UNIVERSIDAD AUTONÓMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APLICACIÓN WEB PARA VISUALIZAR E INTERACTUAR CON RESULTADOS DE EVALUACIONES DE SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN.

Alejandro Redondo Quintero
Tutor: Alejandro Bellogín Kouki
Ponente: Iván Cantador Gutiérrez

JULIO 2015

DESARROLLO DE UNA APLICACIÓN WEB PARA VISUALIZAR E INTERACTUAR CON RESULTADOS DE EVALUACIONES DE SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

AUTOR: Alejandro Redondo Quintero
TUTOR: Alejandro Bellogín Kouki
PONENTE: Iván Cantador Gutiérrez

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2015

Resumen

En los últimos años, Internet se ha convertido en una herramienta utilizada a diario por millones de personas por diversos motivos (trabajo, viajes, entretenimiento, …). Actualmente, todos los movimientos que se hacen en internet quedan registrados, y toda esa información se utiliza para crear un perfil de usuario sobre cada persona. Estos perfiles son utilizados por las grandes empresas de internet para tener un mayor conocimiento de sus clientes y, de esta manera, poder focalizar la información que se envía a cada uno de ellos de una forma personalizada basándose en sus gustos y necesidades.

Por todo esto, son necesarios algoritmos capaces de llevar a cabo recomendaciones precisas basadas en los datos que se tienen sobre los usuarios. La investigación de este tipo de algoritmos es uno de los campos de la computación en los que más interés se ha puesto en los últimos años.

El objetivo de este proyecto es desarrollar una aplicación web mediante la cual los desarrolladores de algoritmos de recomendación puedan visualizar e interactuar con sus conjuntos de datos. Concretamente, se aplicarán distintos algoritmos de recomendación sobre un conjunto de datos de películas, usuarios y valoraciones y se mostrarán al usuario los resultados mediante gráficas generadas automáticamente. Además, se calcularán métricas de evaluación estadísticas basadas en el error y la desviación media de cada uno de los algoritmos con el fin de mostrar comparaciones del rendimiento entre ellos.

Para todo ello es necesario estudiar el funcionamiento de los tipos de algoritmos de este tipo que existen, planificar el desarrollo de la aplicación web, decidir el tipo de visualizaciones que se mostrarán, elegir las tecnologías que se utilizarán en cada módulo de la misma y, finalmente, ejecutar una batería de pruebas que certifique que el desarrollo de la aplicación ha sido un éxito.

Palabras clave

Recomendador, filtrado colaborativo, evaluación, visualización, aplicación web, conjunto de datos.

Abstract

In the last years, the Internet has become in a tool used everyday by millions of people in the world for very different reasons (for working, travelling, entertainment, ….). Nowadays, every move that people takes on the internet gets registered, and all that information is used to create user profiles about every person. These profiles are used by the big internet companies to improve the knowledge of their clients and, therefore, to focus the information that they send to every one of them and customise it according to their tastes and needs.

For all these reasons, algorithms able to make accurate recommendations based on the users data are very necessary. The research about this kind of algorithms has become one of the most interesting fields of computing science across the last years.

The goal of this project is deleveloping a web application to help the recommendation systems researchers to understand their datasets and interact with them. Specifically, different recommendation algorithms will be applied on a movies, users and ratings dataset and the results will be shown by auto generated graphics. In addition, statistic evaluation metrics based on mean error and deviation will be calculated in order to show performance comparisons between them.

For all these tasks it will be necessary to study the behaviour of recommendation algorithms, plan the web development of the web application, decide the type of visualisations that will be shown, choose the technologies that will be used in every part of it and, finally, execute multiple tests to certify that the application development was a success.

Keywords

Recommender, colaborative filtering, evaluation, visualisation, web application, dataset.

Agradecimientos

En primer lugar, la más importante, gracias a mi madre. Ella es la que ha hecho esto posible. Su esfuerzo y sacrificio diario es lo que han permitido que yo haya llegado a donde estoy. Nunca se lo podré agradecer lo suficiente.
Gracias a mi padre, que siempre me ha ayudado a dar lo mejor de mí, a superarme día a día, a no conformarme y a ser mejor de lo que yo pensaba que podría ser.
Gracias a mi hermano, Sergio, que cada día me da motivos para seguir admirándole. Gracias por ser un espejo en el que querer reflejarme.
Gracias a los profesores que he tenido a lo largo de mi vida y que han dejado huella en mí, ayudándome a elegir el camino que quería recorrer. En especial, gracias a mi tutor Alejandro, que ha sabido ayudarme y guiarme a lo largo de todo este proyecto.
Y dejo para el final a las personas más especiales que me he podido encontrar. Gracias a todos mis amigos. No puedo mencionarlos a todos, pero sé que no hace falta. Gracias a todos por ser tan geniales, por estar siempre ahí, por apoyarme, por tener siempre una palabra de ánimo y por enseñarme siempre el lado bueno de la vida.
A todos gracias, de verdad, muchas gracias.

Índice de contenidos

Índice de tablas

Índice de ilustraciones

1. Introducción

1.1 Motivación

En los últimos años, la capacidad de manejo de datos de internet ha aumentado considerablemente y de forma exponencial. Internet es una fuente inagotable de datos que, de un tiempo a esta parte, se ha ido convirtiendo en una herramienta que también recopila información sobre los usuarios que navegan en la red.
Los usos que las empresas de internet pueden dar a la información que los usuarios proporcionan son prácticamente ilimitados, y van a ir siempre enfocados a su propio beneficio. La venta de datos, los correos basura o la publicidad dirigida son prácticas habituales en el día a día de la navegación por internet.
En una sociedad en la que cada vez más empresas ofrecen sus servicios por medio de internet, es muy importante tener cualidades que destaquen sobre la competencia. Esta nueva necesidad de las empresas, junto con las nuevas posibilidades de internet hace que los algoritmos de recomendación de información se hayan convertido en una parte indispensable de la presencia de las compañías en la web. Aquella empresa que sea capaz de concretar los productos que ofrece a cada usuario de la forma más eficaz y personalizada contará con una ventaja estratégica respecto al resto que se verá reflejada en sus beneficios.
Todo lo analizado anteriormente tiene como consecuencia que el desarrollo de metodologías de recuperación y recomendación de información a gran escala sea uno de los campos de la investigación computacional que más esfuerzos y recursos está ocupando. El objetivo de estos estudios y su investigación es conseguir desarrollar nuevas métricas de recomendación que optimicen el tiempo de ejecución al trabajar con conjuntos de datos de gran tamaño (big data) y que a la vez obtengan mejores resultados en la evaluación.
En este contexto, los desarrollares de sistemas de recomendación se convierten en usuarios potenciales de todos aquellos sistemas que les permitan entender y trabajar mejor y de manera más eficiente con aquellos algoritmos y conjuntos de datos que utilicen para trabajar. Dado que actualmente se ofrece un número reducido de aplicaciones de este tipo, existe un hueco y una necesidad en el mercado en el cual resulta interesante trabajar.

1.2 Objetivos

Tras el análisis de la situación actual del desarrollo e investigación de algoritmos de recomendación se ha concluido que existe la necesidad de herramientas de trabajo que faciliten dicha labor. El objetivo de este proyecto será, por tanto, desarrollar una aplicación web para visualizar conjuntos de datos y resultados de sistemas de recomendación aplicados a los mismos.
La finalidad de la aplicación será ayudar a los desarrolladores e investigadores a comprender mejor los datos que están utilizando, ofreciéndoles la posibilidad de generar distintas gráficas dinámicas que representarán los datos del conjunto en función

de una serie de criterios que el usuario podrá seleccionar mediante filtros. Además, se podrá visualizar los resultados de las recomendaciones así como los de las métricas de evaluación que se apliquen a los algoritmos de recomendación.

1.3 Estructura

El presente documento se divide en siete capítulos principales seguidos de un conjunto de anexos técnicos que complementan la información desarrollada. A continuación se resume el contenido de cada uno de los capítulos:

1. **Introducción.**
   Breve explicación del porqué de la realización del proyecto junto con la definición de los objetivos del mismo.
2. **Estudio del estado del arte.**
   Informe en profundidad sobre los conocimientos teóricos necesarios para la comprensión del proyecto o que ha sido necesario estudiar para llevar a cabo el desarrollo del mismo.
3. **Estudio de las tecnologías a utilizar.**
   Resumen de las herramientas que se ha decidido utilizar o descartar para el desarrollo del proyecto junto con una justificación de la decisión tomada.
4. **Diseño y desarrollo.**
   Documentación sobre el proceso de análisis de la aplicación, la división del proyecto en distintos módulos y la forma en la que se han implementado cada uno de ellos.
5. **Pruebas.**
   Explicación de la batería de pruebas ejecutada sobre cada módulo de la aplicación y sobre el sistema completo para comprobar el correcto funcionamiento del mismo.
6. **Resultados de las pruebas.**
   Estudio de los resultados observados tras la ejecución de las pruebas y conclusiones sacadas de los mismos.
7. **Conclusiones y trabajo futuro.**
   Breve resumen de los conocimientos adquiridos a lo largo del desarrollo del proyecto junto con posibles futuras implementaciones que se han ideado para dotar al sistema de más funcionalidades.

2. Estudio del estado del arte

2.1 Introducción a los sistemas de recomendación

Los sistemas de recomendación se pueden definir como cualquier tecnología que implemente una funcionalidad mediante la cual se sugiera al usuario elementos en los que puede estar interesado basándose en el interés que él mismo ha mostrado en el pasado, bien en otros elementos o según el interés que han mostrado otros usuarios con características similares. Por lo tanto, resulta lógico pensar que para poder implementar un sistema de recomendación óptimo, es necesario tener información relativa tanto al conjunto de elementos que se va a estudiar (género, fecha de creación, país de origen, ...) como al grupo de usuarios con el que se va a trabajar (sexo, edad, profesión, nacionalidad, etc.).
En función de cómo se utilice la información de la que se dispone, se pueden clasificar los sistemas de recomendación principalmente en dos grupos: sistemas de filtrado basado en el contenido y sistemas de filtrado colaborativo [1] [2] [3].

2.1.1 Sistemas de filtrado basado en el contenido

También conocidos como sistemas de filtrado cognitivo [2], los sistemas de filtrado basado en contenido analizan la información que se tiene acerca de los ítems y los usuarios para basar en ella las sugerencias que se realizarán a posteriori.
Este tipo de sistemas de recomendación se pueden entender como una aplicación especial de los sistemas de clasificación, en la que el conjunto de datos de entrenamiento estará formado por los elementos que un determinado usuario ha clasificado previamente. Los elementos de este conjunto estarán definidos por un modelo consistente en un conjunto de atributos y una clase, que será la que refleje el resultado de la clasificación. Formalmente, un ítem se describe como un vector $X = (x_1, x_2, \ldots, x_n)$ de n elementos que pueden tener valor nominal, binario o numérico, y que pueden estar basados en información tanto del ítem como de los usuarios [3] [4].
La tarea de un método de clasificación de este tipo consiste en implementar una función capaz de clasificar cualquier elemento del conjunto de datos tras haberse entrenado con el grupo de ítems que el usuario ya había valorado. Dependiendo del tipo de rating con el que se esté trabajando puede implementarse una función que devuelva valores binarios (si se desea clasificar solo mediante valoraciones positivas o negativas) o valores numéricos (por ejemplo, un número entre 1 y 5).
En el ejemplo de la tabla 1, se pueden ver las valoraciones que el usuario Leire ha dado a una serie de blogs. Teniendo en cuenta que ha valorado positivamente los blogs “Linux” y “Programación en la nube”, y sabiendo que otros usuarios a los que les han

gustado esos blogs también ha valorado positivamente el blog “Código abierto”, éste sería la mejor recomendación para el usuario Leire.

Blogs	Leire
Linux	10
Código abierto	-
Programación en la nube	9
Java	-
HTML	1

Contenido similar
Linux
Código abierto
Programación en la nube
…
…

Tabla 1: Ejemplo de tabla de filtrado basado en contenido

2.1.2 Sistemas de filtrado colaborativo

Los sistemas de filtrado colaborativo basan las predicciones y recomendaciones para un usuario concreto en el comportamiento que han tenido los demás usuarios del sistema. Este tipo de algoritmos asume que las opiniones de otros usuarios pueden ser seleccionadas y agregadas con el fin de llevar a cabo una predicción aproximada de las preferencias del usuario activo. Además, se basan en la idea de que si dos usuarios coincidieron en la valoración de algún elemento en el pasado, es probable que vuelvan a coincidir en el futuro.

Una forma de entender estos sistemas de filtrado es visualizar una matriz formada por las valoraciones de los ítems (colocadas en las filas) y los usuarios que han realizado dichas valoraciones (colocados en las columnas). Por ejemplo, en la tabla 2 se puede observar una matriz en la que las valoraciones se miden como número de entradas leídas para cada blog [4].

Blogs	José	Leire	Iván	Víctor
Linux	13	3	11	-
Código abierto	10	-	-	3
Programación en la nube	6	1	9	-
Java	-	6	-	9
HTML	-	7	1	8

Tabla 2: Ejemplo de matriz de filtrado colaborativo.

Estos algoritmos de filtrado se pueden clasificar a su vez en dos categorías distintas [5] [6]:

- Métodos basados en memoria: parte de la matriz de ratings explicada anteriormente para realizar predicciones entre el usuario e ítem que se están

estudiando y el resto de la matriz. Uno de los métodos más populares de este grupo es el basado en vecinos próximos.

- Métodos basados en modelos: a partir de la matriz de datos descrita, se buscan patrones mediante la aplicación de algoritmos de aprendizaje automático que permitan el desarrollo de modelos capaces de predecir valoraciones. Entre los métodos más extendidos de este tipo se encuentran los clasificadores bayesianos o los clasificadores de vecinos próximos basados en clustering.

A partir de los datos de la tabla 2, aplicando algoritmos de clustering basados en vecinos próximos, se pueden juntar a José e Iván en un grupo y a Leire y Víctor en otro. En el primer grupo, se ve que José ha leído muchas entradas del blog de "Código abierto" pero Iván ninguna, por lo que una recomendación para Iván sería ese blog. Para José no se podrán hacer recomendaciones porque las diferencias no son destacables. En el grupo 2, Leire ha leído tres entradas del blog de "Linux" y Víctor ninguna, por lo que es una recomendación buena para él. Ocurre lo mismo pero a la inversa para el blog de "Código abierto", por lo que éste sería una buena recomendación para Leire. En la siguiente tabla puede observarse resaltado en ver las recomendaciones del grupo 1 y en azul las del grupo 2.

Blogs	**José**	**Leire**	**Iván**	**Víctor**
Linux	13	3	11	-
Código abierto	10	-	-	3
Programación en la nube	6	1	9	-
Java	-	6	-	9
HTML	-	7	1	8
Cluster	1	2	1	2

Tabla 3: Ejemplo de matriz de filtrado colaborativo con recomendaciones.

2.1.3 Sistemas híbridos

Como su nombre indica, este grupo engloba aquellos sistemas que toma características de los dos anteriores. Se trata de un sistema que obtiene información de cada ítem aplicando algoritmos basados en el contenido hasta construir un perfil de usuario que estará formado por un conjunto de valores correspondientes a la importancia que asigna cada usuario a cada atributo de los ítems. Con esto, se puede construir una matriz de pesos sobre la que aplicar los métodos de filtrado colaborativo y relacionar así usuarios que pueden compartir gustos por los mismos ítems [4].

2.2 Métodos de evaluación

Además de la aplicación y visualización de sistemas de recomendación sobre un conjunto de datos, otro objetivo que se busca con el desarrollo de la aplicación es proporcionar una herramienta útil y sencilla que permita evaluar los distintos algoritmos que se pueden aplicar. De esta manera se podrá comparar el rendimiento de cada uno de los algoritmos con el fin de discernir cuál es el que mejor resultados consigue para un determinado conjunto de datos.

El procedimiento que se debe seguir para llevar a cabo la evaluación de un algoritmo de recomendación es similar al que se sigue en el caso de otros métodos de recuperación de información. Los algoritmos deben estar compuestos de una función de entrenamiento, en la que desarrollan el modelo que deberán seguir para llevar a cabo las recomendaciones a partir de unos datos conocidos, y una función de test, en la que se llevarán a cabo las propias recomendaciones. Por tanto, los pasos a seguir en el proceso de evaluación de un algoritmo de recomendación [7]:

1. Dividir el conjunto de datos de muestra (típicamente relación entre usuarios e ítems a través de ratings) en dos conjuntos complementarios. Cada uno de estos conjuntos serán nombrados “datos de entrenamiento” y “datos de test”. No es necesario que los conjuntos tengan el mismo número de datos, de hecho, es conveniente que haya más datos en subconjunto de entrenamiento; una proporción de 80-20 suele ser apropiada.
2. Se entrena el algoritmo implementado con el conjunto de datos de entrenamiento.
3. Una vez el algoritmo ha terminado de ejecutar su función de entrenamiento, se pide que genere una predicción para cada elemento del conjunto de datos de test.
4. Se comparan los resultados de las predicciones con los datos reales que hay en el conjunto de datos de test. En función del tipo de algoritmo que se esté evaluando, se podrá medir la calidad del mismo aplicando una serie de métricas u otras.

Seleccionar el método de comparación para la evaluación de los sistemas de recomendación no es una tarea trivial. La variedad de tipos de algoritmos que se pueden encontrar hace que sea difícil establecer una métrica universal que aplicar a todos los algoritmos. Por ejemplo, no se puede evaluar de la misma forma un algoritmo desarrollado para predecir la valoración que un usuario hará de un ítem que un algoritmo cuya prioridad es no realizar recomendaciones erróneas o uno que se concentra en optimizar el ranking de ítems presentado al usuario.

En general, se pueden separar las formas de evaluación de algoritmos en dos tipos: métodos estadísticos y métricas de decisión [7].

2.2.1 Métodos estadísticos.

Utilizados para medir la precisión de sistemas de predicción de ratings numéricos, como por ejemplo una valoración de 1 a 10. Existen diferentes métricas que se pueden calcular a partir de los resultados de la fase de test. Las más utilizadas son las siguientes:

- MAE: el Error Medio Absoluto (Mean Absolute Error), también llamado desviación absoluta, mide la desviación entre las recomendaciones predichas y los valores reales, por lo tanto, un menor MAE implica un mejor algoritmo.

$$\left(\frac{1}{n}\right)\sum_{u,i} \mid p_{u,i} - r_{u,i} \mid$$

  Donde:

  - $p_{u,i}$ es la predicción del usuario u sobre el ítem i.
  - $r_{u,i}$ es el rating del usuario u sobre el ítem i.
  - n es el número de pares de usuario-ítem en el test.

