Trabajo de Fin de Grado

Sistemas de recomendación en R y aplicación a datos reales

Autora: Alba Lerma Micó

Tutora: María Asunción Martínez Mayoral

Grado en Estadística Empresarial

Facultad de Ciencias Sociales y Jurídicas

Universidad Miguel Hernández

Curso 2023-2024

Agradecimientos

A mis padres, los pilares de mi existencia:

> Mamá, gracias por tu amor incondicional, tu infinita sabiduría y por enseñarme a ser igual de resiliente que tú.
>
> Papá, gracias por enseñarme que cualquier obstáculo se puede superar y acompañarme en cada paso de mi vida.

Vuestra guía y apoyo han sido la luz en mi camino y la fuerza detrás de cada uno de mis logros académicos y laborales. Este trabajo es un reflejo de todo lo que me habéis enseñado y del amor que me habéis dado.

Con todo mi amor y agradecimiento,
Alba

Junio 2024

Biblioteca
UNIVERSITAS Miguel Hernández

ÍNDICE

Resumen.... 4
Palabras clave.... 4
Contextualización y marco teórico.... 4
Objetivos.... 6
Objetivos específicos.... 6
Información disponible.... 7
Metodología.... 9
Análisis exploratorio.... 9
Sistemas de recomendación.... 10
Filtrado Colaborativo.... 10
Filtrado Basado en Contenido.... 11
Filtrado Híbrido.... 12
Evaluación.... 13
Sistemas de recomendación en R.... 16
RecommenderLab.... 19
Algoritmia en RecommenderLab.... 19
Resultados.... 29
Análisis Exploratorio.... 29
Aplicación de RecommenderLab.... 34
Conclusión.... 42
Bibliografía.... 45
Anexos.... 52

Biblioteca
UNIVERSITAS Miguel Hernández

Resumen

Este trabajo fin de grado se plantea con el propósito general de hacer una revisión exhaustiva sobre los sistemas de recomendación, tan popularizados en un sinfín de aplicaciones web a través de las cuales los usuarios escogen productos según sus preferencias, pueden evaluarlos y la aplicación les ofrece o recomienda productos similares para promover su retención y fidelización en la aplicación. Se presentan los fundamentos y alternativas existentes sobre los sistemas de recomendación más comunes y en sus tres enfoques principales: filtrado basado en contenido, filtrado colaborativo y filtrado híbrido. Además, se localizan y describen distintos paquetes estadísticos para resolverlos mediante el lenguaje de programación R, y finalmente se describe en detalle la librería RecommenderLab, la algoritmia que tiene implementada, y el modo de aplicarla sobre unos datos reales para obtener recomendaciones. Se plantea cómo aplicar diversos algoritmos alternativos y compararlos entre sí respecto de las métricas habituales, para analizar y evaluar su rendimiento, esto es, su precisión y eficacia.

Se trabaja sobre la popular base de datos MovieLens, que es descrita inicialmente a través del correspondiente análisis estadístico exploratorio, para después aplicar diferentes tipos de filtrado , obtener las recomendaciones pertinentes y compararlas respecto de las métricas de calidad propuestas.

Palabras clave

Sistemas de recomendación, RecommenderLab, Filtrado Colaborativo, Filtrado Basado en Contenido, Filtrado Híbrido.

Contextualización y marco teórico

En nuestro día a día nos enfrentamos a situaciones que nos hacen tomar decisiones, muchas veces sin siquiera darnos cuenta. Desde posponer la alarma cuando nos despertamos, hasta decidir en qué supermercado harás la compra, pasando por priorizar qué tarea doméstica es necesario hacer antes o qué ropa te vas a poner hoy. A lo largo de nuestra vida nos enfrentamos a una multitud de preguntas que contestamos tomando decisiones.

Hace años, solíamos recurrir a la opinión de expertos o amigos para tomar algunas decisiones, pero en la era actual contamos con una amplia gama de posibilidades adicionales. Actualmente no dependemos únicamente de la opinión del vendedor de la tienda de electrodomésticos, sino que tenemos muchísima información al alcance de nuestra mano y contrastamos precios y opiniones en un simple click. Sitios web como Amazon nos brindan la posibilidad de poder hacer este tipo de comparaciones basándose en recomendaciones de otros usuarios con intereses similares a los nuestros, basados en algoritmos complejos y grandes conjuntos de datos.

