

Sistema recomendador de películas

# P2 – Sistemas Inteligentes

Manuel Salvador y Daniel Sabbagh

---

## Contenido

1. Resumen:.....	2
2. Introducción: .....	2
3. Descripción de la base de datos (DB):.....	3
3.1 Movies:.....	3
3.2 Ratings:.....	3
3.3 Links:.....	4
3.4 Tags: .....	4
4. Carga de datos a la BD: CSV -> BD .....	5
5. Recomendaciones: .....	6
5.1 Métodos implementados, Descripción y Pseudocódigos .....	6
6. Resultados y Conclusiones .....	8
7. Manual de Instalación y dependencias.....	9
8. Manual de Usuario .....	9
8.1 Recomendación:.....	10
8.2 Predicción:.....	10
9. Bibliografía: .....	10

## 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 se ha planteado 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 cómo 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:

 links.csv	26/09/2018 22:50	Archivo de valores...	194 KB
 movies.csv	26/09/2018 22:49	Archivo de valores...	483 KB
 ratings.csv	26/09/2018 22:49	Archivo de valores...	2.426 KB
 tags.csv	26/09/2018 22:49	Archivo de valores...	116 KB

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 esta práctica 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 género 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 Integer.
2. **movieId:** ID de la película la cual el usuario le ha dado un rating. Es de tipo Integer.
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** iniciales. La tabla tiene los siguientes atributos:

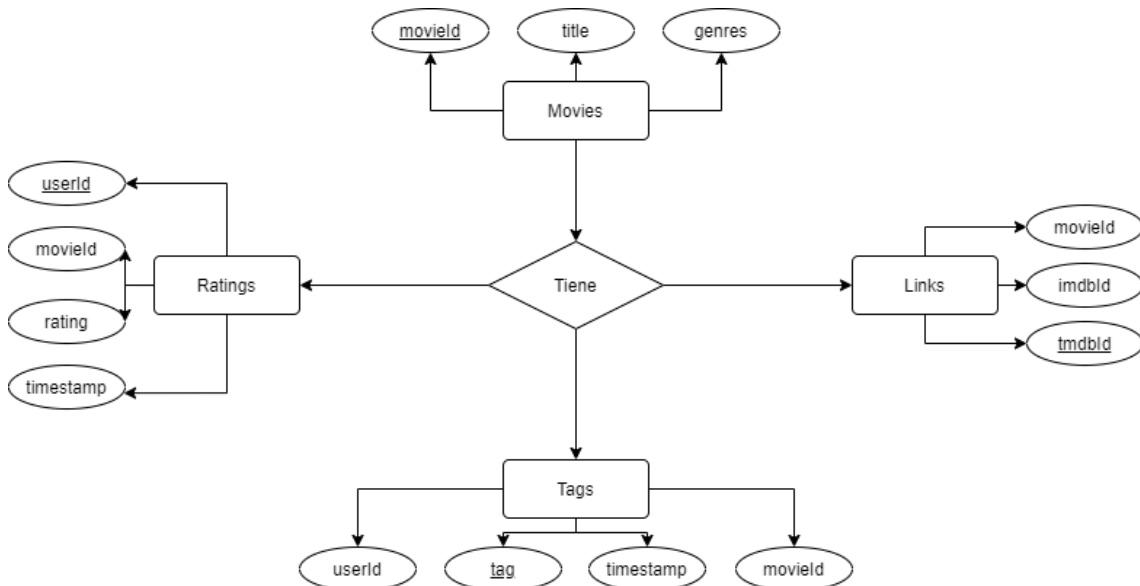
1. **movieId**: ID de la película en cuestión, y es de tipo Integer.
2. **imdbId**: ID para películas usadas en <http://www.imdb.com>.  
Ejemplo: Toy Story es tal que así <https://www.imdb.com/title/tt0114709/>
3. **tmdbId**: ID para películas usadas en <https://www.themoviedb.org>.  
Ejemplo: Toy Story 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:



▼  Tablas (4)	
>  links	CREATE TABLE "links"
>  movies	CREATE TABLE "movies"
>  ratings	CREATE TABLE "ratings"
>  tags	CREATE TABLE "tags"

#### 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 **.csv** 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.

```
def fill_table(file_name, table_name, n_col):
    file = open(file_name, encoding="utf8")
    rows = csv.reader(file)
    next(file)
    insertar_csv_a_tabla(table_name, n_col, rows)

#INSERTAMOS EN LA BASE DE DATOS
def insertar_csv_a_tabla(table_name, n_col, rows):
    con = sqlite3.connect('movies.db')
    cur = con.cursor()
    cur.execute("SELECT COUNT(*) FROM " + table_name)
    result = cur.fetchall()
    values = ""
    i = 0
    for i in range(n_col):
        values = values + "?,"
    values = values[:-1]
    if result[0][0] == 0:
        cur.executemany("INSERT OR IGNORE INTO " + table_name + " VALUES (" + values + ")", rows)
    con.commit()
    con.close()
```