  Como se puede deducir observando la fórmula, el MAE se encuentra en el mismo rango de valores que los ratings. Es una métrica útil para entender la calidad de un algoritmo de recomendación sobre un conjunto de datos, pero no se puede utilizar para comparar los resultados de un mismo algoritmo sobre conjuntos de datos si éstos tienen rangos de valores diferentes.

- NMAE: se trata del Error Medio Absoluto Normalizado (Normalized Mean Absolute Error). Es una métrica similar a la anterior pero se encarga de normalizar el error dividiéndolo entre el rango de posibles ratings.

$$\left(\frac{1}{n(r_{high} - r_{low})}\right)\sum_{u,i} \mid p_{u,i} - r_{u,i} \mid$$

  Donde:

  - $p_{u,i}$ es la predicción del usuario u sobre el ítem i.
  - $r_{u,i}$ es el rating del usuario u sobre el ítem i.
  - n es el número de pares usuario-ítem en el test.
  - r_{high} es el valor máximo que puede tener un rating.
  - r_{low} es el valor mínimo que puede tener un rating.

  Al normalizar los resultados hace que el rango de valores que puede tener sea ente 0 y 1, por lo que resulta más difícil interpretar los resultados en la escala de ratings original. No obstante, está claro que en esta métrica un resultado menor implica también un algoritmo mejor implementado. La principal ventaja de esta métrica respecto al MAE, es que sí se puede utilizar para comparar el

rendimiento de un algoritmo sobre distintos conjuntos de datos, ya que el rango de valores será siempre de 0 a 1.

- RMSE: el Error Cuadrático Medio (Root Mean Squared Error) es una métrica que penaliza especialmente los errores más grandes. De esta forma, en una escala de 1 a 5, un fallo por 2 puntos queda más reflejado en el resultado que 8 fallos de 0.25 puntos.

$$\sqrt{\left(\frac{1}{n}\right)\sum_{u,i}(p_{u,i}-r_{u,i})^2}$$

Donde:

- $p_{u,i}$ es la predicción del usuario u sobre el ítem i.
- $r_{u,i}$ es el rating del usuario u sobre el ítem i.
- n es el número de pares de usuario-ítem en el test.

Como se puede observar, se calcula de forma similar al MAE pero eleva el error al cuadrado antes de sumarlo. También da un resultado en el mismo rango de valores de los ratings y puede ser normalizado, al igual que el NMAE, multiplicándolo por $1/(r_{high} - r_{low})$.

2.2.2 Métricas de decisión.

Tratan de evaluar cómo de efectivo es un sistema de predicción en su labor de ofrecer al usuario elementos que puedan resultar adecuados para él, es decir, calculan con qué frecuenta el sistema de recomendación efectúa recomendaciones correctas. Para ello, es necesario que el sistema de valoraciones del conjunto de datos sea binario (los ítem agradan o no a los usuarios), ya que los métodos estadísticos vistos en el apartado anterior no tendrían mucho sentido en este caso. El problema de aplicar este tipo de métricas es que los resultados pueden variar dependiendo de los datos que haya en los conjuntos de entrenamiento y de test, especialmente del porcentaje de elementos relevantes que el usuario haya votado. La más conocida de estas métricas recibe el nombre de “Precision and Recall”, donde “precision” hace referencia a la probabilidad de que un elemento seleccionado sea relevante y “recall” es la probabilidad de que sea seleccionado un elemento relevante. De cara al usuario que estudia los resultados de la evaluación, esta métrica es la más intuitiva, ya que queda claro que una precisión del 90% indica que 9 de cada 10 predicciones son correctas, mientras que afirmaciones como esa no quedan tan claras obteniendo valores de error cuadrático medio [8].

2.3 Aproximación a algunos algoritmos de recomendación.

En este momento ya conocemos el funcionamiento general de los algoritmos de recomendación y sabemos las bases sobre las que están desarrollados. A continuación, procederemos a profundizar un poco más en el estudio de ciertos algoritmos concretos con el fin de ser conscientes de las diferencias que existen entre ellos.
Dado que el objetivo de este apartado es comprender mejor los resultados que se obtendrán al ejecutar los métodos de recomendación sobre los datos, nos centraremos en estudiar únicamente los algoritmos que se han decidido visualizar desde la aplicación.

2.3.1 Algoritmo de vecinos próximos basado en ítems

Es conocido tradicionalmente como algoritmo "ítem KNN" por las siglas de parte de su nombre en inglés (ítem-based K-Nearest Neighbor). Se basa en la filosofía de que para determinar el rating (r) de un usuario (u) sobre un ítem (i), basta con encontrar los K ítems más "similares" a i que el usuario u haya valorado y calcular r a partir de los ratings que dicho usuario asignó a los ítems. Aunque parece una idea sencilla, la complicación radica en definir una función capaz de calcular la similitud que un ítem guarda con otro, o lo que es lo mismo, definir la métrica de similitud. Existen varias fórmulas matemáticas que se pueden usar para calcular la similitud de dos ítems. La más utilizada es la llamada Similitud basada en el Coseno (Cosine-based Similarity), también conocida como Similitud basada en Vectores (Vector-based Similarity) ya que considera los dos ítems y sus ratings como vectores y define la similitud entre ellos como el ángulo que forman. Aplica, por lo tanto, una función de geometría básica [9] [10]:

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

donde:

- Sim(i, j) es la similitud entre los ítems en forma de vector i y j.

Como ya se ha mencionado, esta no es la única función que puede utilizarse para establecer la similitud entre dos elementos del conjunto de datos. Otras métricas de similitud que pueden aplicarse son Similitud basada en Correlación de Pearson (Pearson Correlation-based Similarity), que mide la desviación que los ratings de dos usuarios con un par de ítems en común provocan sobre los ratings medios de dichos ítems, o la Similitud basada en el Coseno Ajustada (Adjusted Cosine Similarity), que modifica la Similitud basada en Vectores con el fin de evitar las consecuencias que pueden tener las diferentes pautas de los usuarios a la hora de valorar (ya que puede haber usuarios que entiendan de forma diferente el significado de cada puntuación).

2.3.2 Algoritmo de vecinos próximos basado en usuarios

Su nombre original es User-based K-Nearest Neighbor. La funcionalidad de este algoritmo se basa en la misma filosofía que el anterior, aplicar una métrica de similitud para relacionar dos elementos. En este caso para calcular el rating (r) que un usuario (u) asignará a un ítem (i), se buscan los usuarios más similares a u que hayan valorado a i. La predicción se calculará a partir de los ratings que los usuarios similares hayan asignado a i. Las funciones de similitud que se pueden aplicar en este algoritmo son equivalentes a las que se utilizan en el basado en ítems.

1.3.3 Algoritmo de factorización de matrices

Este algoritmo define un conjunto de factores en los que se pueden descomponer todos los ítems y usuarios, pudiendo representarlos así en forma de vector. El rating que un ítem recibirá de un usuario se podrá calcular realizando el producto entre los vectores de cada uno [11] [7].

$$\hat{r}(u,i) = q_i \cdot p_u$$

El objetivo del algoritmo es minimizar el error cuadrático medio, esto es, la cantidad de errores que comete en la predicción de ratings sin que se produzca sobreajuste.

2.3.4 Baseline con usuarios e ítems

Se trata de unos de los algoritmos más sencillos para recomendación. Predice un rating para un usuario y un ítem aplicando una fórmula sobre la media de los ratings realizados por ese usuario y la valoración media de ese rating.

3. Estudio de las tecnologías a utilizar

3.1 Tecnologías a utilizar en el desarrollo de la aplicación.

En la última década, se ha producido una enorme evolución y proliferación de sitios web que ofrecen a los usuarios herramientas y funcionalidades concretas que les permiten llevar a cabo tanto actividades laborales como de ocio de una manera rápida y sencilla. Este tipo de sitios son conocidos como aplicaciones web, ya que se basan en software codificado en lenguajes capaces de ser interpretados por un navegador web común. Algunos de los motivos principales de la popularización de esta clase de aplicaciones son:

- No hace falta instalar software adicional en la máquina del cliente más allá del navegador web que utilice.
- No es necesario adaptar el código fuente de la aplicación para poder utilizarla en distintos sistemas operativos.
- La actualización y el mantenimiento de esta clase de aplicaciones resultan mucho más sencillos y rápidos que en el caso de las aplicaciones "tradicionales".

Una consecuencia directa de la enorme magnitud que está alcanzando esta clase de aplicaciones, es que se han desarrollado una gran cantidad de tecnologías, librerías y herramientas que permiten a los programadores elegir los métodos de trabajo que mejor se adapten a las necesidades del sitio web que decidan crear.
En este contexto superpoblado de alternativas para el desarrollo de aplicaciones web, se ha llevado un análisis de las más populares a día de hoy para seleccionar la más óptima para implementar el sistema que aquí se plantea [12].

3.1.1 Ruby on Rails (RoR)

Se trata de un framework para desarrollo web ágil basado en el lenguaje de programación orientada a objetos Ruby bajo licencia MIT [13] [14] [15]. El principal objetivo de esta clase de framework es ahorrar al desarrollador tiempo de programación proporcionándole un conjunto de herramientas que le permiten simplificar tareas comunes y repetitivas. Las principales características de Ruby on Rails son:

- Sigue el patrón de diseño Modelo-Vista-Controlador (MVC), que se basa en la separación de los datos y la lógica de la aplicación [15]. Entiende los "modelos" como las clases de las que se va a componer la aplicación y que se corresponden con tablas en la base de datos que manejará la misma. La vista es el conjunto de código que formará la interfaz del sitio web, generalmente HTML. Por último,

los controladores serán los encargados de manejar la interactuación de los usuarios con la aplicación.

- Es de código abierto, lo que permite a los desarrolladores de todo el mundo crear librerías (en este caso conocidas como "Ruby gems") para mejorar las funcionalidades las aplicaciones.
- Soporta multitud de gestores de bases de datos SQL. De hecho, se puede acceder a la base de datos de forma totalmente abstracta para el programador, sin necesidad de escribir consultas en código SQL, aunque también están permitidas. Además, ya que a lo largo del desarrollo de una aplicación pueden cambiar las necesidades de la misma, el framework está preparado para que la migración de un gestor de bases de datos a otro no suponga un quebradero de cabeza para el programador.

3.1.2 PHP

Uno de los lenguajes de programación por excelencia cuando se habla de programación web es PHP. Se trata de un lenguaje que inicialmente fue concebido para ejecutarse del lado del servidor y generar contenido de forma dinámica que luego es enviado al navegador del cliente para ser interpretado [16]. Algunas de las ventajas de su utilización son:

- Adquirió gran popularidad cuando fue publicado debido a que fue uno de los primeros lenguajes que se podían incorporar en el documento HTML. Esto, junto al largo recorrido que tiene el lenguaje desde su lanzamiento, han hecho que sea uno de los más extendidos en programación web, lo que ha ayudado a ampliar y perfeccionar la documentación que se puede encontrar en su sitio web oficial.
- Posee un gran parecido con otros lenguajes de programación estructurada como C, siendo relativamente fácil de aprender para los desarrolladores que ya estén familiarizados con este tipo de programación.
- La popularización del lenguaje ha hecho que se encuentre disponible para la mayoría de servidores y sistemas operativos.
- Permite trabajar con los principales motores de bases de datos actuales, tanto SQL como NoSQL, lo cual es un indicativo de que con el paso del tiempo ha ido evolucionando y adaptándose a las necesidades de los desarrolladores.

3.1.3 Django (Python)

Django es un framework de alto nivel para desarrollo web escrito en Python, un lenguaje de programación interpretado diseñado para tener una sintaxis que favorezca un código

legible. Al igual que en el caso de Ruby on Rails, fomenta un desarrollo rápido y trata de ofrecer al programador soluciones sencillas que le permitan centrarse en la lógica de su aplicación por encima de todo [17] [18]. Algunas de las características más destacables de Django son:

- La prioridad cuando fue diseñado era proporcionar a los desarrolladores un conjunto de herramientas que les permitieran desarrollar aplicaciones web complejas en el menor tiempo posible. Por lo tanto, se puede decir que es un framework orientado a la productividad.
- Aunque se recomienda el uso de PostgreSQL como motor de base de datos, Django es compatible con otros motores basados en SQL como MySQL y SQLite.
- Uno de los objetivos de Django es ayudar a los programadores a evitar cometer fallos de seguridad que pongan a prueba la integridad de las aplicaciones que desarrollen.
- Las aplicaciones desarrolladas con este framework se optimizan al máximo, siendo muy recomendable su uso en el desarrollo de sitios web en los que se prevé abundante tráfico.

3.1.4 Tecnología elegida para el desarrollo

Tras analizar los lenguajes y frameworks explicados anteriormente y valorar las ventajas y desventajas que presentan cada uno de ellos, se ha tomado la decisión de llevar a cabo el desarrollo de la aplicación web en Ruby on Rails. Se ha considerado que es el sistema que mejor se adapta a las necesidades del tipo de aplicación que se quiere desarrollar, valorando especialmente las facilidades que ofrece en las tareas de programación respecto a otros sistemas. Además, es una tecnología que se encuentra en auge desde hace varios años, lo que garantiza que no habrá problemas en el futuro cuando se precise llevar a cabo mejoras o tareas de mantenimiento en la aplicación.

3.2 Almacenamiento de información.

A la hora de almacenar los datos que serán mostrados a través de la aplicación web se pueden tomar distintas decisiones que afectarán en el futuro al rendimiento de la misma. Se puede optar por una decisión tan simple como organizar la información en ficheros almacenados en el servidor y leer o escribir en ellos según se necesite o llevar a cabo un diseño complejo basado en bases de datos no relacionales con el fin de obtener un mayor rendimiento en el almacenaje y extracción de grandes volúmenes de datos. Dado que los requisitos de la aplicación que se va a desarrollar no exigen un volumen de datos excesivo ni la capacidad de soportar un tráfico muy elevado, un sistema de bases de datos relacionales cubrirá las necesidades del sitio web.

Teniendo claro qué tipo de almacenamiento de datos se va a utilizar, enfrentarse a la elección del motor de base de datos relacional que se utilizará en la aplicación supone

un reto teniendo en cuenta la cantidad de opciones que se ofrecen. El hecho de haber decidido que se utilizará Ruby on Rails para el desarrollo de la aplicación acota ligeramente la búsqueda ya que no tiene compatibilidad con todos los gestores de bases de datos existentes, lo cual nos limita ligeramente las posibilidades a la hora de decidir. A continuación se hace un breve análisis de los principales motores de bases de datos soportados por RoR que se han tenido en cuenta como posibles soluciones para el desarrollo de la aplicación web.

3.2.1 MySQL

Probablemente el gestor de bases de datos más popular que existe. Se caracteriza por ser una tecnología bastante óptima para sitios web con tráfico alto. Ofrece multitud de tipos de datos, por lo que se adapta bien a las necesidades de cada aplicación [19].

3.2.2 SQLite

Un gestor de bases de datos sencillo. Básicamente almacena toda la información en un fichero con un formato concreto, lo cual resulta cómodo si fuese necesario migrar la aplicación. Está pensado principalmente para desarrollo y testeo de aplicaciones. No es un gestor óptimo para entornos de producción en los que se espere concurrencia de usuarios accediendo a la vez [20] [21].

3.2.3 PostgreSQL

Se trata de un motor de bases de datos enfocado a la fiabilidad e integridad de la información. Debido al largo recorrido de esta tecnología, cuenta con una comunidad de desarrolladores muy amplia que ha creado un gran conjunto de herramientas para aumentar sus funcionalidades.

3.2.4 Tecnología elegida para el almacenamiento de datos

Como se ha mencionado anteriormente, la elección de Ruby on Rails como sistema para desarrollar la aplicación web condiciona también la elección del sistema de almacenamiento de datos. Tras valorar los pros y contras de cada motor de bases de datos analizado, se ha decidido utilizar SQLite durante la fase de desarrollo y pruebas para almacenar la información que se va a representar. El principal motivo de esto es la facilidad con la que se podrá migrar la base de datos en caso de tener que trasladar la aplicación de un sistema local a otro. No obstante, como se ha mencionado anteriormente, este motor no está pensado para ejecutar conexiones concurrentes, por lo que cuando se considere que el proyecto está listo para pasar a la fase de producción, la mejor opción será migrar los datos a una base de datos basada en MySQL cuando se considere que está lista para pasar a la fase de producción. Este proceso de migración no precisará de muchos cambios en el código de la aplicación gracias al conjunto de herramientas que proporciona Ruby on Rails para permitir al desarrollador abstraerse

del motor de base de datos y trabajar en el controlador de la aplicación de forma ajena a cómo se almacenen los datos.

3.3 Librería para llevar a cabo las recomendaciones.

Dado que una de las partes más importantes del proyecto que se está desarrollando se basa en la aplicación de métodos de recomendación, es importante seleccionar la librería que mejor se adapte a nuestras necesidades para obtener las funcionalidades deseadas. Llegados a este punto, ya se ha elegido la tecnología principal del desarrollo de la aplicación (RoR) y el gestor de bases de datos que se va a utilizar (SQLite). Estas elecciones pueden condicionar o limitar la cantidad de tecnologías que se pueden introducir en la aplicación para llevar a cabo las recomendaciones. En cualquier caso, se realizará un estudio de las mejores alternativas que se pueden implementar para aplicar los algoritmos valorando especialmente su compatibilidad con las herramientas ya seleccionadas.

3.3.1 MyMediaLite

Se trata de una librería escrita en C# que implementa distintos algoritmos de recomendación y que permite trabajar con los dos principales escenarios posibles de filtrado colaborativo explicados previamente: predicción de ratings y predicción de ítems [22] [23]. Las principales ventajas que ofrece esta librería frente a otras son:

- Implementa un enorme número de algoritmos de recomendación, con la posibilidad de personalizar los parámetros de aquellos que lo permitan modificando la sintaxis de la llamada al método correspondiente.
- Tanto en el caso de la predicción de ratings como en la de ítems, la librería ofrece un conjunto de métodos para evaluar la efectividad del algoritmo. Dichos métodos consisten en el cálculo de diferentes métricas estandarizadas como el error absoluto medio o la desviación cuadrática media.
- Contiene métodos para leer directamente los ficheros con los datos de entrada en un formato concreto.
- Aunque está escrito en C# y dirigido para plataformas basadas en .Net, puede utilizarse en otras arquitecturas por medio de aplicaciones externas como Mono [24] o IronRuby [25].