Un **sistema de recomendación** es una herramienta que ayuda a los usuario en la toma de decisiones y reduce las opciones disponibles a la vez que destaca las más relevantes. Utiliza las preferencias del usuario y el historial de comportamiento de usuarios similares para priorizar las recomendaciones.

La historia de los sistemas de recomendación es un relato fascinante que abarca décadas de innovación y avance tecnológico. Desde sus comienzos en la década de 1970 hasta la actualidad, los sistemas de recomendación han revolucionado la manera en que interactúan con la información y tomamos decisiones en base a éstos.

En el año 1979, Elaine Rich planteó el primer sistema de recomendación, Grundy (Beel, 2016; Rich, 1979), con el objetivo de sugerir libros a los usuarios según sus preferencias. Su enfoque implicaba recopilar información específica a partir de preguntas con el fin de clasificarlos en grupos de preferencias o "estereotipos" para ofrecerles recomendaciones de libros acordes a su clasificación.

Posteriormente, durante la década de 1990, Jussi Karlgren (Karlgren, 2017) describió otro sistema de recomendación llamado "estantería digital", el cual fue implementado y desarrollado a mayor escala por Karlgren y su equipo a partir de 1994. Este trabajo fue llevado a cabo en grupos de investigación liderado por Pattie Maes y Paul Resnick en el Instituto Tecnológico de Massachusetts (MIT) y Will Hill en Bellcore.

Durante el año 1997, el laboratorio de investigación GroupLens lanzó el proyecto MovieLens y gracias a este conjunto de datos se pudieron entrenar las primeras versiones de los modelos de recomendación. La colaboración de Resnick con GroupLens fue premiada con el ACM (*Association for Computing Machinery*) Software Systems Award en 2010 (Resnick et al., 1994).

Por otro lado, Montaner (Montaner, 2003) ofreció una de las primeras visiones generales de los sistemas de recomendación desde la perspectiva de un agente inteligente. Adomavicius (Adomavicius, 2005) presentó una nueva descripción alternativa de estos sistemas, mientras que Herlocker (Herlocker, 2004) abordó técnicas de evaluación para los mismos. Por otro lado, Joeran Beel (Beel & Langer, 2013) discutió los desafíos de las evaluaciones fuera de línea y realizó estudios de literatura sobre sistemas de recomendación de investigación y sus desafíos.

El cambio de milenio trajo consigo un interés creciente en la evaluación centrada en el usuario, con investigadores proponiendo nuevos enfoques para medir la efectividad de los sistemas de recomendación desde la perspectiva del usuario. Este cambio de paradigma culminó en la celebración de la primera Conferencia ACM en la UMN (*University of Minnesota*) sobre los Sistemas de Recomendación en 2007 (ACM Recommender Systems, 2007), marcando un hito importante en la comunidad académica.

En paralelo, la industria experimentó un rápido crecimiento en la aplicación de sistemas de recomendación gracias a empresas líderes como Net Perceptions reconociendo su valor estratégico en el ámbito del comercio online. La introducción de modelos como la regresión logística (LR) y las Máquinas de Factorización (FMs) proporcionó nuevas herramientas para mejorar la precisión y eficacia de las recomendaciones, abriendo nuevas oportunidades en el campo.

A medida que avanzaba la década, los modelos de recomendación basados en redes neuronales emergieron como una fuerza poderosa tanto en el ámbito académico como en el de la industria. Con sistemas como este se buscaba mejorar la calidad y relevancia de las recomendaciones, especialmente en el comercio electrónico y plataformas de streaming.

Además, se observó un creciente interés en abordar los sesgos en los sistemas de recomendación, con investigadores explorando enfoques basados en la inferencia causal y la equidad en las recomendaciones. Estos esfuerzos reflejan una preocupación por garantizar que los sistemas de recomendación sean justos y éticos para todos los usuarios.