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:

```

if __name__ == "__main__":
    fill_table('movies.csv', 'links', 3)
    fill_table('ratings.csv', 'links', 4)
    fill_table('links.csv', 'links', 3)
    fill_table('tags.csv', 'links', 4)

```

## 5. Recomendaciones:

### 5.1 Métodos implementados, Descripción y Pseudocódigos

A continuación, se listan y se explican los métodos utilizados, en primer lugar, para poder aplicar las predicciones:

- “fill\_table”: Se encarga de rellenar las tablas de la base de datos leyendo los archivos .csv, para ello utiliza el método “insert\_csv\_to\_table”.

```

fill_table(nombreFila, nombreTabla, numCol){
    abrir csv en utf-8.
    leerlo.
    insertarlo con insert_csv_to_table(nombreTabla, numCol, filas).
}

```

- “insert\_csv\_to\_table”: Se encarga de conectar el proyecto Python a la base de datos en local Sqlite3, para ello hace uso del import sqlite3 y se conecta con dicha base de datos llamada “movies.db”, que se encuentra en la raíz del proyecto, se establece un cursor que apunta a la consulta select para extraer todas las tablas y posteriormente rellenar cada tabla con el nombre del archivo csv que le pertenece a cada una, todo esto automático sin necesidad de hacerlo tabla a tabla. Por último, se cierran las conexiones de la base de datos.

```

insert_csv_to_table(nombreTabla, numCol, filas){
    conexión Sqlite3 en /movies.db.
    query obtención de tablas.
    query insercción valores.
}

```

- “vectorInicial”: Se encarga de generar un vector los id de usuarios y las valoraciones de todos los usuarios que han visto la película seleccionada en la interfaz.

```

vectorInicial(pelicula){
    query de obtencion de usuarios que han visto la pelicula del parametro
    return vector de dichos usuarios.
}

```

- “pelisVistas”: Pasandole por parámetro el id del usuario seleccionado en la interfaz se consigue un vector de todas las películas que este usuario ha visto.

```

pelisVistas{
    query de peliculas que ha visto el usuario seleccionado.
    return vector de dichas peliculas.
}

```

- “PelisVistasUsuarioSeleccionado”: Aquí recibe por parámetro lo que se devuelve del método anterior, “pelisVistas”, y se genera un nuevo vector de los id de los usuarios y sus valoraciones teniendo en cuenta que se contendrán varios vectores dentro de otros, una lista de listas al ser más de uno los usuarios que han visto las películas.

```
PelisVistasUsuarioSeleccionado{  
    query de usuarios y valoraciones donde movieId sea cada uno de la lista de PelisVistas()  
    return vector de listas de todos los usuarios con sus valoraciones.  
}
```

- “ComparandoVectores”: Una vez que obtenemos todos los usuarios y sus valoraciones de las películas que ha visto el usuario seleccionado y todas las películas de este mismo usuario seleccionado a partir de la película seleccionada y sus respectivas valoraciones, comprobamos que tengan la misma cantidad de películas vistas y que lleguen a coincidir que hayan visto las mismas películas todos para poder usar posteriormente la similitud del coseno y empezar con el proceso de predicción, este método devuelve un array de ratings que hayan sido aceptados

```
comparandoVectores{  
    comparación de tamaño y datos de los vectores obtenidos.  
    return vector con datos validos para la similitud del coseno.  
}
```

- “similitudCoseno”: Para calcular la similitud del coseno, hacemos uso de la librería spacy (spatial), donde le pasamos, tanto la lista principal de películas que el usuario seleccionado ha visto y valorado, como las listas de otros usuarios que sean compatibles con ésta. Todo esto nos devuelve una cantidad de similitudes de coseno igual a la cantidad de vectores que hayan coincidido en el método “Comparando Vectores”.

```
similitudCoseno{  
    Calculo de similitud del coseno con los valores obtenidos de comparandoVectores() y vectorInicial().  
    return lista de similitudes del coseno.  
}
```



## 6. Resultados y Conclusiones

En este apartado vamos a hablar de los resultados obtenidos en la práctica que hemos realizado. Estos resultados se dividen en dos categorías, resultados de recomendación y resultados de predicción.