3.3.2 Recommendable

Recommendable es un gem de Ruby (nombre que reciben los paquetes, extensiones y librerías desarrollados en un formato estándar y autocontenido para este lenguaje) [26]. Las principales características de esta tecnología son:

- Las recomendaciones que hace la librería están basadas en un algoritmo implementado por el propio desarrollador y no da la posibilidad de utilizar

distintos algoritmos. Dicho algoritmo debe ser utilizado con un sistema de valoraciones positivas o negativas, sin posibilidad de asignar una calificación a un ítem.

- Se trata de un paquete escrito totalmente en Ruby, por lo que la integración con el framework en el que se va a desarrollar la aplicación sería perfecta.
- Se trata de un proyecto de código abierto, lanzado por primera vez hace 3 años y que ha ido actualizándose desde entonces. La última versión fue lanzada a fecha de 12 de junio de 2015, por lo que sigue recibiendo mantenimiento y mejoras. Su larga trayectoria hace que se hayan efectuado una gran cantidad de descargas de la librería, por lo que es un sistema que ha sido probado en profundidad por la comunidad.

3.3.3 Recommendify

Se trata de otro gem de Ruby, aunque está escrito parcialmente en C nativo [27]. Algunas de las características de este paquete son:

- Solo implementa un algoritmo y éste se basa en valoraciones positivas o negativas en vez de asignar un rating numérico a cada ítem.
- La primera versión fue lanzada hace 3 años y se fue actualizando ligeramente pero desde hace un año no hay actividad por parte del desarrollador, por lo que no se puede considerar que la librería esté actualizada. No obstante, ha recibido bastantes peticiones de descarga por lo que tiene bastante feedback por parte de los desarrolladores.

3.3.4 Librería para llevar a cabo las recomendaciones

La tecnología elegida para aplicar los algoritmos de recomendación sobre el conjunto de datos es MyMediaLite. Se ha visto que esta librería tiene un gran número de ventajas sobre el resto de las estudiadas. El gran número de algoritmos de recomendación que implementa, junto con la posibilidad de evaluar y comparar distintos recomendadores entre ellos son dos de los factores que más se han tenido en cuenta para elegir esta librería.

Dado que no se trata de una extensión nativa de Ruby, sino que habrá que utilizar una tecnología aparte para ejecutar los métodos que proporciona la librería; por ello, no será posible calcular los resultados de las recomendaciones de manera dinámica a medida que sean solicitados por el usuario. Por este motivo, se ha tomado la decisión de precalcular dicha información, y escribirla en ficheros de salida, que serán leídos y almacenados en la base de datos. De esta manera, se gana en eficiencia a la hora de mostrar la información a los usuarios a través de la web. Esta forma de actuar, además, posibilita la opción de poder mostrar en la aplicación resultados de algoritmos de recomendación propios, ya que bastaría con ejecutar el código deseado (incluso usando otra librería) sobre el conjunto de datos y guardar los resultados en la base de datos.

3.4 Otras tecnologías utilizadas

Para la construcción de la aplicación web será necesario utilizar otras tecnologías que funcionen junto con las descritas anteriormente. A continuación se procederá a realizar una descripción del resto de tecnologías utilizadas sin que sea necesario compararlas con otras que ofrezcan la misma funcionalidad, ya que en estos casos se ha tenido claro desde el primer momento que eran la mejor elección posible.

3.4.1 HTML

El HyperText Markup Language (lenguaje de marcas de hipertexto) es el lenguaje de programación por excelencia para la elaboración de páginas web. Es el encargado de definir la estructura básica y el contenido de los sitios web. Su funcionamiento se basa en el envío de los ficheros con el código HTML del servidor a los clientes, cuyos navegadores serán los encargados de interpretar dicho código y representarlo. A día de hoy, prácticamente la totalidad de los navegadores estándar del mundo son compatibles con este lenguaje. Es un estándar a cargo de la W3C [28].

3.4.2 CSS

Corresponden en inglés a las siglas de Cascading StyleSheet (Hojas de Estilo en Cascada). Se trata de un estándar que define la manera en que se deben presentar los componentes de una web declarados en el código HTML. Está permitido definir el la forma en que se deben representar los elementos dentro del propio fichero HTML en el que se declaran o bien hacerlo en un fichero separado que solo contendría código CSS. Esta última forma de distribuir el código es el paradigma de la idea que había tras el surgimiento de este estándar: separar la presentación de los elementos (su estilo) de la declaración de los mismos [29].

3.4.3 JavaScript

Se trata de un lenguaje de programación orientado a objetos que se ejecuta del lado del cliente, principalmente con el fin de llevar a cabo mejoras en la interfaz de usuario y la creación de páginas web dinámicas. Se ha desarrollado una gran cantidad de frameworks y librerías basadas en este lenguaje de programación, que permiten dar a los programadores una mayor facilidad a la hora de aprovechar las capacidades del lenguaje. Concretamente, en el desarrollo de este proyecto, se han utilizado dos de estas librerías, las cuales analizaremos a continuación [30].

3.4.3.1 jQuery

Probablemente sea la librería más extendida de JavaScript. Está formada por una serie de clases y funciones que facilitan el manejo de eventos en los elementos HTML y que permiten ejecutar animaciones en los mismos. Además, ofrece una API para realizar peticiones HTTP asíncronas mediante AJAX que resulta muy útil para dar dinamismo a la interfaz de la aplicación, permitiendo recuperar datos almacenados en el servidor sin necesidad de recargar la página en la que el usuario está navegando [31] [32].

3.4.3.2 D3js

Se trata de una librería diseñada para la representación de información. Ofrece un conjunto de métodos que permiten enlazar una serie de datos con elementos DOM y llevar a cabo transformaciones en ellos. Las funciones de esta librería se encargan de generar y/o alterar código HTML y CSS mediante la creación de SVG (Scalable Vector Graphics). La principal característica de esta librería, lo que la diferencia en gran medidas de muchas de las que han surgido a partir de ella, es que se centra en proporcionar a los desarrolladores un conjunto de funcionalidades que les permiten manipular elementos HTML basados en conjuntos de datos. Es decir, no ofrece un conjunto de opciones para representar los datos de distinta manera, sino que proporciona las herramientas para que sea el desarrollador quien construya las representaciones que mejor se adapten a sus necesidades. [33] [34]

3.4.4 IronRuby

Se trata de una tecnología que permite combinar librerías de Ruby y los lenguajes del framework de .Net. Se ha precisado de su utilización a lo largo del proyecto para desarrollar scripts en Ruby que llamasen a las funciones de la librería de MyMediaLite que, como se ha explicado anteriormente, está escrita en C#. [25]

3.5 Conjunto de datos utilizado.

Debido al tipo de sistema web que se pretende implementar, es necesario utilizar un conjunto de datos con información apropiada para generar recomendaciones. Dichos datos deben contener referencias a un conjunto de usuarios, un conjunto de ítems y una serie de valoraciones (ratings) que los usuarios habrían hecho de dichos ítems. Sería recomendable que, además, los datos contuvieran información adicional sobre los ítems y sobre los usuarios, con el fin de poder dotar a la aplicación con más funcionalidades en la visualización de dichos datos.

Desde el momento en que se planteó la aplicación se ideó a partir del conjunto de datos proporcionado por Grouplens, un laboratorio de investigación de la universidad de Minnesota especializado en, entre otras cosas, sistemas de recomendación. Una de las

actividades de este grupo de trabajo es la recopilación de distintos conjuntos de datos (datasets) y publicación de los mismos. Uno de los datasets más comunes que han publicado es el que utiliza la web http://movielens.org/, un proyecto sin ánimo de lucro que ofrece la posibilidad a los usuarios de crear un perfil, valorar películas y calcular recomendaciones basadas en los gustos de cada usuario. [9] [35]
Las principales ventajas que ofrece este conjunto de datos (dataset) respecto a otros similares son:

- Aparte de tratarse de uno de los sitios más populares entre los desarrolladores que buscan datasets de este tipo, se trata de una fuente de gran prestigio, por lo que los datos que se van a obtener son de una gran calidad y habrán sido altamente testeados.
- El formato en el que vienen los datos es simple y está bastante bien documentado. Además, más allá de todo eso, destaca el hecho de que la librería de recomendaciones seleccionada (MyMediaLite) espera los ficheros de entrada en ese formato.
- Junto con la información del dataset, se incluyen un conjunto de "splits" (fragmentos del total de los datos preparados para ser utilizados como conjuntos de entrenamiento y test) que también se presentan en el formato de entrada que esperan recibir las funciones de lectura de ficheros de MyMediaLite. Estos splits son estándar dentro del área de la recomendación.
- Los ratings contenidos en este dataset son números enteros que van de 1 a 5 (siendo 1 lo peor y 5 lo mejor) lo cual nos permitirá usar las funciones de predicción de ratings que ofrece MyMediaLite. No obstante, se considera favorable el hecho de que los ratings vengan en este formato porque se podrían adaptar los datos para convertirlos en un sistema de puntuaciones binario (fijando un umbral a partir del cual se considera un rating positivo o negativo) y poder utilizar así en un momento dado las funciones de recomendación de ítems, ampliando así las funcionalidades de la aplicación.

En la web de Grouplens se han publicado varios conjuntos de datos, cada uno de los cuales abarca un período de tiempo distinto en función del tamaño de los mismos. En este caso, se ha seleccionado el dataset "MovieLens 100k", un conjunto de datos formado por 100000 ratings de 943 usuarios y 1682 películas, que fue publicado en abril de 1998. El dataset se puede descargar desde la web de Grouplens y contiene los siguientes ficheros:

- u.data: fichero principal del conjunto de datos. Contiene una relación de películas y usuarios a través de los ratings que éstos han realizado. Antes de la publicación del dataset se llevó a cabo una limpieza de los datos para eliminar aquellos usuarios que no hubieran valorado un mínimo de 20 películas. El fichero se compone básicamente de cuatro columnas que contienen el identificador del

usuario, el identificador de la película, un valor numérico entre 1 y 5 que indica el rating y el timestamp del rating.

- u.info: fichero de texto plano con el número de usuarios, ratings e ítems contenidos en u.data.
- u.item: información sobre los ítems del dataset. Además del identificador numérico de la película y el título de la misma, contiene información adicional como la fecha de estreno, fecha de salida en video, la URL de la película en IMDb [36] o los géneros a los que pertenece.
- u.genre: listado de los géneros a los que puede pertenecer una película junto con el identificador numérico de cada género. Cada película puede pertenecer a más de un género.
- u.user: información demográfica sobre los usuarios incluidos en el fichero u.data. Dicha información consiste en: identificador numérico del usuario, edad, género, ocupación y código postal.
- u.occupation: lista de las posibles ocupaciones disponibles para cada usuario.
- Ficheros de entrenamiento y test tomados de forma aleatoria: son cinco parejas de ficheros con nombre “un.base” y “un.test” donde “n” es un número de 1 a 5. Básicamente consisten en extracciones del fichero u.data repartidas tal que el 80% de los datos están en el conjunto de entrenamiento y el 20% restante en el test.
- Ficheros de entrenamiento y test tomados según un patrón: son dos parejas de ficheros (ua.base, ua.test, ub.base y ub.test) tomados de manera que en cada fichero de entrenamiento haya exactamente 10 ratings por usuario.
- mku.sh: script de consola de Linux que genera todos los ficheros de entrenamiento y test descritos anteriormente de forma automática.

4. Diseño y desarrollo

El objetivo del proyecto a realizar es el desarrollo de una aplicación web que facilite a los usuarios la visualización y el estudio de conjuntos de datos, así como los resultados de la aplicación de algoritmos de recomendación sobre ellos.
En los siguientes apartados se profundizará en el proceso de diseño, análisis y desarrollo del proyecto.

4.1 Análisis de requisitos

El análisis de los requisitos de un proyecto se basa en la descripción de todas las cualidades que debe cumplir el mismo para que se pueda considerar que se ha completado su desarrollo. Dichas características deben quedar claramente descritas de manera que no existan dudas sobre su consecución a posteriori.

4.1.1 Requisitos funcionales

Son aquellos requisitos relativos a la funcionalidad de la aplicación, es decir, a las funciones que el sistema debe ser capaz de satisfacer. En estos términos, el sistema deberá:

1. Tener acceso a una base de datos en la que se almacene la información del dataset seleccionado organizada de forma coherente para permitir realizar consultas en relación a ella.
2. Almacenar en la misma base de datos el conjunto de resultados obtenidos tras aplicar distintos métodos de recomendación sobre la información original contenida en la base de datos. También deberá almacenar los resultados de las evaluaciones de los distintos recomendadores utilizados.
3. Dividir la información que se muestra en la pantalla de manera que en un área se muestren las opciones de filtrado que el usuario puede aplicar y en el resto de la pantalla se carguen las gráficas que el usuario desee visualizar en cada momento.
4. Ser capaz de leer las opciones de filtrado seleccionadas por el usuario, hacer una llamada a la base de datos solicitando la información requerida y cargar la gráfica seleccionada por el usuario.
5. Ofrecer a los usuarios distintas formas de visualización de los datos almacenados en la base de datos. [33] [34] Las posibilidades que se deben ofrecer son:
    - Agrupación del número de películas por periodos de tiempo en un diagrama de barras en función de su fecha de estreno.

- Agrupación del número de valoraciones realizadas por periodos de tiempo en un diagrama de barras en función de la fecha de estreno de la película que califican.
- Treemap de tres niveles: en el nivel superior se agruparán los datos por períodos de tiempo, en el segundo nivel por el género al que pertenecen y en el tercer nivel se verán las 10 películas mejor valoradas. El color del cuadrilátero que representa cada película refleja la media de todos los ratings de dicha película: las valoradas con más de 4.5 se verán en verde, las que estén entre 4 y 4.5 en azul y las que tengan una media inferior a 4 en amarillo.
- Gráfico circular que reflejará el número de películas que han sido votadas distinguiendo si la persona que ha realizado la votación era hombre o mujer.
- Gráfico circular que muestra el número de películas agrupadas en función del género al que pertencen.
- Gráfico de líneas que relaciona un conjunto de usuarios con las películas que mejor ha valorado. Se debe dar la posibilidad de elegir los usuarios que se quieren mostrar o de que la aplicación genere 5 usuarios aleatorios para mostrar su información.
- Gráfico de líneas que relaciona un conjunto de usuarios con las películas que no ha valorado pero que se predice que valorará con más nota. Se debe dar la posibilidad de elegir el algoritmo cuyas predicciones se desea conocer, así como los ids de los usuarios que se quieren mostrar o de que la aplicación genere 5 usuarios aleatorios.
- Gráficos de líneas en los que deben visualizar los resultados de las métricas de evaluación para cada uno de los algoritmos de recomendación utilizados en la aplicación. Dado que, como se ha estudiado anteriormente, los algoritmos no tiene los mismos rangos de valores, se separarán las distintas métricas en dos gráficas distintas: en una quedarán representados el MAE y el RMSE y en la otra el NMAE y el CDB (rango de 0 a 1).
- Treemap de dos niveles en el que se mostrará en el primer nivel cada uno de los recomendadores y en el segundo nivel los valores de las métricas de evaluación de cada uno de ellos.

6. Permitir al usuario de una manera sencilla cambiar el tipo de gráfico y la agrupación de información que se está representando.
7. Ofrecer al usuario una serie de filtros basados en los atributos de los ítems que se han almacenado en la base de datos con el fin de que pueda limitar la información que se va a mostrar en la visualización. Dichos filtros pueden estar basados en:

- Género de la película: se podrán elegir uno o varios de estos géneros, que pueden ser: “Action”, “Adventure”, “Animation”, “Children”, “Comdey”, “Crime”, “Documentary”, “Drama”, “Fantasy”, “Film-noir”, “Horror”, “Musical”, “Mystery”, “Romance”, “Sci-Fi”, “Thriller”, “War”, “Western”. [35]
- Periodo de tiempo: forma en la que se quiere realizar la agrupación temporal en los gráficos que lo permitan. Habrá tres opciones y solo se podrá elegir una: “By week” (semanal), “By month” (mensual) y “By decade” (por décadas). En los dos primeros casos será necesario especificar el año en el que se quiere realizar la agrupación.

8. Ofrecer al usuario la opción de elegir el algoritmo cuyos resultados desea ver en todas aquellas visualizaciones que impliquen mostrar resultados de la aplicación de métodos de recomendación. Las opciones que se podrán elegir en este filtro serán: “Baseline”, “UserKnn”, “ItemKNN”, “Global Average” y “Matrix factorization”. [22]
9. Hacer aparecer un tooltip cuando el usuario sitúe el cursor del ratón sobre la gráfica que contendrá información acerca del elemento sobre el cual el usuario se encuentre.
10. Mostrar un botón que haga aparecer un cuadro de ayuda con información acerca del funcionamiento de la aplicación y de los algoritmos que se utilizan en ella.

4.1.2 Requisitos no funcionales

Este conjunto de requisitos no guardan relación con las funcionalidades de la aplicación, sino que definen características y comportamientos que el sistema debe cumplir durante su funcionamiento. En cuanto a requisitos no funcionales, el sistema deberá:

1. Ofrecer una interfaz de trabajo amigable y sencilla de utilizar, en la que la funcionalidad de todos los elementos quede clara.
2. Mostrar una estética suave, sin cambios de color bruscos y sin combinaciones de color de fondo y de fuente que dificulten la lectura de los textos.
3. Mantener una relación coherente de fuentes y tipografías en toda la aplicación haciendo que los títulos compartan fuente entre ellos y el resto de textos.
4. Asegurar que el tiempo de reacción desde que el usuario pulsa el botón de “Update graphic” hasta que se completan las transiciones de eliminación y creación de gráficas es inferior a 3 segundos (sin contar el tiempo de recuperación de datos, que depende de factores ajenos a la aplicación, como la conexión del cliente).
5. Ofrecer toda la información escrita en la aplicación en el mismo idioma, en este caso el idioma elegido será el inglés.

4.2 Diseño de la aplicación

Basándonos en las necesidades del proyecto y debido a su magnitud, es aconsejable dividir el conjunto de funcionalidades en módulos más sencillos con el fin de facilitar su comprensión e implementación posterior. La primera división que se puede realizar agrupará los elementos que componen la aplicación en función de la forma en que interactúen con la misma. De esta manera, podremos diferenciar dos grandes grupos: elementos relativos a las funcionalidades de la aplicación y elementos relativos a la carga de datos.