En un mundo cada vez más saturado de información y de opciones, los sistemas de recomendación se han convertido en una herramienta imprescindible para simplificar la toma de decisiones y mejorar exponencialmente la experiencia del usuario. Actualmente la importancia de los sistemas de recomendación radica en su capacidad para filtrar y procesar grandes cantidades de información, analizando las preferencias y comportamientos del usuario con la finalidad de ofrecer sugerencias que se ajusten a sus necesidades y gustos individuales. Este hecho no solo facilita la búsqueda de información y la selección de de productos y servicios, sino que también permite contribuir a aumentar la satisfacción del cliente y fomentar la fidelidad a la empresa o marca.

Los sistemas de recomendación juegan un papel importante en el comercio electrónico y los servicios de streaming online, como Netflix, YouTube y Amazon. Generar la recomendación correcta para el próximo producto, música o película aumenta la retención y la satisfacción del usuario, lo que genera un crecimiento de las ventas y las ganancias. Las empresas que compiten por la fidelidad de sus clientes invierten en sistemas que capturan y analizan las preferencias del usuario y ofrecen productos o servicios con mayor probabilidad de compra personalizado a cada usuario.

Este trabajo tiene como principal enfoque el desarrollo y el análisis de la fiabilidad y utilidad de los sistemas de recomendación, utilizando técnicas como el filtrado colaborativo. Además, cabe destacar que este trabajo está directamente relacionado con el análisis del creciente interés por parte de todas las industrias y la importancia de los sistemas de recomendación en la actualidad, contribuyendo a la mejora de la experiencia del usuario y para brindar soluciones innovadoras a la hora de afrontar los desafíos del mercado.

Objetivos

Planteamos como objetivo general de este trabajo hacer una revisión de los principales sistemas de recomendación existentes, y en particular los programados en librerías del lenguaje de programación R. Profundizamos en la librería RecommenderLab, describiendo su algoritmia y el modo de aplicarla a cualquier base de datos.

Objetivos específicos

Se plantean como objetivos específicos del trabajo:

- Introducir y definir los sistemas de recomendación, sus tipologías, fundamentos y métricas de evaluación.

- Revisar los sistemas de recomendación disponibles e implementados en R.
- Describir en detalle los algoritmos implementados en la librería RecommenderLab de R.
- Describir en detalle las funciones disponibles en dicha librería y el modo de implementarlas sobre una base de datos para conseguir un sistema de recomendación, evaluarlo y compararlo con otros.
- Describir la base de datos MovieLens, disponible en la librería RecommenderLab.
- Aplicar todas las funciones descritas para la implementación, evaluación y comparación de sistemas de recomendación.
- Obtener las conclusiones pertinentes a partir del análisis comparativo de los diversos algoritmos de recomendación empleados.

Información disponible

Partimos de una base de datos generada con información registrada por GroupLens, un laboratorio de investigación de la Universidad de Minnesota que ha recopilado y puesto a disposición pública datos de clasificación de películas en el sitio web MovieLens.

El conjunto de datos completo de MovieLens describe un servicio de recomendación de películas, con clasificaciones de hasta 5 estrellas. Contiene 33.832.162 clasificaciones y 2.328.315 aplicaciones de etiquetas en 86.537 películas. Estos datos fueron creados por 330.975 usuarios entre el 9 de enero de 1995 y el 20 de julio de 2023.

Los usuarios fueron seleccionados al azar para su inclusión. Todos los usuarios seleccionados habían calificado al menos 1 película.

Los datos están contenidos en los archivos genome-scores.csv, genome-tags.csv, links.csv, movies.csv, ratings.csv y tags.csv. A continuación se ofrecen más detalles sobre el contenido y el uso de todos estos archivos.

- **Etiquetar genoma (genome-scores.csv y genome-tags.csv)**
  Este conjunto de datos incluye una copia actual del Tag Genome (Vig et al., 2012). El genoma de etiquetas es una estructura de datos que contiene puntuaciones de relevancia de etiquetas que codifica con qué fuerza las películas exhiben propiedades particulares representadas por etiquetas (atmosféricas, estimulantes, realistas, etc.). El genoma de la etiqueta se calculó utilizando un algoritmo de aprendizaje automático en contenido aportado por el usuario, incluidas etiquetas, calificaciones y reseñas textuales.
  El genoma se divide en dos archivos:
  - El archivo genome-scores.csv contiene datos de relevancia de etiquetas de película en el siguiente formato:
    - movieId
    - tagId
    - relevancia
  - El segundo archivo, genome-tags.csv, proporciona las descripciones de las etiquetas para los ID de etiqueta en el archivo del genoma, en el siguiente formato:

- ID de etiqueta
- etiqueta

Los valores de tagId se generan cuando se exporta el conjunto de datos, por lo que pueden variar de una versión a otra de los conjuntos de datos de MovieLens.