En la parte de recomendación, una vez hemos introducido valores en todos los campos necesarios, pulsamos el botón de recomendar, el cual, va a procesar todos los datos para poder aportar el número de recomendaciones que el usuario ha elegido. Este resultado podrá ser observado en la tabla de ranking en la que se ve de la siguiente manera:

**Recomendaciones**

Selecciona un Usuario (ID)  ▼ Recomendar!

Items ranking  Umbral de similitud:

**Ranking**

	ID item	Prediccion
1	333	5.0
2	1272	4.5
3	2746	4.5
<div>&lt; <span style="background-color: #ccc; width: 100px; display: inline-block;"></span> &gt;</div>		

Con respecto al sistema de predicción se selecciona la película no vista y el usuario y en dicha película, teniendo en cuenta la algoritmia seguida en el backend podrá llegar a predecir qué valoración a continuación se facilita un ejemplo:

Selecciona un Usuario:  ▼

Selecciona una película:  ▼ Predecir!

Predicción: **3,5**

Como podemos observar vemos el campo de predicción con un valor que representa la predicción que daría el usuario elegido a la película escogida.

## 7. Manual de Instalación y dependencias

Para la instalación, tan solo hay que descargarse el código, abrirlo en un IDE si es necesario para facilitar la comodidad y correr el fichero de `ventana.py` el cual ejecutara la ventana principal y una vez dentro el usuario realizará las diferentes acciones que el desee y cuando pulse los botones de Recomendar o Predecir, en segundo plano se realizara este proceso en el que recomendaremos películas al usuario o predeciremos la valoración que le dará el usuario a una película escogida por el mismo.

Las dependencias de esta práctica son las siguientes:

1. SQLite3: Ya está incluida.
2. De PyQt5 importamos QtCore, QtGui, Qtwidgets: `pip install PyQt5`.
3. CSV: No hace falta instalarla.
4. De Spicy se importa spatial: `pip install spicy`.

## 8. Manual de Usuario

Para poder usar la aplicación necesita tener instalado todo lo descrito en el apartado 7 donde hablamos sobre las dependencias usadas. Una vez todo instalado ya se puede correr el programa, teniendo que ejecutar en este caso `ventana.py` la cual abrirá una pantalla tal que así:

The screenshot shows a Qt application window titled "MainWindow". The interface is split into two horizontal panels. The top panel, titled "Recomendaciones", features a dropdown menu "Selecciona un Usuario (ID)" with "1" selected, a text box "Items ranking" containing "0", and another text box "Umbral de similitud:" containing "0". A "Recomendar!" button is positioned to the right. Below these controls is a table titled "Ranking" with two columns: "ID item" and "Prediccion". The bottom panel contains a "Selecciona un Usuario:" dropdown menu showing "Null", a "Selecciona una película:" dropdown menu, and a "Predicción:" text box. A "Predecir!" button is located to the right of these elements.

Una vez se ha abierto esta pestaña ya puede empezar a usar la aplicación. Hay dos partes a la aplicación siendo la primera la parte de recomendaciones que se encuentra en la parte superior de la pantalla y la parte de predicción en la parte inferior de la pantalla.

## 8.1 Recomendación:

En esta parte el usuario puede elegir los diferentes parámetros que quiere para poder realizar diferentes predicciones. El usuario puede elegir que usuario quiere escoger para realizar la predicción sobre ese mismo, los ítems que quiere que aparezcan en el ranking (cada ítem es una película recomendada) y el umbral de similitud que desea escoger siendo este entre 0 y 1. Una vez ha elegido todos los parámetros, puede pulsar el botón de **Recomendar!**, donde se empezará a realizar el proceso de recomendación acordemente con los datos que el usuario ha introducido, una vez se ha realizado este proceso la tabla de Ranking, se actualizara, mostrando el número de recomendaciones que el usuario ha elegido, ordenándolas de mayor recomendación a menor siendo la recomendación más alta de 5 y la más baja de 0.

## 8.2 Predicción:

En esta parte encontrada en la parte inferior de la pantalla, es la parte en la que el usuario podrá predecir qué valoración le dará un usuario determinado a una película que el elija. Esta parte consta de dos campos que el usuario puede rellenar, siendo el primero el usuario que desea elegir para la predicción y el segundo la película que desea seleccionar. Una vez ha elegido el usuario que desea, se realizara un proceso en el que películas, se actualizarán películas que el usuario no ha visto en el campo de selección de Película. Cuando elija la película que el usuario desea, puede pulsar el botón de predecir el cual comenzará un proceso en el que se predecirá la valoración que le dará ese usuario a la película elegida. Una vez se ha predicho, el resultado podrá ser visible en el apartado de Predicción en la esquina inferior izquierda.

## 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/>