En general, en el primer grupo se englobarán todas las clases y métodos que dotan de funcionalidad a la aplicación y que son necesarios para permitir el correcto funcionamiento de la misma. Las clases y métodos que engloba este grupo se ejecutarán durante todo el ciclo de vida de la aplicación (o hasta que se considere oportuno eliminar la funcionalidad que proporcionan).

Por otro lado, en el segundo grupo, se encontrarán los procesos que sólo se ejecutan en ciertos momentos concretos de la vida del sistema, ya que los resultados de sus acciones permanecen en la aplicación (almacenados de manera persistente).

A continuación se realizará un análisis más preciso de cada uno de estos grupos para poder dividirlos a su vez en sub-módulos más sencillos.

4.2.1 Elementos relativos a las funcionalidades de la aplicación.

Como ya se ha mencionado, en este grupo se pretende incluir todos los elementos que proporcionan una funcionalidad concreta a la aplicación. La separación en módulos más sencillos que se hará dentro de los elementos de este grupo está basada en la parte de la aplicación que manejan: controlador o vista.

- **Elementos del controlador.**
  Forman parte de este grupo aquellos elementos encargados de la lógica de la aplicación que se ejecutan del lado del servidor. Sus principales funciones se basan en conectar con la base de datos para solicitar información y realizar operaciones con los datos que le llegan.
- **Elementos de la vista**
  A pesar de que, por definición, los elementos de la vista de una aplicación no deben aportar funcionalidad a la misma sino simplemente encargarse de su estética, se ha considerado necesario separar ciertos elementos en un módulo aparte ya que contarán con la lógica necesaria para la creación de gráficas y el control de los eventos de la aplicación.

4.2.2 Elementos relativos a la carga de datos

Aunque en el nombre de este conjunto se menciona la carga de datos, en él se engloban tanto las funciones que realizan la carga de la información incluida en los ficheros proporcionados en el dataset en la base de datos como el trabajo previo de creación de la base de datos y tratamiento y conversión de los datos. Las distintas acciones en las que se puede dividir este grupo son:

- **Creación de la base de datos:** creación de la base de datos y definición de las tablas necesarias para almacenar la información de la aplicación.
- **Lectura y carga de los ficheros del conjunto de datos:** proceso por el cual se lee la información del conjunto de datos y se almacena en las tablas correspondientes de la base de datos.
- **Generación de recomendaciones:** dado que se ha decidido evitar las limitaciones de rendimiento que supondría ejecutar las recomendaciones a medida que son solicitadas por el usuario, será necesario realizarlas previamente y escribir los resultados en un fichero de texto.
- **Carga de las recomendaciones en la base de datos:** proceso mediante el cual se lee el fichero con las recomendaciones generadas y se van almacenando en la base de datos.
- **Evaluación de los recomendadores seleccionados:** se evaluarán todos los recomendadores incluidos en la aplicación y los resultados, al igual que las predicciones de ratings, se escribirán en un fichero de texto.
- **Carga de los resultados de las evaluaciones en la base de datos:** proceso por el cual se lee el fichero con los datos de las evaluaciones y se van almacenando en la base de datos de la aplicación.

4.3 Diseño de la base de datos

El motor de base de datos que se va a utilizar en la aplicación es SQLite. La base de datos que utilizará el sistema implementado deberá ser capaz de almacenar la información contenida en el dataset seleccionado de MovieLens junto con los resultados de las evaluaciones de los sistemas de recomendación sobre dichos datos, así como sus predicciones.

Se pueden distinguir dos contextos dentro las necesidades de la base de datos de la aplicación:

- Conjunto de tablas en las que se almacenará la información acerca de usuarios, películas y ratings contenida en el dataset. Simplemente se leerán los ficheros fuente con la información y se guardará en la base de datos.
- Conjunto de tablas en las que se almacenarán los resultados de aplicar las métricas de recomendación sobre la información del dataset original. En este grupo se pueden definir a su vez dos subgrupos: por un lado se deben guardar

los ratings predichos por los algoritmos de recomendación y, por otro, los resultados obtenidos al evaluar los recomendadores utilizados.

En la siguiente ilustración se muestra un esquema con la base de datos relacional que se ha diseñado para almacenar la información de la aplicación.

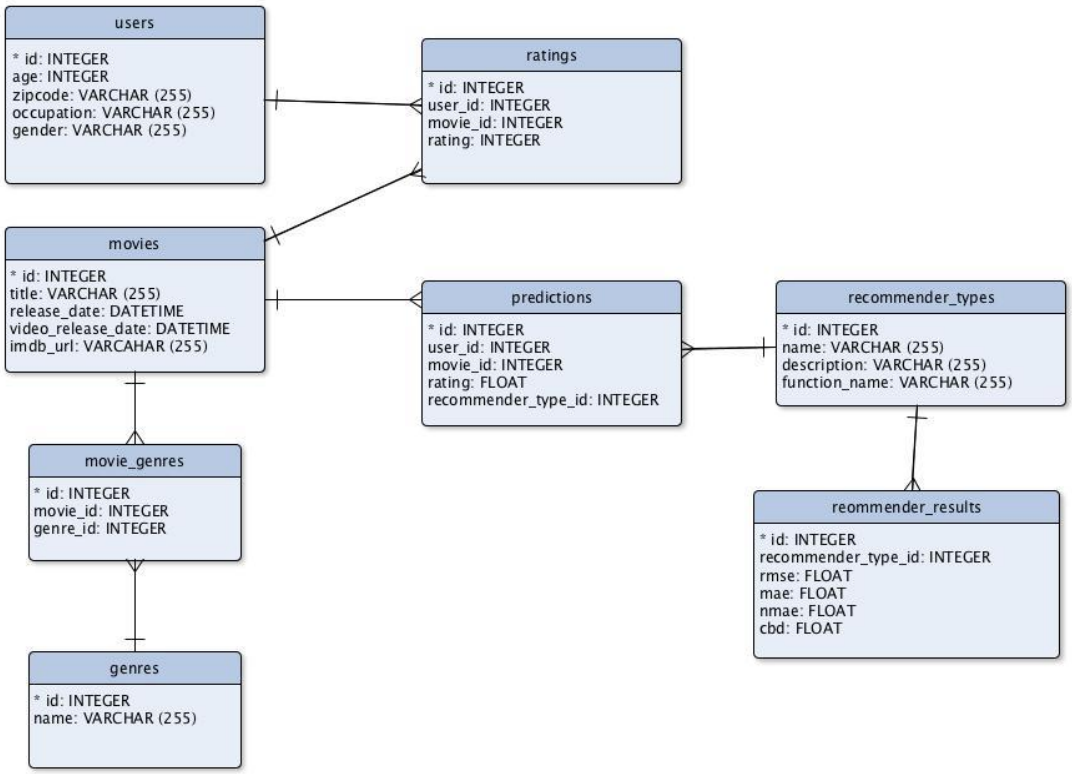


Imagen 1: Diseño de la base de datos de la aplicación.

A continuación se procederá a describir brevemente las entidades que componen la base de datos del diagrama anterior:

- Users: tabla que contiene los datos demográficos de los usuarios junto al identificador único de cada uno.
- Movies: tabla con los datos de las películas junto con el identificado asignada a cada una. Esta tabla se podría haber llamado "ítems" con el fin de hacer la base de datos más genérica, pero ya que la aplicación se va a centrar en los datos del dataset de películas, se ha tomado la decisión de especificar el tipo de ítem que se usa.
- Movie_genres: un listado que relaciona los identificadores de las películas con los de los géneros a los que pertenece. Si una película pertenece a varios géneros, su identificador aparecerá en la tabla varias veces.
- Genres: listado con los géneros a los que puede pertenecer una película con un identificador único para cada uno.
- Ratings: relación de los usuarios con los ítems (películas) que han votado junto con la valoración que le dieron.

- Predictions: relación de los usuarios con los ítems que no han valorado junto con la nota que se predice para cada par usuario-ítem y el identificador del recomendador que ha generado la predicción.
- Recommender_types: lista de recomendadores incluidos en la aplicación. Esta tabla contiene un campo llamado “function_name” en el que se almacenará el nombre de la función de la librería de MyMediaLite que se usa para generar cada recomendador.
- Recommender_results: tabla donde se almacenarán los resultados de las cuatro métricas de evaluación para los recomendadores incluidos en la aplicación.

4.4 Desarrollo del proyecto

En este apartado se llevará a cabo una explicación detallada de los procedimientos que se han seguido para completar la implementación de los módulos descritos en el apartado de diseño de la aplicación.

4.4.1 Creación de la base de datos.

Una vez terminado el análisis de requisitos de la aplicación y llevado a cabo el diseño de la base de datos, se puede proceder a su creación.

Mientras que en otros marcos de trabajo, el proceso de creación de la base de datos consiste en la ejecución en cadena de una serie de complejas sentencias SQL, gracias a las herramientas que proporciona Ruby on Rails, el proceso resulta más sencillo y amigable si se hace correctamente. Es necesario, por tanto, profundizar un poco más en el funcionamiento de Ruby on Rails para comprender el proceso que se ha seguido.

Ya se ha mencionado que, en Rails, cada modelo que se declara se corresponde con una tabla de la base de datos. La generación de los modelos debe realizarse a través de la consola de Rails, mediante comandos que siguen la siguiente sintaxis:

> “rails generate model Rating user_id:integer movie_id:integer rating:float timestamp:datetime”

Mediante le ejecución de este comando, se generará un fichero “rating.rb” en la carpeta correspondiente a los modelos de la aplicación. Como se puede observar, en el comando se define el nombre del modelo así como los atributos que tendrá el mismo. Cabe destacar que Rails asigna automáticamente un id a cada modelo, por lo que no es necesario especificar este atributo en la creación del mismo. Otro hecho reseñable es que el funcionamiento y las relaciones que establece Rails se basan en muchas ocasiones en la nomenclatura de los elementos. Mediante la ejecución del comando anterior, Rails sabe que se está creando un modelo llamado Rating pero además, reconoce los elementos “user_id” y “movie_id” como referencias a los ids de los modelos “User” y “Movie”, respectivamente, por lo que ya se crea una relación entre el nuevo modelo y los otros dos.

En este punto, ya se ha creado el modelo Rating, pero no se ha llevado a cabo ninguna acción con respecto a la base de datos. No obstante, el fichero “rating.rb” no es el único que se crea al ejecutar el comando de generación del modelo. Se debe haber creado otro fichero en la carpeta del proyecto “/db/migrate” con un nombre que siga la nomenclatura “XXXXXXXXXXXXX_create_ratings.rb” donde las equis son números que se corresponden a la fecha de creación del fichero. Estos ficheros reciben el nombre de “migraciones” y contienen las sentencias de código Ruby necesarias para realizar modificaciones en las tablas de la base de datos. En este caso concreto, el código que genera el comando anterior es:

```
create_table :ratings do |t|
  t.integer :movie_id
  t.integer :user_id
  t.integer :rating
  t.datetime :timestamp
  t.timestamps
```

Puede sorprender el hecho de que, a pesar de tratarse de sentencias que deben interactuar con la base de datos, no hay código SQL en ellas. En esto reside una de las principales ventajas de Ruby on Rails: la abstracción de la base de datos. Las sentencias de código Ruby de las migraciones son ejecutadas por un controlador que se encarga de traducirlas a SQL. En función del motor de bases de datos utilizado en la aplicación, el controlador convertirá la instrucción de Ruby en el código SQL correcto. De esta manera, el cambio de un motor de bases de datos a otro no implica tener que llevar a cabo modificaciones en todas las partes de la aplicación en las que se modifique la base de datos.

Es importante destacar que, aunque tras generar los modelos los ficheros con las migraciones han sido creados, se debe ejecutar por consola el siguiente comando para que se ejecuten [36]:

```
rake db:migrate
```

De hecho, Rails sabe qué migraciones ya han sido ejecutadas en la base de datos y mediante este comando solo se ejecutarán aquellas que no lo hayan hecho aún. Es más, en el comando se puede especificar hasta qué migración se desea ejecutar. También existen comandos para deshacer migraciones que ya se habían ejecutado, en caso de haber cometido algún error en la definición de un modelo.

Una vez se ha comprendido el proceso de generación de modelos en Rails, se pueden terminar de crear todos los modelos necesarios para el funcionamiento de la aplicación, junto con sus respectivas migraciones. Como se ha dicho antes, los modelos a crear están definidos en el diagrama de la base de datos.

4.4.2 Carga de información en la base de datos.

Una vez creadas todas las tablas necesarias en la base de datos, es necesario cargarlas con los datos que se mostrarán en la aplicación. Para ello, se utilizará una funcionalidad de Rails que permite programar tareas y ejecutarlas desde la consola de comandos. Dicha funcionalidad se llama "Rake" y ya se han visto alguna de sus funciones en la descripción de la creación de la base de datos. La sintaxis de los comandos que se ejecutaran una vez programados los scripts de carga debe ser similar a:

```
rake data_load:load_ratings
```

En dicho comando se especifica el nombre del fichero en el que está escrita la tarea (dicho fichero debe estar en la ruta del proyecto "/lib/tasks/data_load.rake") y la tarea concreta dentro del fichero de Rake que se quiere ejecutar ("load_ratings").
Para realizar la carga de datos del dataset se han programado un total de cinco tareas de Rake:

- Load_ratings: rellena la tabla "ratings" a partir de los datos almacenados en el fichero "u.data".
- Load_users: rellena la tabla "users" a partir de los datos del fichero "u.user"
- Load_movies: rellena la tabla "movies" con los datos almacenados en el fichero "u.item".
- Load_genres: rellena la tabla de "genres" con la información del fichero "u.genre".
- Load_movie_genres: rellena la tabla "movie_genres" leyendo la información relativa a los géneros de las películas del fichero "u.item".

Para saber cómo llevar a cabo la lectura de la información de los ficheros se acudirá a la documentación del conjunto de datos. En ella se explica que la información de cada fichero viene separada por líneas y que los atributos de cada línea están separados por tabuladores o por barras verticales ('|'). La carga de datos de cada tabla consistirá, por tanto, en ir leyendo el fichero correspondiente línea a línea y separar los atributos de cada línea para ir creando elementos de cada tipo y cargarlos en la base de datos.
Hay que destacar que en esta ocasión, al igual que en la creación de la base de datos, no será necesario utilizar código SQL para rellenar las tablas. Basta con crear nuevas instancias del modelo correspondientes a la tabla que se quiera rellenar, insertar los valores de sus atributos y guardar la instancia mediante una llamada al método "save".
En el "Anexo A: Carga de un fichero de MovieLens en la base de datos" se puede ver un ejemplo del código correspondiente a una carga de datos de la aplicación.

4.4.3 Generación de las recomendaciones

Como ya se ha mencionado en los apartados anteriores, la librería de recomendaciones que se ha elegido para utilizar en la aplicación es MyMediaLite. Dadas las limitaciones de rendimiento que supondría ejecutar las recomendaciones en el momento en que sean solicitadas por el usuario, se tomó la decisión de aplicar los algoritmos de recomendación sobre el conjunto de datos y tener almacenados los resultados obtenidos en la base de datos. De esta manera, cuando un usuario solicite resultados de recomendaciones, solo será necesario realizar una consulta para obtener la información.
También se ha comentado ya la necesidad de utilizar IronRuby como intermediario para poder combinar scripts de Ruby con las librerías de MyMediaLite. El primer paso de la generación de las recomendaciones se realizará mediante llamadas a tareas de Rake. Se escribirá una tarea distinta para cada recomendador que se desee introducir en la aplicación. En dichas tareas, se ejecutará la función "sh" de Ruby, a la que se le pasa un string que será ejecutado por consola. En dicho string simplemente se realizará una llamada al comando "ir", que al ejecutarse por consola lanzará la aplicación IronRuby pasándole como argumentos el script de Ruby que debe generar las recomendaciones y el número de usuarios y películas que hay en la base de datos. Se han implementado un total de cinco script que debe ejecutar IronRuby, uno por cada recomendador que se debe aplicar a los datos. Estos scripts se han almacenado en la carpeta "prediction_scripts" y son los siguientes:

- complete_base_line_prediction.rb
- complete_user_knn_prediction.rb
- complete_item_knn_prediction.rb
- complete_matrix_factorization_prediction.rb
- complete_global_average_prediction.rb

Cada script encargado de generar las recomendaciones consta de dos fases: en la primera se declara el recomendador correspondiente y se entrena con uno de los conjuntos de datos de entrenamiento y en la segunda se ejecuta un bucle doble en el que el recomendador generará predicciones para cada usuario y para cada película de la base de datos, siempre que no hubiera una entrada para dicho usuario y película en el conjunto de datos con el que se ha entrenado al recomendador. Las recomendaciones serán almacenadas en un fichero de texto en el que cada línea contendrá el id del usuario, el id de la película y el rating predicho separados por tabuladores.
Dado que para generar las recomendaciones se debe ejecutar un script distinto para cada recomendador, las tareas implementadas para esta tarea se encuentran en el fichero "/lb/tasks/rating_prediction.rake" y son:

- predict_ratings_base_line
- predict_ratings_user_knn
- predict_ratings_item_knn
- predict_ratings_matrix_factorization
- predict_ratings_global_average

En “Anexo B: Scripts de generación de recomendaciones.” se puede estudiar en profundidad el funcionamiento de la generación de recomendaciones implementada a través del código de los scripts que la ejecutan.

4.4.4 Evaluación de los recomendadores seleccionados para la aplicación.

Ya que también se utilizará la librería de MyMediaLite para llevar a cabo la evaluación de los recomendadores, será necesario de nuevo ejecutar un script que realice la tarea por medio de IronRuby. Dicho script inicializará cada recomendador de la aplicación, los entrenará utilizando el conjunto de datos de entrenamiento y después los evaluará gracias al conjunto de datos de test. El resultado de las evaluaciones será almacenado en un fichero de texto en el que cada línea corresponderá a un recomendador y cada columna, separadas por tabuladores, a una métrica.
La tarea que lleva a cabo la labor de evaluar los recomendadores se encuentra en el fichero “/lib/tasks/recommender_results.rake” y se trata de:

- execute_recommender_results_script

En “Anexo C: Scripts de evaluación de los algoritmos de recomendación” se ha desarrollado el código utilizado para la evaluación de los métodos de recomendación y la carga de datos de los resultados generados.