- **Estructura del archivo de datos de enlaces (links.csv)**

  Los identificadores que se pueden utilizar para vincular a otras fuentes de datos de películas se encuentran en el archivo links.csv. Cada línea de este archivo después de la fila del encabezado representa una película y tiene el siguiente formato:
  - movieId. Identificador de películas utilizado por https://movielens.org. Por ejemplo, la película Toy Story tiene el enlace https://movielens.org/movies/1.
  - imdbId. Identificador de películas utilizado por http://www.imdb.com. Por ejemplo, la película Toy Story tiene el enlace http://www.imdb.com/title/tt0114709/.
  - tmdbId. Identificador de películas utilizado por https://www.themoviedb.org. Por ejemplo, la película Toy Story tiene el enlace https://www.themoviedb.org/movie/862.

- **Estructura del archivo de datos de películas (movies.csv)**

  La información de la película está contenida en el archivo movies.csv. Cada línea de este archivo después de la fila del encabezado representa una película y tiene el siguiente formato:
  - movieId
  - título
  - géneros

  Los títulos de las películas se ingresan manualmente o se importan desde https://www.themoviedb.org/ e incluyen el año de lanzamiento entre paréntesis. Pueden existir errores e inconsistencias en estos títulos.

  Los géneros son una lista separada por barras verticales y se seleccionan entre los siguientes:
  - Acción
  - Aventura
  - Animación
  - Para niños
  - Comedia
  - Delito
  - Documental
  - Drama
  - Fantasía
  - Cine negro
  - Horror
  - Musical
  - Misterio
  - Romance
  - Ciencia ficción
  - Suspenso

- Guerra
- Occidental
- (no hay géneros listados)

- **Estructura del archivo de datos de calificaciones (ratings.csv)**
  Todas las calificaciones están contenidas en el archivo ratings.csv. Cada línea de este archivo después de la fila del encabezado representa una calificación de una película por parte de un usuario y tiene el siguiente formato:
  - ID de usuario
  - ID de película
  - Clasificación
  - Marca de tiempo

  Las líneas dentro de este archivo están ordenadas primero por ID de usuario y luego, dentro de usuario, por ID de película.
  Las calificaciones se realizan en una escala de 5 estrellas, con incrementos de media estrella (0,5 estrellas - 5,0 estrellas).

- **Estructura del archivo de datos de etiquetas (tags.csv)**
  Todas las etiquetas están contenidas en el archivo tags.csv. Cada línea de este archivo después de la fila del encabezado representa una etiqueta aplicada a una película por un usuario y tiene el siguiente formato:
  - ID de usuario
  - ID de película
  - Etiqueta
  - Marca de tiempo

  Las líneas dentro de este archivo están ordenadas primero por ID de usuario y luego, dentro de usuario, por ID de película. Las etiquetas son metadatos generados por el usuario sobre películas. Cada etiqueta suele ser una sola palabra o frase corta. El significado, el valor y el propósito de una etiqueta en particular lo determina cada usuario.

La base de datos MovieLense tiene una muestra de sus datos integrados en la librería *RecommenderLab*. MovieLense es un objeto realRatingMatrix que contiene una muestra de 943 usuarios que han evaluado películas en 1664 calificaciones anónimas. Cada fila corresponde a un usuario, cada columna a una película y cada valor a una calificación. Estos datos fueron recopilados entre el 9 de enero de 1995 y el 20 de julio de 2023.

Metodología

Análisis exploratorio

Para el análisis exploratorio de la base de datos MovieLense hemos utilizado diversos tipos de gráficos y estadísticas descriptivas que proporcionan una visión clara y detallada de la distribución y las tendencias en los datos.

