. Manual de Usuario

# 9. Bibliografía:

En este apartado vamos a incluir todos los enlaces a las fuentes que hemos usado para realizar la práctica:

1. <https://stackoverflow.com/questions/41880447/i-want-to-populate-a-ttk-combobox-with-results-from-database-query>

2. <https://stackoverflow.com/questions/8421614/how-to-add-items-to-a-qcombobox-in-pyqt-pyside>

3. <https://stackoverflow.com/questions/24044421/how-to-add-a-row-in-a-tablewidget-pyqt>

4. <https://www.it-swarm-es.com/es/python/similitud-de-coseno-entre-2-listas-de-numeros/1040572616/>

5. <https://stackoverflow.com/questions/44707794/pyqt-combo-box-change-value-of-a-label>

6. <https://stackoverflow.com/questions/19929186/combobox-null-in-if-statement>

7. <https://stackoverflow.com/questions/38001976/pyqt5-qcombobox-get-value-of-combobox>

8. <https://gis.stackexchange.com/questions/36835/run-function-if-item-in-qcombobox-is-selected>

9. <https://www.mysqltutorial.org/mysql-where/>

10. <https://towardsdatascience.com/step-by-step-guide-building-a-prediction-model-in-python-ac441e8b9e8b>

11. <https://stackoverflow.com/questions/16856647/sqlite3-programmingerror-incorrect-number-of-bindings-supplied-the-current-sta>

12. <https://www.techbeamers.com/program-python-list-contains-elements/>