4.4.5 Carga de los resultados de la recomendación y la evaluación en la base de datos

Dado que se ha elegido el mismo formato para la escritura de los ficheros de los resultados de recomendaciones y evaluaciones, el proceso de carga de ambos tipos de ficheros será similar y seguirán el mismo procedimiento que el descrito en la carga de los datos originales del dataset.
Para la carga de los resultados de los recomendadores se han implementado las siguientes tareas en el fichero “/lb/tasks/rating_prediction.rake”, cada una correspondiente a un recomendador:

- load_base_line_prediction
- load_user_knn_prediction
- load_item_knn_prediction
- load_matrix_factorization_prediction

- load_global_average_prediction

Para la carga de los resultados de las evaluaciones solo es necesaria la ejecución de una tarea del fichero "/lib/task/recommenders_results.rake":

- load_recommenders_results

4.4.6 Implementación del controlador.

Durante el desarrollo del proyecto sólo se ha considerado necesaria la implementación de un controlador, que se encargará de manejar la lógica de toda la aplicación y recibe el nombre de "MovieController". Dicho controlador está compuesto por un conjunto de funciones que se van llamando a medida que el usuario realiza peticiones de visualización de datos. Además comprueba que las llamadas que recibe contienen el número de argumentos necesarios para poder ejecutarse correctamente y realiza llamadas a otras funciones que recuperan los datos almacenados. Algunas de las funciones implementadas en este controlador precisan modificar la presentación de los datos que recibe para poder enviarlos al cliente en el formato adecuado para la visualización solicitada.
La funcionalidad del controlador descrito se ve ampliada por un conjunto de ficheros que en este marco de trabajo reciben el nombre de "ayudante" o "helpers". En estos ficheros se permite declarar funciones auxiliares que pueden ser llamadas desde cualquier controlador. Los motivos para la utilización de estas clases auxiliares son principalmente reducir la cantidad de código del controlador para hacerlo más legible y hacerlo más modular para facilitar la realización de pruebas. En este caso, se podrán separar las funciones implementadas en los "helpers" en dos tipos: aquellas que cuando son llamadas realizan una petición de datos a la base de datos para devolvérselos al controlador (de este tipo serán la mayor parte) y aquellas que son llamadas por el controlador para modificar el formato en el que vienen los datos y así hacerlos válidos para poder enviárselos al cliente. En el desarrollo de la aplicación se ha considerado necesaria la creación de dos "helpers": DataVisualizationHelper que realizará funciones relativas a la visualización de los datos originales y el DataRecommendationHelper cuyas funciones estarán relacionadas con la recuperación de los datos obtenidos de las recomendaciones.

4.4.7 Implementación de la vista.

En este apartado se va a realizar una descripción de dos entidades principales que otorgan funcionalidad a la vista. Dado que se trata de elementos que forman parte de la vista, se deben ejecutar en el lado del cliente, por lo que están escritos en lenguaje JavaScript.

En primer lugar, existe una función de JavaScript que se ejecuta cuando se carga la pantalla principal de la aplicación. En ella, aparte de inicializar ciertos elementos estéticos de la interfaz mediante la librería de jQuery, se inicializará el control de eventos de la aplicación. Es gracias a esta función que la vista de la aplicación interactúa con las acciones de los usuarios de forma dinámica, sin que sea necesario recargar la página web cada vez que el usuario selecciona una opción de la interfaz.
En segundo lugar, se han implementado una serie de funciones de JavaScript por medio de la librería D3js que se utilizan para la representación de gráficas en la aplicación. Cada una de estas funciones se ha desarrollado en un documento de JavaScript distinto en función del tipo de gráfica que se encarguen de representar. Como ya se mencionó anteriormente, la librería D3 permite la creación de distintos tipos de gráficas por medio de JavaScript, pero su funcionalidad va más allá de eso. Permite al desarrollador personalizar la estética de las gráficas, las transiciones y efectos para su creación y destrucción y las respuestas a los eventos que tienen lugar sobre ellas. Dado que la aplicación debía cumplir la funcionalidad de crear, destruir y actualizar gráficas, se han implementado tres funciones para cada tipo de gráfica tratando de conseguir una experiencia lo más fluida posible para el usuario:

- Funciones de creación: se encargan de crear el elemento svg sobre el que posteriormente se dibujarán las gráficas en el div que las debe contener. Además asignarán el tamaño del mismo a partir de los parámetros que recibe. Un ejemplo de este tipo de funciones sería

  CreateTreemap(graphic_width, graphic_height, graphic_margin, graphic_div).

- Funciones de carga: reciben como parámetros el nombre del elemento div sobre el que se debe dibujar la gráfica para crearla y los datos que debe mostrar. Es preciso que antes de llamar a esta función se haya realizado previamente la llamada a la función de creación correspondiente. La complejidad de estas funciones reside en que deben ser capaces tanto de crear gráficas partiendo de cero como de actualizar los datos que muestra una gráfica ya existente. Una función de este tipo es:

  LoadBarChartGraphic(data, graphic_width, graphic_height, graphic_margin, graphic_div).

- Funciones de borrado: deben tener la funcionalidad de hacer desaparecer mediante transiciones los elementos de la gráfica correspondiente para luego poder eliminarlos para dejar el elemento div que contiene las gráficas vacío y listo para que se pueda crear una gráfica nueva. Una de las funciones de borrado implementadas es:

  RemoveLineChartGraphic(graphic_width, graphic_height, graphic_margin, graphic_div).

En “Anexo D: Esquema de la vista de la aplicación” se puede encontrar una maqueta de la interfaz implementada para la vista de la aplicación.

5. Pruebas

5.1 Pruebas unitarias

Se entiende por pruebas unitarias aquellas que se realizan individualmente sobre cada uno de los módulos en los que se ha dividido la aplicación durante la fase de diseño. El objetivo es comprobar que cumplen correctamente con todas las funcionalidades que se espera de ellos así como que no presentan comportamientos erráticos.

5.1.1 Pruebas unitarias en el módulo de generación de predicciones.

Debido al enorme número de usuarios e ítems que componen la aplicación resulta inviable pretender comprobar que la generación de predicciones se ha realizado de forma correcta para todos los casos. Es necesario buscar, por tanto, otras formas de verificar de manera fiable que dichas predicciones son correctas. Parar ello, se deben analizar los ficheros que generan cada script de recomendación para comprobar que se cumplen los siguientes requisitos:

- El número de predicciones que se llevan a cabo debe ser igual al producto del número total de usuarios por el número total de ítems (ya que se debe hacer una predicción de cada usuario para todos lo ítems) menos el número de ratings que haya en el fichero utilizado para entrenar al recomendador. Es decir, cada script de recomendación debe generar un fichero de 943*1682-80000 = 1506126 filas.
- No debe haber ninguna predicción en el fichero de resultados que se encuentre en el fichero de entrenamiento utilizado. Como el fichero de entrenamiento contiene 80000 filas, se elegirán 20 entradas aleatorias en el fichero de entrenamiento y se comprobará que no están incluidas en el fichero de los resultados.
- Los resultados de las recomendaciones incluidas en el fichero de salida deben ser correctos, entendiéndose por “correcto” iguales a los que genera el recomendador correspondiente. Para ellos se elegirán otra vez20 entradas aleatorias del fichero de resultados y se ejecutará un script que genere estas recomendaciones para comparar así los resultados obtenidos.

5.1.2 Pruebas unitarias en el módulo de evaluación de recomendadores.

Dado que éste es el módulo que menor cantidad de resultados calcula, su contenido es el más fácil de verificar. Para comprobar su correcto funcionamiento es necesario asegurarse de que cumple las siguientes características:

- Comprobar que se extraen las métricas deseadas para cada uno de los recomendadores, esto es, comprobar que en el fichero se escriben cuatro

métricas (MAE, NMAE, RMSE y CBD) para cada uno de los cinco recomendadores que se incluirán en la aplicación.

- Comprobar que los valores almacenados en el fichero por el script de evaluación se corresponden con los valores ideales de cada uno de los recomendadores.

5.1.3 Pruebas unitarias en el módulo de carga de datos.

Este conjunto de pruebas debe verificar que la carga de la información de los ficheros a la base de datos es correcta, tanto los ficheros incluidos en el dataset como los generados tras la aplicación de las métricas de recomendación. En primer lugar se comprobarán aquellos datos que tienen una extensión razonable para poder ser estudiados:

- Los datos de la tabla de "recommender_results" son los mismos que los del fichero de salida del script de evaluación.
- El número de géneros creados en la tabla "genres" se corresponde con los que hay en los datos de MovieLens.

El resto de tablas, lamentablemente, contienen demasiados datos como para comprobar su integridad manualmente. La solución buscada para solventar este problema es abrir lo ficheros "u.item", "u.user", "u.data" y los ficheros de resultados de recomendaciones con un gestor de hojas de cálculo que permita filtrar los datos. Con esta herramienta, se pueden realizar las siguientes comprobaciones.

- Filtrar la hoja de cálculo de los ítems y comprobar que el número de ítems de cada género insertado en la base de datos se corresponde con el número real.
- Comprobar que las fechas de los ítems insertadas en la base de datos son correctas filtrando la hoja de cálculo correspondiente por año y mes.
- Filtrando por la columna de "sexo" de la hoja de cálculo de los usuarios, comprobar que el número de usuarios masculinos y femeninos reales se corresponde con los insertados en la base de datos.
- Filtrar la hoja de cálculo de "u.data" para verificar que el número de ratings correspondientes a un cierto usuario o ítem es correcto. Aunque se use este método, no se puede comprobar la totalidad de los datos en este caso, por lo tanto, se limitará la comprobación a los ratings correspondientes a 10 usuarios y 10 ítems aleatorios.
- Para comprobar que los datos de los scripts de recomendaciones también se han insertado correctamente, se aplicará el mismo método que para el fichero "u.data" con la hoja de cálculo correspondiente a cada uno de los recomendadores cuyas predicciones se han calculado.

Si se superan con éxito las pruebas descritas, se puede dar por hecho que la carga de datos se ha llevado a cabo de manera correcta.

5.1.4 Pruebas unitarias en el controlador de películas.

Se incluirán en este grupo también los tests correspondientes a las funciones implementadas en los "helpers" de la aplicación. Las pruebas que deben llevarse a cabo en este módulo son:

- Comprobar que todas las funciones del controlador fallan si no se le mandan los parámetros correctos.
- Comprobar que, según los parámetros recibidos, las funciones del controlador realizan llamadas a las funciones correspondientes de los "helpers".
- Comprobar que las queries que realizan las funciones de los "helpers" están correctamente codificadas y devuelven los datos que deben.
- Comprobar que las funciones encargadas de dar formato a los datos devueltos por la base de datos funcionan correctamente, devolviendo los datos correctos bien formateados.

5.1.5 Pruebas unitarias de la funcionalidad de la vista.

En este grupo se pretenden testear especialmente las funciones relativas a la visualización de los datos y a la gestión de eventos entre visualizaciones. Como su propio nombre indica, la mayor parte de las pruebas del módulo correspondiente a la vista no pueden automatizarse, sino que hay que comprobar el correcto funcionamiento del módulo "a ojo". Para ello tras crear un conjunto de datos con el formato adecuado para cada gráfica, se debe verificar que:

- La creación de gráficas se produce de manera correcta en la interfaz, sin redimensionar elementos "div" de la interfaz ni desplazar otros elementos de su sitio.
- La actualización de los datos de las gráficas funciona de manera óptima, sin que queden elementos salientes de la gráfica sin borrar ni elementos que deban actualizar con sus valores antiguos.
- Comprobar que el tiempo de espera de una nueva gráfica que se carga es suficiente respecto al tiempo que tarda la gráfica cargada previamente en realizar las transiciones de borrado. De esta forma, no se debe producir solapamiento entre los elementos de dos gráficas distintas.
- Comprobar que cuando se realiza el borrado de una gráfica no queda ningún elemento residual sin borrar que posteriormente pueda suponer errores en otras visualizaciones.
- La totalidad de los eventos que se reciben de la interfaz deben ser atendidos por la función de JavaScript que se ejecuta cuando se carga la página. Es especialmente importante comprobar la gestión de eventos que impliquen

cambios en el menú de filtros de la aplicación y los que impliquen recargar la visualización de datos que se está realizando en un determinado momento.

5.2 Pruebas de integración

Las pruebas de integración tienen como objetivo verificar el correcto funcionamiento de los módulos en los que se divide la aplicación trabajando juntos.

5.2.1 Integración de eventos con carga de gráficas.

Esta prueba consiste en modificar el tipo de gráfica que se quiere cargar mediante los filtros de la interfaz y comprobar que, cuando se pulsa el botón de "Update graphic" la aplicación funciona como debe, a saber:

- En caso de que el tipo de gráfico no cambie, se actualicen los datos mostrados sin eliminar el gráfico.
- En caso de que se haya modificado el tipo de gráfico que se representa, se oculte el gráfico que se estaba mostrando, se borren los elementos que lo componían y se cree el nuevo gráfico seleccionado.

Es necesario realizar las pruebas descritas con todas las posibilidades de gráficos que se puedan seleccionar para poder confirmar que la aplicación supera las pruebas por completo.

5.2.2 Integración de eventos con controlador.

Las pruebas de este grupo se basan en generar distintos tipos de filtros en la interfaz y comprobar que, cuando se pulsa el botón de "Update graphic", se construyen las URL necesarias para que se ejecuten llamadas a las funciones del controlador en el servidor. Es importante verificar que cuando se realizan esas llamadas, ninguna falla debido a la ausencia de parámetros necesarios en la llamada.
Una vez más, para poder afirmar que la aplicación supera las pruebas de este conjunto es necesario probar todos los posibles filtros que se pueden seleccionar en la interfaz.

5.2.3 Integración de controlador con carga de datos.

Este conjunto de pruebas serán las que verifiquen que el formato de las respuestas que se obtienen de las llamadas a las funciones del controlador son correctos.
Cada gráfica que puede ser creada por la aplicación necesita un formato de datos distinto. Es por esto que, dependiendo del tipo de gráfica que se desee visualizar, el controlador deberá devolver los datos en un formato o en otro. Para llevar a cabo las pruebas será necesario realizar llamadas a las funciones del controlador solicitando datos para todos los tipos de gráficas que se ofrecen en la aplicación y comprobar que el formato de los datos devueltos es correcto.

5.2.4 Integración de vista con carga de datos.

Estas pruebas deberán comprobar que los elementos de la interfaz que dependen de la base de datos reciben la información de forma correcta. Hay dos elementos a comprobar en este apartado: el selector de años del menú de filtros, que debe mostrar la lista de todos los años en los que se tiene información del estreno de una película, y el selector del método de recomendación, que debe contener los recomendadores seleccionados para la aplicación, que habrán sido introducidos en la base de datos.

5.3 Pruebas de sistema

En estas pruebas se comprobará que el sistema funciona correctamente en todo su conjunto, y que la aplicación cumple con todos los requisitos descritos en el apartado 3.1.

Para llevar a cabo la ejecución de estas pruebas, se describirán unos casos de uso de la aplicación que deberán ser completados para considerar que el proyecto ha sido finalizado satisfactoriamente:

- **Caso de uso 1:** visualizar en un diagrama de barras la evolución del número total de películas estrenadas agrupadas por décadas.
- **Caso de uso 2:** visualizar en un diagrama de barras la evolución del número de películas del género “thriller” agrupadas por décadas.
- **Caso de uso 3:** visualizar en un diagrama de barras el número de votaciones que han recibido las películas pertenecientes a los géneros “action”, “adventure” o “animation” agrupados por décadas en función de la fecha de estreno de la película.
- **Caso de uso 4:** Comparar el número de votaciones realizadas por usuarios masculinos y femeninos en películas estrenadas en 1998 pertenecientes al género “Horror” frente a las del género “Musical”.
- **Caso de uso 5:** Observar la diferencia del reparto de los géneros de las películas estrenadas en los años 1950, 1960, 1970, 1980 y 1990.
- **Caso de uso 6:** Comparar las votaciones de las mejores películas del género “action” de la década de 1960 con las de la década de 1980.
- **Caso de uso 7:** Estudiar las recomendaciones que realiza el algoritmo “ItemKnn” a los usuarios 10, 20, 30, 40 y 50 y compararlas con las películas mejor valoradas por cada uno de ellos. Filtrar las películas para que solo se incluyan las del género “Action”.
- **Caso de uso 8:** observar los valores de las métricas de evaluación aplicadas a todas las métricas de recomendación y mostrarlas en forma de gráfica de líneas.

6. Resultados de las pruebas

En este apartado se ejecutarán las pruebas propuestas en el punto 4.3 “Pruebas de sistema” y se comprobará que los resultados obtenidos son correctos. Además, con el fin de llevar a cabo el caso de uso correspondiente a cada prueba, se realizará un breve estudio con las conclusiones que se extraen de la información proporcionada por la aplicación en cada una.

- **Prueba 1: visualizar en un diagrama de barras la evolución del número total de películas estrenadas agrupadas por décadas.**
  La información mostrada en el gráfico muestra un aumento considerable del número de películas estrenadas en cada década que se dispara a partir de la década de los 90.
- **Prueba 2: visualizar en un diagrama de barras la evolución del número de películas del género “thriller” agrupadas por décadas.**
  En esta ocasión se repite el ascenso brusco del número de películas estrenadas en la década de los 90, pero la tendencia con el resto de décadas no ha sido tan clara como en el caso anterior. En esta gráfica, el número de películas estrenadas varía en los distintos períodos de tiempo de forma desigual.
- **Prueba 3: visualizar en un diagrama de barras el número de votaciones que han recibido las películas pertenecientes a los géneros “action”, “adventure” o “animation” agrupados por décadas en función de la fecha de estreno de la película.**
  El número de votaciones tiende a crecer a medida que se va hacia décadas más recientes, lo cual quiere decir que las películas recientes son más vistas que las más antiguas.
- **Prueba 4: Comparar el número de votaciones realizadas por usuarios masculinos y femeninos en películas estrenadas en 1998 pertenecientes al género “Horror” frente a las del género “Musical”.**
  Observando los resultados, se ve una clara tendencia por parte del género masculino a realizar más votaciones que el género femenino. Esta diferencia es todavía más clara al comparar las películas del género “Horror”, donde las mujeres no llegan a realizar ni un cuarto de las votaciones totales, con las del género “Musical”, donde el porcentaje de votaciones realizadas por mujeres ronda el 33% del total.
- **Prueba 5: Observar la diferencia del reparto de los géneros de las películas estrenadas en los años 1950, 1960, 1970, 1980 y 1990.**
  La evolución del reparto de los géneros de las películas estrenadas en esos años permite llegar a la conclusión de que a medida que ha ido avanzando el tiempo, ha aumentado la variedad de géneros de las películas estrenadas. Por lo que no solo se han estrenado más películas, sino que son de temas más diferentes.