Entre los gráficos y estadísticas descriptivas están están los cuartiles, histogramas, mapas de calor, gráficos de barras, porcentajes y conteos. Los cuartiles los utilizamos para representar cómo están distribuidos las visualizaciones y las valoraciones. Por otra parte, los histogramas los hemos utilizado para visualizar la distribución de las valoraciones de las películas, los mapas de calor ilustran las valoraciones de películas por usuarios y esto ayuda a identificar patrones de puntuación y la intensidad de las valoraciones. Por último, añadir que los gráficos de barras se han utilizado para mostrar el número de visualizaciones de las películas mejor valoradas, el recuento de películas clasificadas por género y para conocer la distribución de usuarios por sexo y edad.

Sistemas de recomendación

Actualmente los **sistemas de recomendación** se definen como herramientas diseñadas para interactuar con conjuntos extensos y complejos de información, con el fin de proporcionar, de manera automatizada, información relevante para el usuario en base a su historial de comportamiento. Explicado con otras palabras, los sistemas de recomendación implementan algoritmos que analizan patrones en datos históricos de usuarios para predecir y sugerir elementos que pueden ser de interés a otro usuario.

A continuación, explicaremos la clasificación de los tipos de filtrado en los sistemas de recomendación se basan en tres enfoques principales:

- Filtrado Basado en Contenido: este enfoque "*utiliza la similitud entre elementos para recomendar elementos similares a lo que le gusta al usuario*" (Google for Developers, 2022). Este tipo de filtrado se centra en analizar las características de cada elemento y las compara con las preferencias del usuario para ofrecer recomendaciones atractivas para él.
  Por ejemplo, si a un usuario le gustan películas de ciencia ficción, este método recomendará otras películas del mismo género o con temas similares, basándose en características como el género, el director, etc.
- Filtrado Colaborativo: este método "*utiliza similitudes entre las consultas y los elementos de forma simultánea para proporcionar recomendaciones*" (Google for Developers, 2022). Se basa en la premisa de que se pueden predecir las preferencias de un usuario en base a las preferencias de usuarios con el mismo comportamiento.
  A modo de ejemplo, podemos decir que si el usuario A se comporta de manera similar al usuario B, y al usuario B le gustan las películas de ciencia ficción, el sistema le puede recomendar al usuario A películas de ciencia ficción.
- Filtrado Híbrido: Combina aspectos tanto del filtrado basado en contenido como del colaborativo. Este enfoque utiliza una combinación de análisis de contenido y de comportamiento del usuario para generar recomendaciones personalizadas que tienen en cuenta tanto las preferencias individuales como las similitudes con otros usuarios.

Para elaborar este trabajo hemos utilizado algunos trabajos recopilatorios de los sistemas de recomendación como los de Lu et al. (2015) y Sohail et al. (2017) y Portugal et al., (2018).

Filtrado Colaborativo

Los sistemas de recomendación basados en el Filtrado Colaborativo utilizan la noción de la sabiduría de la multitud (*wisdom of crowds*) para hacer sugerencias de contenido. Funcionan bajo la

premisa de que los usuarios con gustos similares en el pasado seguirán teniendo preferencias similares en el futuro.

Existen dos categorías principales de sistemas de Filtrado Colaborativo: aquellos que utilizan métodos basados en la vecindad, y los que utilizan métodos basados en modelos.

A. *Enfoque basado en vecindad.* Estos métodos se basan en las valoraciones de los usuarios para calcular similitudes entre usuarios o entre elementos. Estas similitudes se utilizan luego para generar recomendaciones. Un ejemplo es el algoritmo de *K-Nearest Neighbours* (K-NN) (*K Vecinos Más Próximos*, 2023).
    a. *Basados en usuario.* Se identifican usuarios con patrones de valoración similares y se recomiendan elementos que han sido bien valorados por los usuarios afines.
    b. *Basados en elementos.* Se agrupan los elementos por afinidad/proximidad, y se sugieren elementos similares a los que el usuario ha evaluado positivamente en el pasado.
B. *Enfoque basado en modelos.* Estos métodos emplean las valoraciones para aprender o estimar un modelo que se usa posteriormente para predecir valoraciones. Algunos ejemplos de estos modelos son los clasificadores bayesianos, las redes neuronales, los algoritmos genéticos, los sistemas borrosos y la descomposición matricial basada en la Descomposición de Valores Singulares (SVD, por sus siglas en inglés, *Singular Value Decomposition*) (*Descomposición En Valores Singulares*, 2024).