- **Prueba 6: Comparar las votaciones de las mejores películas del género "action" de la década de 1960 con las de la década de 1980.**
  En este caso, además de comprobar que en la década de los años 60 se estrenaron menos películas que en los 80, se puede observar que las películas de la década de 1980 están en general mejor valoradas que las de 1960 gracias a los colores de los cuadros de cada treemap.
- **Prueba 7: Estudiar las recomendaciones que realiza el algoritmo "ItemKnn" a los usuarios 10, 20, 30, 40 y 50 y compararlas con las películas mejor valoradas por cada uno de ellos. Filtrar las películas para que solo se incluyan las del género "Action".**
  Para realizar el análisis en primer lugar se han estudiado las películas del género "Action" mejor valoradas para cada usuario. Después, observando los resultados de la visualización se han encontrado las siguientes coincidencias: el usuario 20 y el 30 tienen dos películas en común entre sus favoritas, por lo tanto, es lógico pensar que se le puedan hacer recomendaciones a cada uno basadas en los gustos del otro. Esta teoría se confirma al ver que al usuario 20 se le recomienda "Titanic (1997)", valorada con un 5 por el usuario 30 y que al usuario 30 se le recomienda "Braveheart (1995)", a la que el usuario 20 le puso de valoración un 5.

- **Prueba 8: observar los valores de las métricas de evaluación aplicadas a todas las métricas de recomendación y mostrarlas en forma de gráfica de líneas.**
  Observando el comportamiento de cada métrica en las gráficas se comprueba que el mejor recomendador de los incluidos en el sistema es el que se basa en ItemKNN.

Mediante la observación de los resultados de las pruebas y la comparación de dichos resultados con los valores de la base de datos, se ha confirmado el correcto funcionamiento de la aplicación en todos los casos de uso. Si se desea ver los resultados de las visualizaciones en cada prueba solo es necesario acudir a "Anexo E: Visualizaciones generadas por la aplicación en cada una de las pruebas de sistema".

7. Conclusiones y trabajo futuro

7.1 Conclusiones

A lo largo del desarrollo de este proyecto se ha estudiado el funcionamiento de diversas tecnologías que, sin estar directamente relacionadas entre sí, se han utilizado juntas para formar un sistema completo y funcional. Mediante la combinación de estas tecnologías se ha construido una herramienta web que permite a los usuarios entender conjuntos de datos y estudiar y evaluar el comportamiento de distintos sistemas de recomendación.
Las tecnologías principales utilizadas en la implementación del sistema y en las que más se ha tenido que profundizar son dos:

- Sistemas de recomendación. Se ha realizado un estudio exhaustivo del funcionamiento de los algoritmos de recomendación de datos para comprender mejor la teoría que hay detrás de ellos. Además del funcionamiento de los algoritmos, se han investigado los diferentes tipos que existen, las métricas de evaluación que se pueden aplicar sobre ellos y la forma en que el comportamiento de unos y otros varía al aplicarse sobre distintos conjuntos de datos. Mediante este proceso se han adquirido unos conocimientos que posteriormente han podido aplicarse en la elección de las tecnologías que mejor se adaptan a las necesidades del proyecto.
- Tecnologías de visualización de datos. Concretamente, se han investigado métodos de representación de datos en páginas web. El principal problema al buscar una tecnología de este tipo era encontrar aquella que permitiera generar visualizaciones dinámicas, que no obligaran a recargar la página web cuando el usuario desease modificar los datos. La solución se encontró con la librería de JavaScript D3js, que además de ofrecer métodos de representación de gráficas, implementa un sistema de enlazado de datos con elementos HTML que permite personalizar las vistas a gusto del programador.

Siendo estas dos las tecnologías principales en las que se basa el proyecto, ha sido necesario aplicar también un conjunto de conocimientos que merecen mención. Algunos de dichos conocimientos habían sido vistos durante el transcurso de la carrera pero otros se han tenido que aprender por cuenta propia:

- Framework Ruby on Rails: uno de los sistemas de desarrollo web más demandados y utilizados a día de hoy. Dado que al principio del proyecto no se conocía demasiado su funcionamiento, ha sido necesario un proceso de aprendizaje previo. Tras el desarrollo de la aplicación, se puede concluir que es un framework altamente recomendable para este tipo de sistemas.
- SQLite y bases de datos relacionales: es uno de los conocimientos estudiados en la carrera que más se precisa dominar para el mundo profesional. A pesar del auge de otras alternativas, las bases de datos relacionales siguen siendo las elegidas para la mayor parte de sitios web.

- Patrones de diseño: como ya se ha mencionado, uno de las ventajas de Ruby on Rails es que obliga a seguir la metodología de Modelo-Vista-Controlador, muy útil en el desarrollo de aplicaciones de este tipo. Además, se podría considerar un patrón de diseño la metodología seguida en la implementación de las funciones de creación, actualización y borrado de gráficas, cuyo objetivo principal era evitar repetir código innecesariamente.
- Diseño y análisis de proyectos: se ha precisado de un ejercicio de abstracción y objetividad para poder limitar las funcionalidades que se querían incluir en la aplicación. El análisis de los requisitos del sistema ha supuesto uno de los puntos críticos, ya que sirve tanto para darle forma a la aplicación durante el desarrollo como para comprobar si se han cumplido los objetivos al final.

En resumen, una vez más queda patente la importancia de un buen proceso de estudio y análisis de las diferentes tecnologías disponibles al enfrentarse al desarrollo de un proyecto. Durante dicho proceso, se van adquiriendo conocimientos y habilidades destacables que harán que el resultado final del proyecto sea mejor, más eficiente y menos costoso.

7.2 Trabajo futuro

Durante el proceso de diseño e implementación, fueron surgiendo ideas y mejoras aplicables al proyecto que hubo que ir descartando en favor de otras funcionalidades más necesarias para una primera versión de la aplicación. Algunas de estas ampliaciones del sistema son:

- Inclusión de nuevos tipos de visualizaciones en la aplicación. Dada la variedad de gráficas predefinidas que proporciona la librería D3js y la posibilidad de implementar tipos de gráficas personalizadas, existen multitud de opciones que pueden ser incorporadas a la aplicación. En este sentido, la imaginación del desarrollador es el límite.
- Posibilidad de trabajar con distintos conjuntos de datos. El diseño de la aplicación y de la base de datos se ha hecho pensando en la posibilidad de añadir nuevas tablas con más conjuntos de datos que sigan el mismo esquema que MovieLens (ítems, usuarios, ratings de usuarios a los ítems, información extra sobre los ítems, etc). La idea sería que el usuario eligiera el conjunto de datos del que quiere realizar un estudio al acceder en la aplicación.
- Nuevos recomendadores. Dada la variedad de algoritmos que implementa la librería de MyMediaLite, existen muchos recomendadores que pueden aplicarse a los datos de la aplicación para comparar su funcionamiento y rendimiento. Incluso cabría la posibilidad de incorporar algoritmos de recomendación basados en predicción de ítems.
- Carga de datos procedentes de recomendadores propios. El hecho de tener los resultados de las recomendaciones precargados en la base de datos, hace que sea posible que los usuarios ejecuten sus propios algoritmos de recomendación sobre el conjunto de datos y tengan la posibilidad de subirlos a la aplicación. Ofreciéndoles una herramienta con la que comparar de manera muy visual su rendimiento con el del resto de algoritmos.

8. Referencias

[1] Joonseok Lee, Mingxuan Sun, and Guy Lebanon, "A Comparative Study of Collaborative Filtering Algorithms," May 2012.

[2] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, *Recommender Systems Handbook*, 1st ed. New York, USA, 2010.

[3] Gediminas Adomavicius and Alexander Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-theArt and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749, Junio 2005.

[4] (2012) Recommender Systems. [Online]. http://recommender-systems.org/content-based-filtering/

[5] Sheng Wen, "Recommendation System Base on Collaborative Filtering," Diciembre 2008.

[6] IBM. (2015, Junio) IBM developerWorks. [Online]. http://www.ibm.com/developerworks/library/os-recommender1/

[8] Carleton College, Northfield, MN. Model-based recommendation systems. [Online]. http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/modelbased.html

[7] Carleton College, Northfield, MN. Memory-based algorithms. [Online]. http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/memorybased.html

[9] Sergio Manuel Galán Nieto, "Filtrado Colaborativo y Sistemas de Recomendación," Inteligencia de Redes de Comunicaciones, Universidad Carlos III de Madrid, Madrid,.

[10] Michail D. Ekstrand, John T. Riedl, and Joseph A. Konstan, Collaborative Filtering Recommender Systems, 2010.

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, "Item-Based Collaborative Filtering Recommendation Algorithims," GroupLens Research Group/Army HPC Research Center Department of Computer Science and Engineering , University of Minnesota, Minneapolis,.

[12] Carleton College, Northfield, MN. Item-based collaborative filtering. [Online]. http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html

[13] Iván López Espejo, "Aprendizaje para Clasificación con Factorización Matricial Basado en Listwise para Filtrado Colaborativo," 2012.

[14] Sergio Luján Mora, *Programación de aplicaciones web: historia, principios básicos y clientes web*. Alicante, Alicante, España: Editorial Club Universitario, 2002.

[15] David Heinemeier Hansson. Ruby on Rails Framework. [Online]. http://rubyonrails.org/

[16] Rails Guides. [Online]. http://guides.rubyonrails.org/getting_started.html

[18] Microsoft. Microsoft Developer Network. [Online]. https://msdn.microsoft.com/en-us/library/ff649643.aspx

[17] Open Source Initiative. Open Source Initiative. [Online]. http://opensource.org/licenses/MIT

[19] PHP. [Online]. http://php.net/

[20] Python Software Foundation. (2015, Junio) Python web page. [Online]. https://www.python.org/

[21] Django Software Foundation. The Django Project. [Online]. https://www.djangoproject.com/
[22] Oracle Corporation. MySQL. [Online]. https://www.mysql.com/
[23] SQLite. [Online]. https://www.sqlite.org/
[24] The PostgreSQL Global Development Group. PostgreSQL. [Online]. http://www.postgresql.org/
[25] MyMediaLite. [Online]. http://www.mymedialite.net/documentation/index.html
[26] zenogantner. GitHub. [Online]. https://github.com/zenogantner/MyMediaLite
[28] IronRuby Community. IronRuby. [Online]. http://ironruby.net/
[27] The Mono Project. [Online]. http://www.mono-project.com/
[29] David Celis. GitHub. [Online]. https://github.com/davidcelis/recommendable
[30] Paul Asmuth. GitHub. [Online]. https://github.com/paulasmuth/recommendify
[31] W3Schools. w3schools. [Online]. http://www.w3schools.com/html/
[32] W3Schools. w3schools. [Online]. http://www.w3schools.com/css/
[33] W3Schools. w3schools. [Online]. http://www.w3schools.com/js/
[34] The jQuery Foundation. jquery. [Online]. https://jquery.com/
[35] The jQuery Foundation. jQuery UI. [Online]. https://jqueryui.com/
[36] Mike Bostock. d3js. [Online]. http://d3js.org/
[38] GroupLens. MovieLens. [Online]. http://grouplens.org/datasets/movielens/
[37] Mike Bostock. GitHub. [Online]. https://github.com/mbostock/d3
[39] IMDb.com, Inc. IMDb. [Online]. http://www.imdb.com/
[40] David Heinemeier Hansson. Ruby on Rails guider. [Online]. http://guides.rubyonrails.org/command_line.html

Anexos técnicos

Anexo A: Carga de un fichero de MovieLens en la base de datos

En la siguiente imagen se puede ver el código correspondiente a la carga del fichero de ratings en la base de datos de la aplicación. Como se puede observar, por cada línea del fichero, se crea una instancia de la clase “Rating”, se rellenan sus campos y se almacena, todo mediante código Ruby, sin necesidad de escribir código SQL.

```ruby
desc "Load ratings in database"
task :load_ratings => :environment do
  File.open("movielens_files/u.data", "r").each_line do |line|
    array = line.split
    rating = Rating.new
    rating.user_id = array[0]
    rating.movie_id = array[1]
    rating.rating = array[2]
    rating.timestamp = Time.at(array[3].to_i).to_datetime
    rating.save
  end
  puts "Ratings load complete"
end
```

Anexo B: Scripts de generación de recomendaciones.

En las imágenes adjuntadas a continuación, se puede ver el código correspondiente a los scripts que generan las recomendaciones del algoritmo de UserKNN y que posteriormente las guardan en la base de datos de la aplicación.
En primer lugar se ejecuta un script que hace una llamada a la herramienta IronRuby pasándole el fichero que se desea que ejecute como parámetro.

```
desc "Execute knn rating prediction script"
task :predict_ratings_knn => :environment do
  sh "ir prediction_scripts/complete_knn_prediction.rb #{User.count.to_s} #{Movie.count.to_s}"
end
```

El script que se le pasa a IronRuby es el que contiene las llamadas a las funciones de la librería de MyMediaLite, y en este caso es el siguiente:

```
#!/usr/bin/env ir

require '/Users/Alejandro/Downloads/MyMediaLite-3.10/lib/mymedialite/MyMediaLite.dll'

# load the data
train_data = MyMediaLite::IO::RatingData.Read("movielens_files/u1.base")
test_data = MyMediaLite::IO::RatingData.Read("movielens_files/u1.test")

# set up the recommender
recommender = MyMediaLite::RatingPrediction::UserKNN.new()
recommender.Ratings = train_data
recommender.Train()


File.open('prediction_outputs/complete_knn_prediction.out','w') do |s|
  for i in 1..ARGV[0].to_i
    for j in 1..ARGV[1].to_i
      array = train_data.TryGetIndex(i, j)
      if !array[0]
        prediction = recommender.Predict(i, j).to_s
        s.puts "#{i}\t#{j}\t" + prediction
      end
    end
    puts "Prediccion del usuario #{i} terminada"
  end
end
```

Por último, se ejecuta el script que lee el fichero “complete_knn_prediction.out”, que contiene todas las recomendaciones generadas para introducirlas en la base de datos:

```ruby
desc "Load knn predictions in database"
task :load_knn_predictions => :environment do
  recommender_type = RecommenderType.where(:name => "KNN").first
  count = RecommenderType.where(:name => "KNN").count
  if count == 0
    recommender_type = RecommenderType.new
    recommender_type.name = "KNN"
    recommender_type.function_name = "UserKNN"
    recommender_type.save
    puts "New recommender type created"
  end

  File.open("prediction_outputs/complete_knn_prediction.out").each_line do |line|
    array = line.split(%r{[|\t]})
    prediction = Prediction.new
    prediction.user_id = array[0]
    prediction.movie_id = array[1]
    prediction.rating = array[2]
    prediction.recommender_type_id = recommender_type.id
    prediction.save
    puts "Prediction #{prediction.rating} for user #{prediction.user_id} and movie #{prediction.movie_id} saved"
  end
  puts "KNN predictions load complete"
end
```

Anexo C: Scripts de evaluación de los algoritmos de recomendación

Para la evaluación de los algoritmos de recomendación también se han implementados tres algoritmos, siguiendo el mismo patrón que para la generación de los resultados de recomendaciones.
En primer lugar se ejecuta una función que llama a la herramienta IronRuby para que ejecute el script que evalúa los recomendadores.

```
desc "Execute recommenders tests and write results in a file"
task :execute_recommenders_results_script => :environment  do
  sh "ir prediction_scripts/recommenders_results_script.rb"
end
```

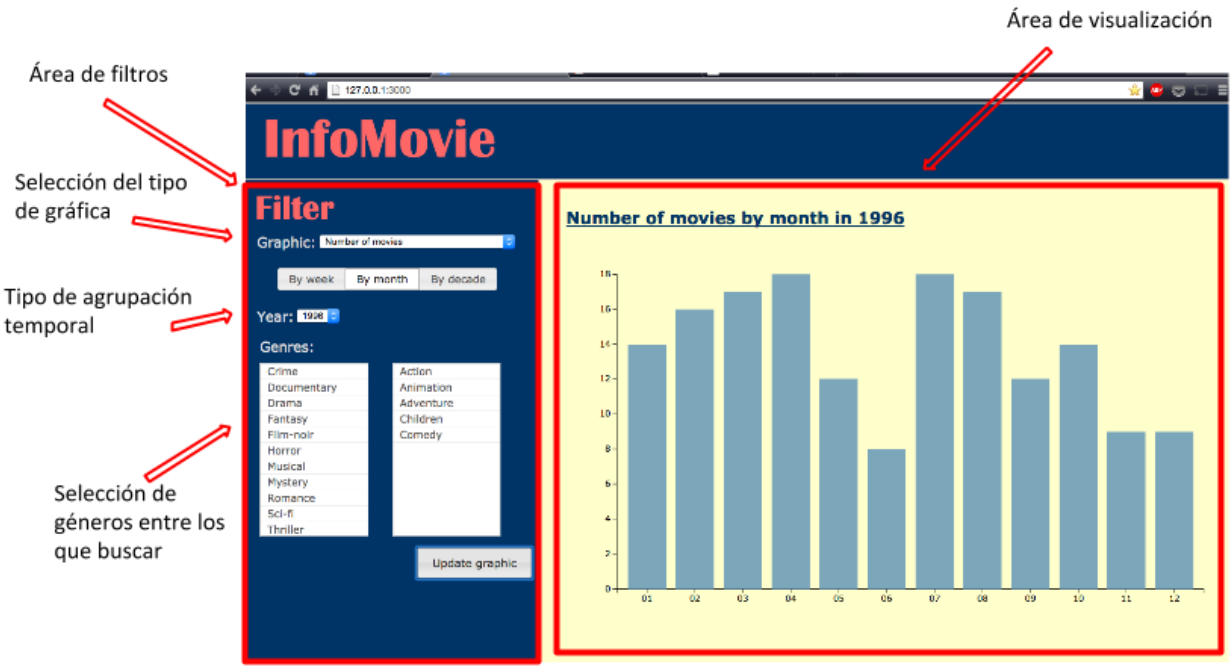
El script "recommenders_results_script.rb" es el que contiene las llamadas a las funciones de la librería de MyMediaLite para la evaluación de recomendadores. En este caso, dado que la evaluación es más rápida que la generación de todos los ratings, se evalúan todos los recomendadores de una sola vez.

```
#!/usr/bin/env ir

require '/Users/Alejandro/Downloads/MyMediaLite-3.10/lib/mymedialite/MyMediaLite.dll'

# load the data
train_data = MyMediaLite::IO::RatingData.Read("movielens_files/u1.base")
test_data = MyMediaLite::IO::RatingData.Read("movielens_files/u1.test")

# set up the base_line_recommender
base_line_recommender = MyMediaLite::RatingPrediction::UserItemBaseline.new()
base_line_recommender.Ratings = train_data
base_line_recommender.Train()

# set up the user_knn_recommender
user_knn_recommender = MyMediaLite::RatingPrediction::UserKNN.new()
user_knn_recommender.Ratings = train_data
user_knn_recommender.Train()

# set up the item_knn_recommender
item_knn_recommender = MyMediaLite::RatingPrediction::ItemKNN.new()
item_knn_recommender.Ratings = train_data
item_knn_recommender.Train()

# set up the matrix_factorization_recommender
matrix_factorization_recommender = MyMediaLite::RatingPrediction::MatrixFactorization.new()
matrix_factorization_recommender.Ratings = train_data
matrix_factorization_recommender.Train()

# set up the global_average_recommender
global_average_recommender = MyMediaLite::RatingPrediction::GlobalAverage.new()
global_average_recommender.Ratings = train_data
global_average_recommender.Train()

# measure the accuracy on the test data set
base_line_eval_results = MyMediaLite::Eval::Ratings::Evaluate(base_line_recommender, test_data)
user_knn_eval_results = MyMediaLite::Eval::Ratings::Evaluate(user_knn_recommender, test_data)
item_knn_eval_results = MyMediaLite::Eval::Ratings::Evaluate(item_knn_recommender, test_data)
matrix_factorization_eval_results = MyMediaLite::Eval::Ratings::Evaluate(matrix_factorization_recommender, test_data)
global_average_eval_results = MyMediaLite::Eval::Ratings::Evaluate(global_average_recommender, test_data)

File.open('prediction_outputs/recommenders_results.out','w') do |s|
  s.puts "Baseline\t" + base_line_eval_results["RMSE"].to_s + "\t" + base_line_eval_results["MAE"].to_s + "\t"
    + base_line_eval_results["NMAE"].to_s + "\t" + base_line_eval_results["CBD"].to_s
  s.puts "User Knn\t" + user_knn_eval_results["RMSE"].to_s + "\t" + user_knn_eval_results["MAE"].to_s + "\t"
    + user_knn_eval_results["NMAE"].to_s + "\t" + user_knn_eval_results["CBD"].to_s
  s.puts "Item Knn\t" + item_knn_eval_results["RMSE"].to_s + "\t" + item_knn_eval_results["MAE"].to_s + "\t"
    + item_knn_eval_results["NMAE"].to_s + "\t" + item_knn_eval_results["CBD"].to_s
  s.puts "Matrix factorization\t" + matrix_factorization_eval_results["RMSE"].to_s + "\t" + matrix_factorization_eval_results["MAE"].to_s + "\t"
    + matrix_factorization_eval_results["NMAE"].to_s + "\t" + matrix_factorization_eval_results["CBD"].to_s
  s.puts "Global average\t" + global_average_eval_results["RMSE"].to_s + "\t" + global_average_eval_results["MAE"].to_s + "\t"
    + global_average_eval_results["NMAE"].to_s + "\t" + global_average_eval_results["CBD"].to_s
end
```

Por último, se lee el fichero “recommender_results.out” línea a línea para almacenar todas las métricas de evaluación para cada recomendador.

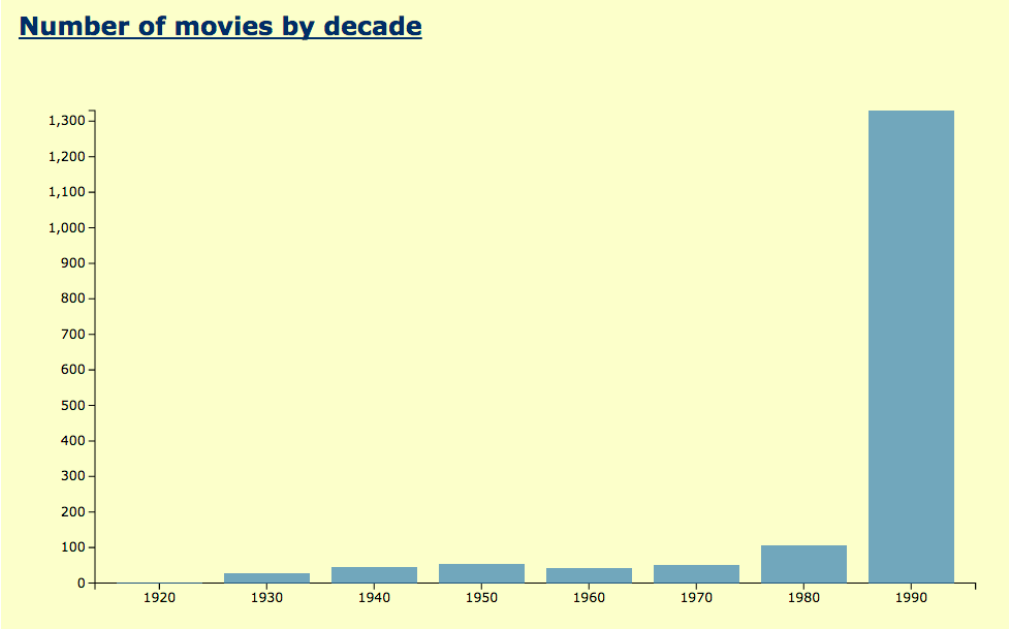
```
desc "Load recommenders results in database"
task :load_recommenders_results => :environment do
  File.open("prediction_outputs/recommenders_results.out").each_line do |line|
    array = line.split(%r{[|\t]})
    recommender_type = RecommenderType.where(:name => array[0]).first
    recommender_result = RecommenderResult.new
    recommender_result.recommender_type_id = recommender_type.id
    recommender_result.rmse = array[1]
    recommender_result.mae = array[2]
    recommender_result.nmae = array[3]
    recommender_result.cbd = array[4]
    recommender_result.save
    puts "Resultado de recomendador " + recommender_type.name + " guardado en la base de datos."
  end
end
```

Anexo D: Esquema de la vista de la aplicación

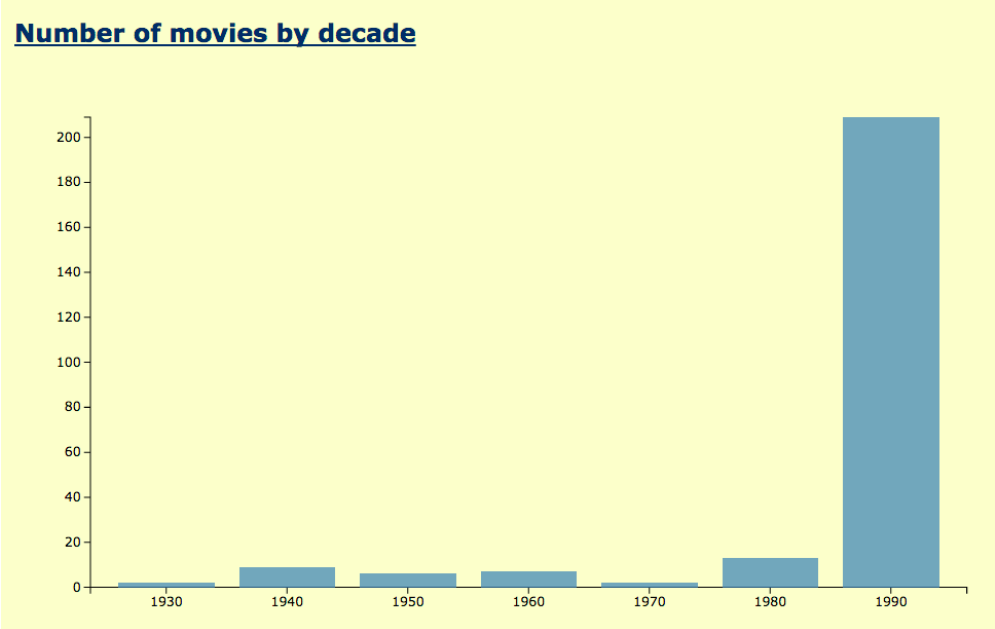


Anexo E: Visualizaciones generadas por la aplicación en cada una de las pruebas de sistema

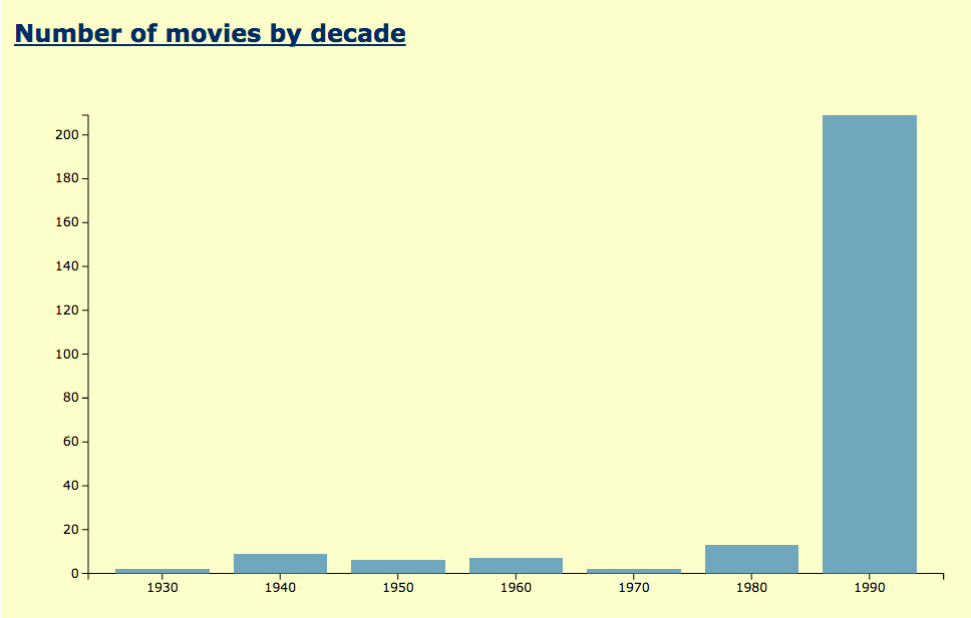
Prueba 1:



Prueba 2:

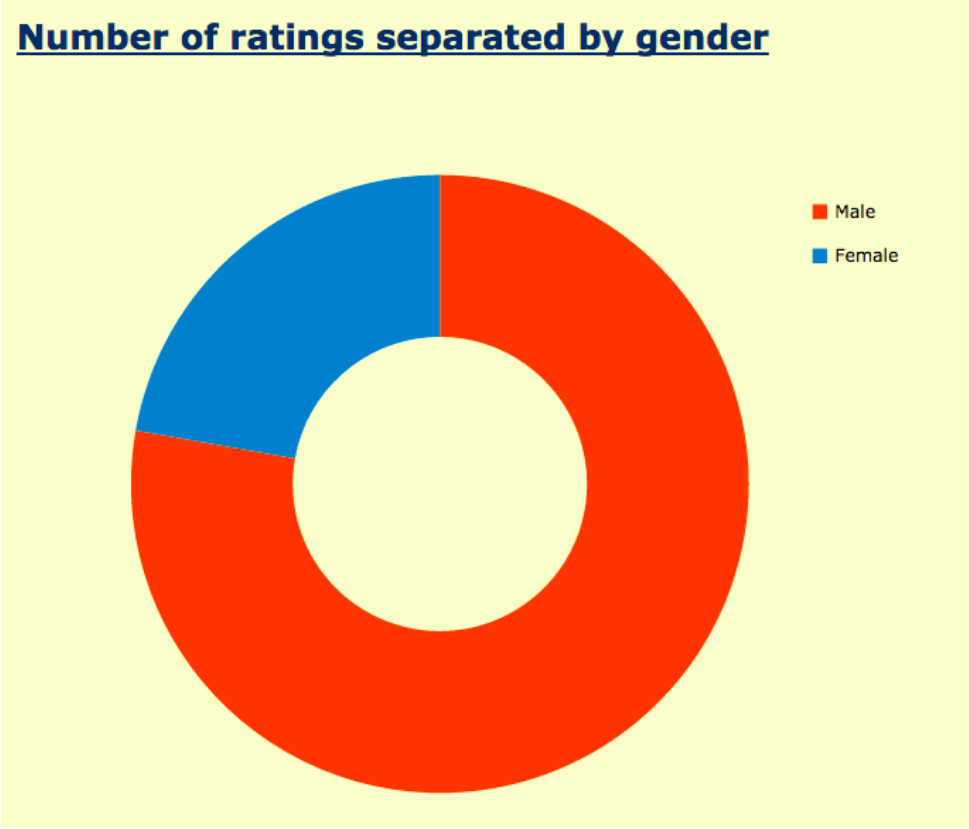


Prueba 3:

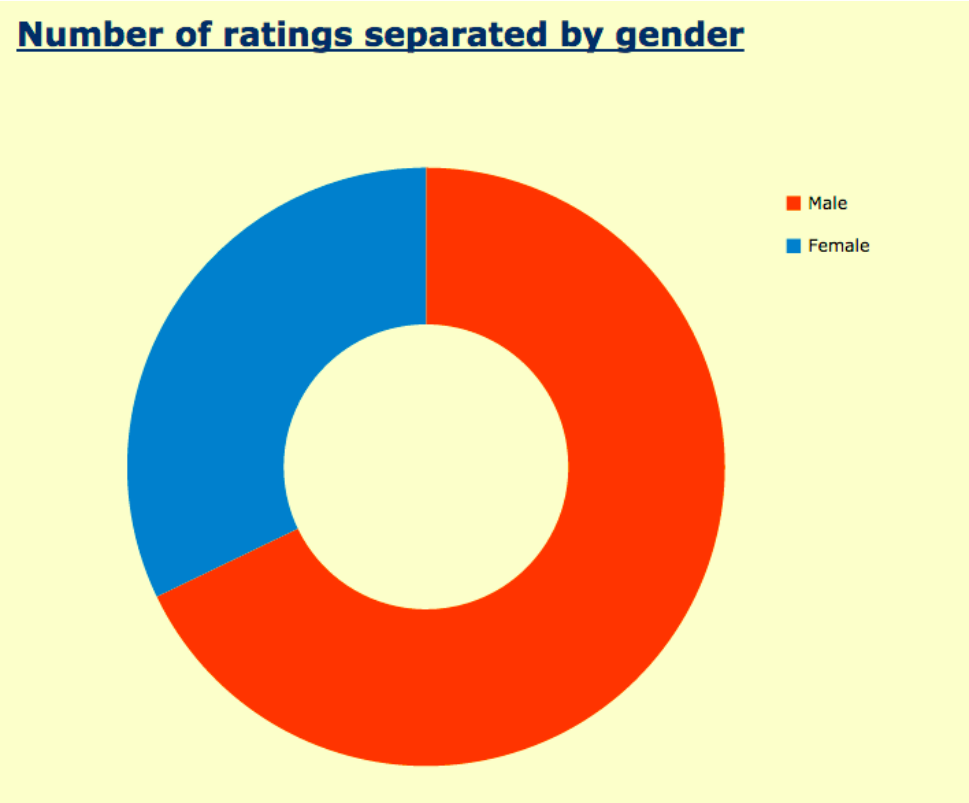


Prueba 4

Proporción de votaciones por sexo en películas del género "Horror".

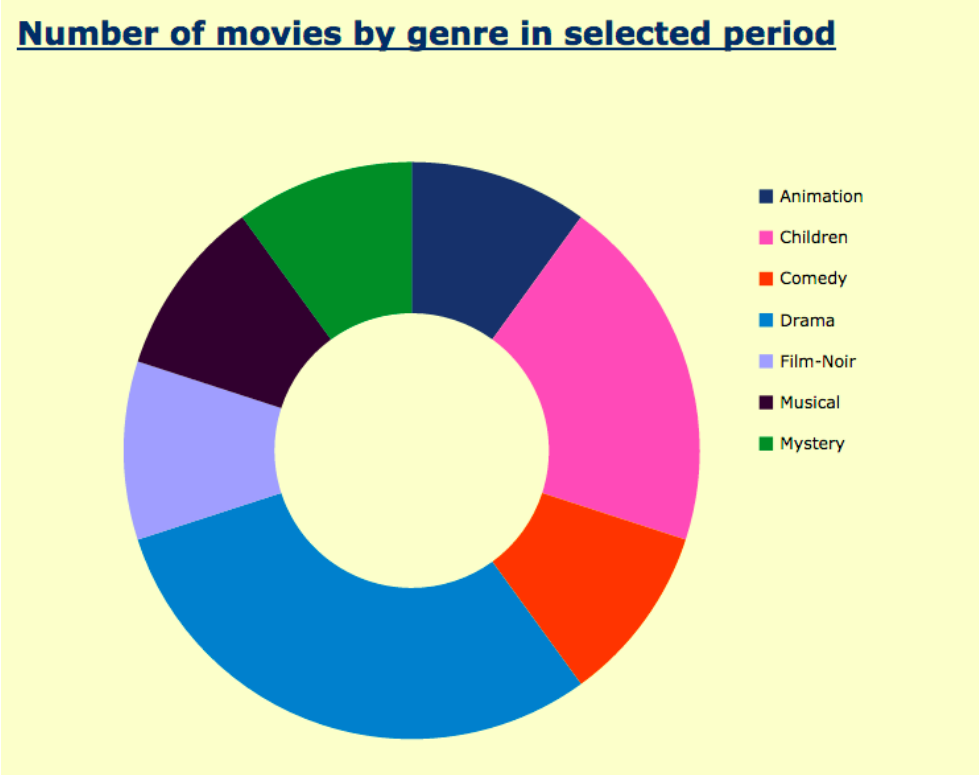


Proporción de votaciones por sexo en películas del género "Musical".

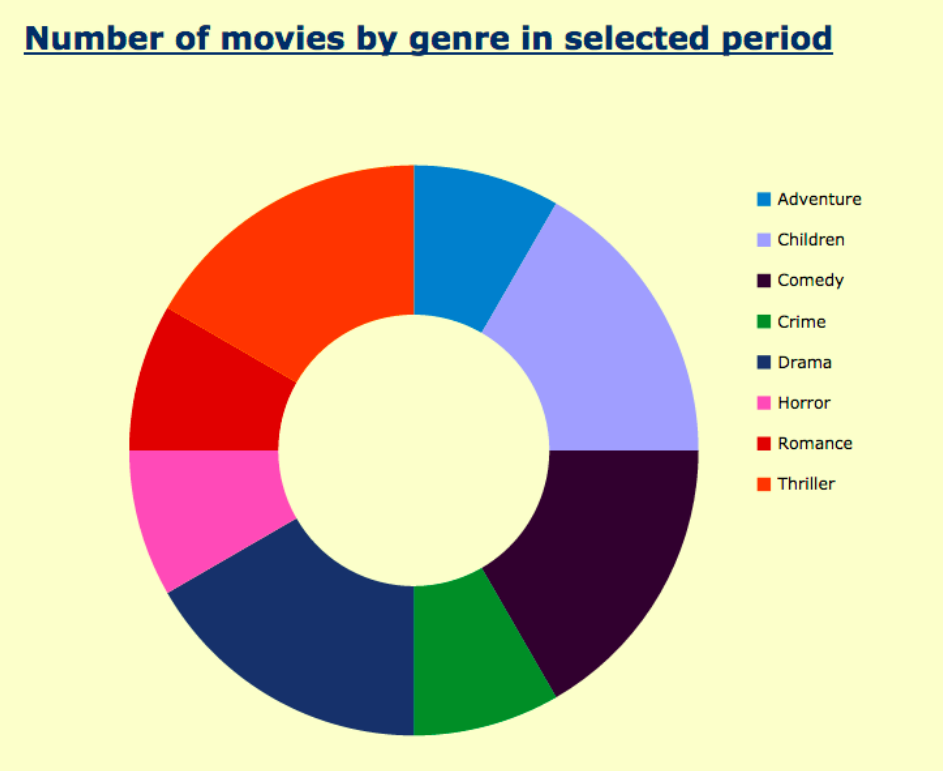


Prueba 5

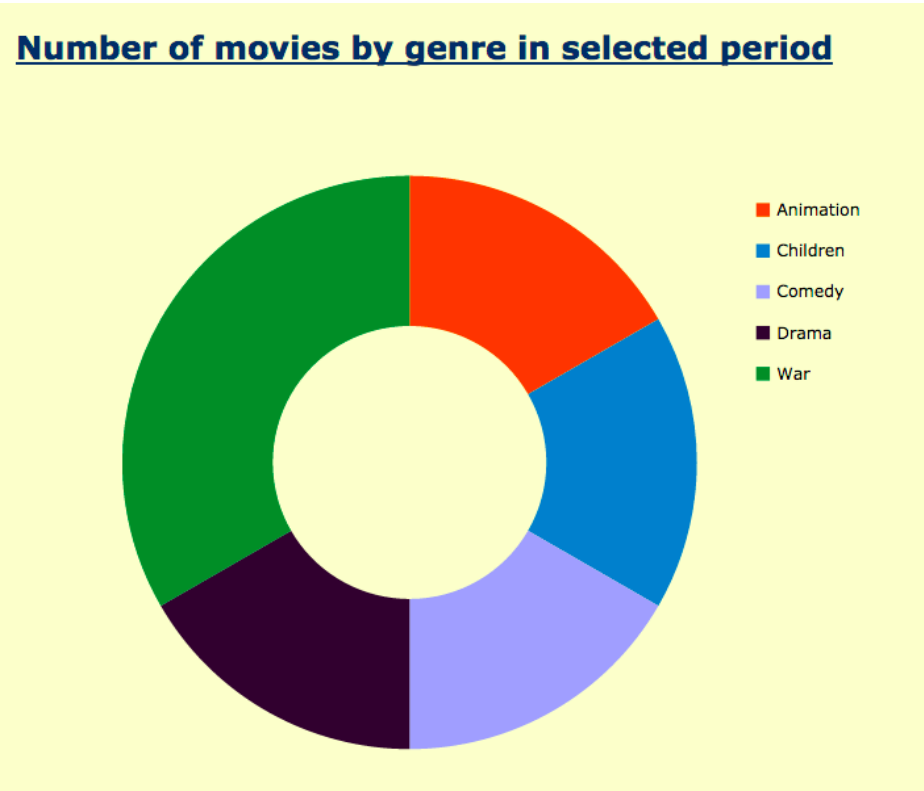
1950



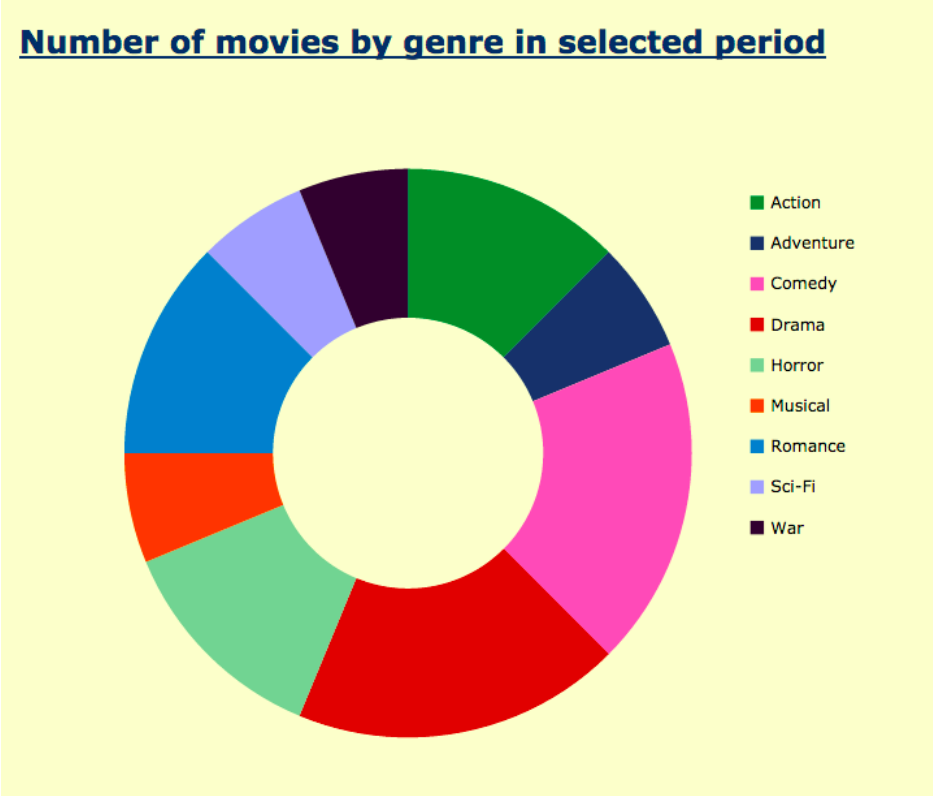
1960


Number of movies by genre in selected period
Adventure
Children
Comedy
Crime
Drama
Horror
Romance
Thriller

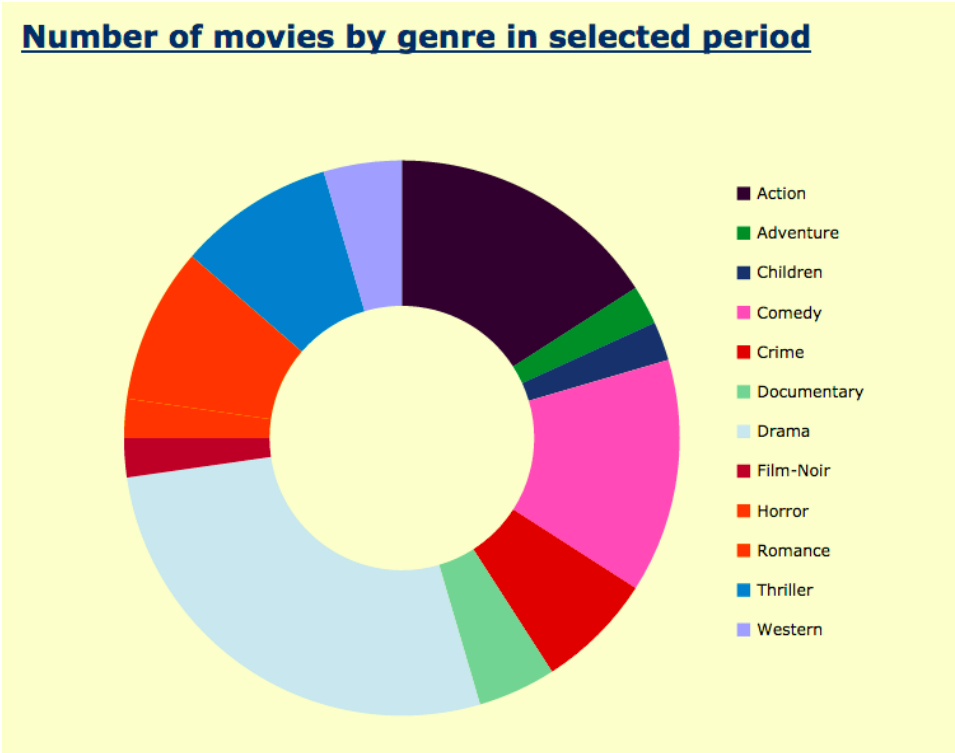
1970


Number of movies by genre in selected period
Animation
Children
Comedy
Drama
War

1980


Number of movies by genre in selected period
Action
Adventure
Comedy
Drama
Horror
Musical
Romance
Sci-Fi
War

1990


Number of movies by genre in selected period
Action
Adventure
Children
Comedy
Crime
Documentary
Drama
Film-Noir
Horror
Romance
Thriller
Western

Prueba 6

Década de los 60

Number of movies by genre in selected period

Movies.1960.Action	
Butch Cassidy and the Sundance Kid (1969)	Good, The Bad and The Ugly, The (1966)
Faster Pussycat! Kill! Kill! (1965)	

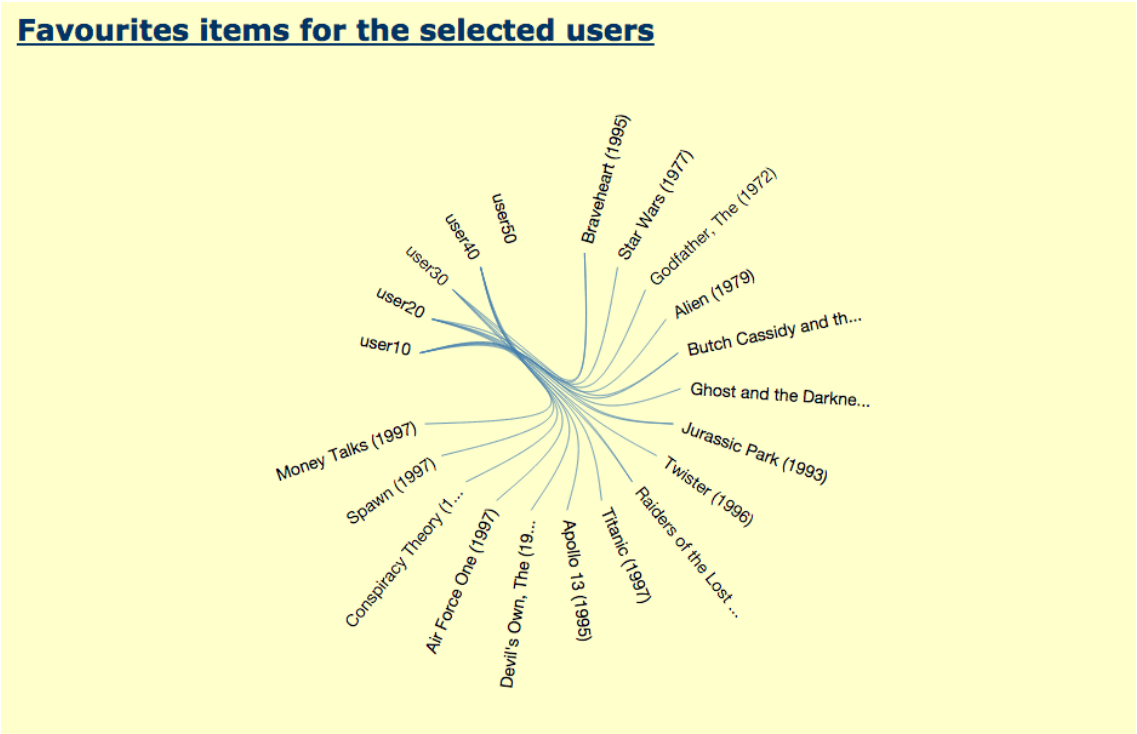
Década de los 80

Number of movies by genre in selected period

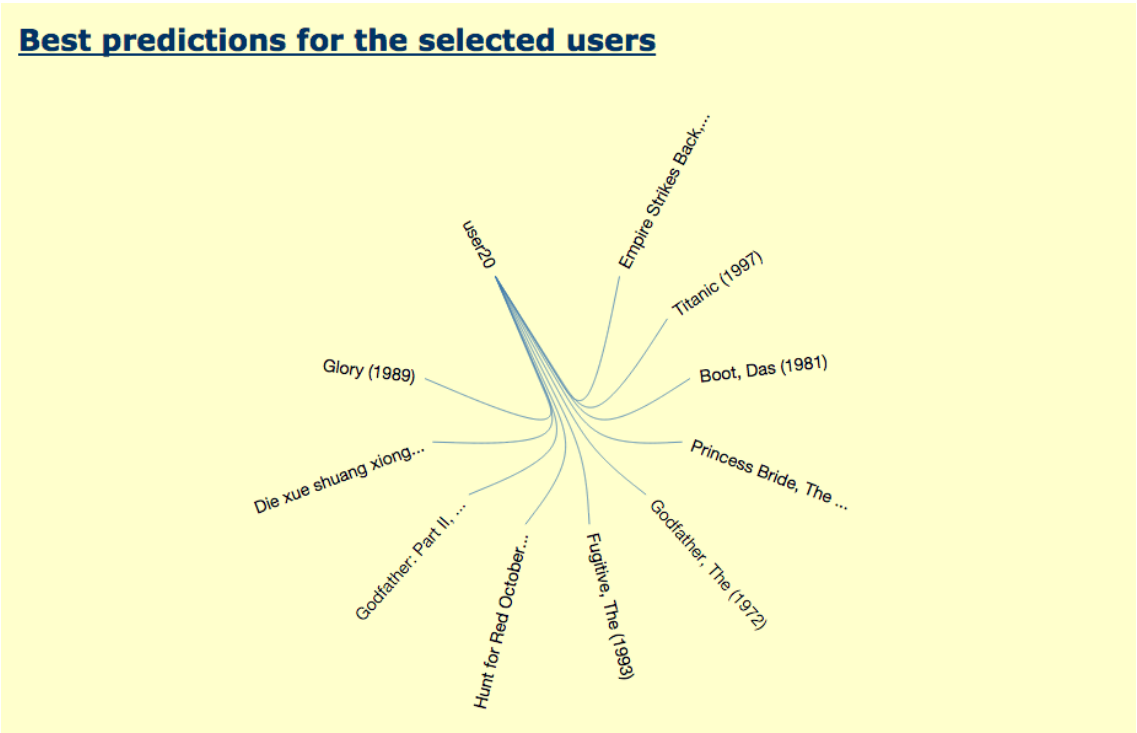
Movies.1980.Action		
Raiders of the Lost Ark (1981)	Empire Strikes Back, The (1980)	Princess Bride, The (1987)
Glory (1989)	Aliens (1986)	Indiana Jones and the Last Crusade (1989)
		Die Hard (1988)
Die xue shuang xiong (Kill...	Terminator, The (1984)	
		Blues Brothers, The (1980)

Prueba 7:

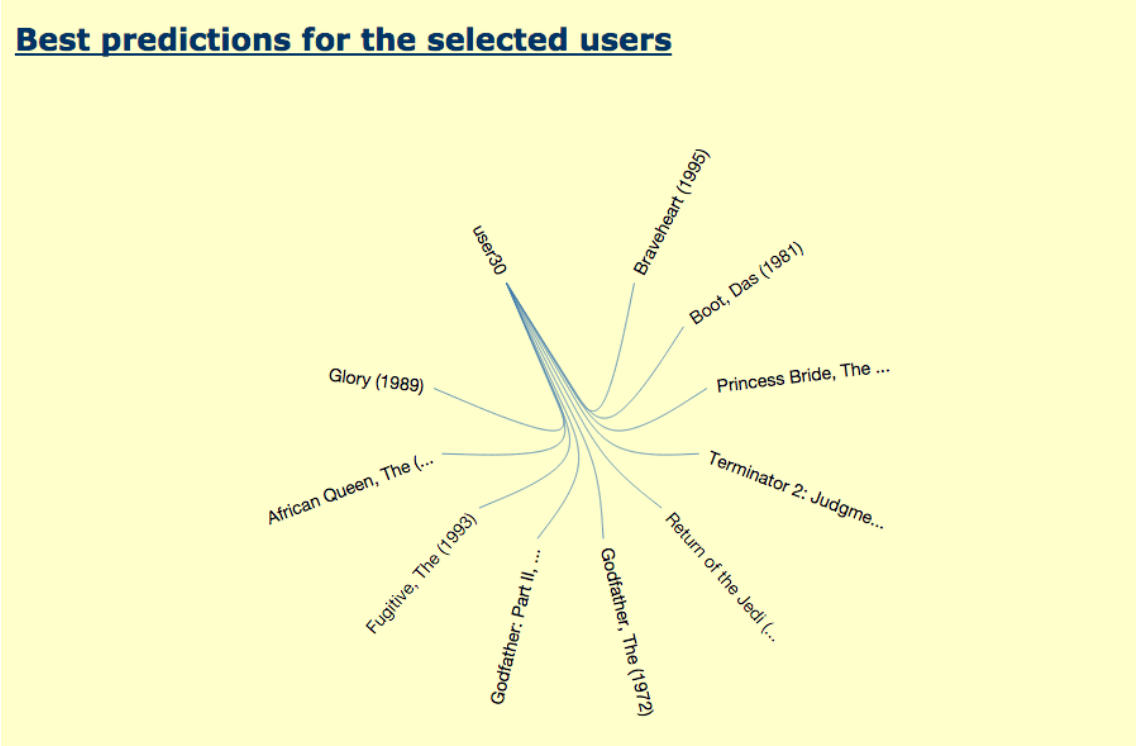
Películas favoritas de los usuarios 10, 20, 30, 40 y 50



Mejores recomendaciones para el usuario 20.



Mejores recomendaciones para el usuario 30.



Prueba 8