Para realizar un Filtrado Colaborativo se necesitan los siguientes componentes: una lista de usuarios, una lista de elementos, una matriz de valoraciones de usuarios por elementos, una métrica para medir la similitud entre usuarios-elementos, un método para seleccionar un subconjunto de vecinos y un método para predecir una valoración de los ítems que el usuario objetivo aún no ha valorado. Además, es importante remarcar que la matriz de utilidad, que contiene las valoraciones de los usuarios para distintos elementos, suele contener valores dispersos dado que todos los usuarios no valoran todos los elementos y por tanto existen datos vacíos.

En cuanto a la medición de la similitud, ésta puede llevarse a cabo utilizando diversas técnicas como la Similitud de Jaccard (Real & Vargas, 1996) o la Distancia del Coseno (*Similitud Coseno*, 2022). La Distancia del Coseno Ajustada se emplea para normalizar las valoraciones, lo que implica restar a cada valoración el promedio del usuario. Esto ayuda a transformar los valores bajos en negativos y los valores altos en positivos, lo que permite tener en cuenta las diferencias en las escalas de valoración utilizadas por los usuarios. Las predicciones de valoraciones se llevan a cabo seleccionando un grupo de vecinos (K-vecinos más cercanos) basados en la similitud calculada. Estas predicciones se realizan únicamente con los K-NN que han evaluado los elementos sobre los que se desea realizar la predicción.

Una **ventaja** principal del filtrado colaborativo es que son altamente eficaces y fáciles de usar e implementar. No obstante, presenta algún **inconveniente** como la escasez de datos. Cuando los datos son escasos, no es confiable puesto que los elementos comunes son pocos.

Filtrado Basado en Contenido

Los sistemas de recomendación basados en el contenido sugieren elementos a usuarios que guardan similitudes con los que han mostrado interés en dicho contenido en el pasado. Por ejemplo, se podrían recomendar películas con los mismos actores, directores o género, noticias con temas afines o personas con conexiones de amistad o intereses compartidos.

Para implementar este tipo de sistemas se requieren dos componentes esenciales: perfiles de elementos y perfiles de usuarios. Estos perfiles se utilizan para entender y analizar las preferencias y características de los contenido/elemento y usuarios, lo que permite ofrecer recomendaciones acertadas.

Un *perfil de elemento* se define como un conjunto de atributos descriptivos que caracterizan al contenido en cuestión. En sistemas basados en contenido, se acostumbra a trabajar con descripciones textuales que detallan las características más relevantes de los elementos. Estas características pueden presentarse en forma estructurada o no estructurada. Cuando las descripciones de los elementos son en formato de texto libre, se utilizan técnicas de extracción automática de palabras clave para construir el perfil. Una técnica ampliamente utilizada para este propósito es TF-IDF (Term Frequency – Inverse Document Frequency), que evalúa la importancia de un término tanto dentro de un documento como en un conjunto de documentos simultáneamente.

El propósito de un *perfil de usuario* es capturar una representación de los intereses del usuario mediante la recopilación de sus evaluaciones/valoraciones o comportamiento/compras anteriores. Una manera sencilla de crear un perfil de usuario es calcular un promedio ponderado de los perfiles de los contenidos que el usuario ha valorado, utilizado o comprado. No obstante, este método básico no considera las preferencias específicas del usuario en relación con los elementos, como sí hacen los procedimientos basados en filtrado colaborativo.

Para crear recomendaciones, se utiliza el comportamiento del usuario con los elementos y la similitud entre su perfil y el de los elementos. Es habitual utilizar la similitud del coseno como medida de la similaridad entre el vector del perfil del usuario y el del elemento. Después de evaluar las similitudes, un enfoque simple para ofrecer recomendaciones es el método de *K-Nearest Neighbours* (K-NN) (*K Vecinos Más Próximos*, 2023), el cual sugiere los k elementos con mayor similitud al perfil del usuario.

No obstante, la implementación de estos sistemas también presenta ciertas limitaciones y desafíos. Identificar las características idóneas para describir los elementos puede resultar complicado, sobre todo cuando se trata de elementos complejos como imágenes, películas o música. Asimismo, la creación de perfiles para usuarios nuevos puede ser una tarea difícil, y existe el riesgo de caer en la sobre-especialización, donde las recomendaciones se limitan demasiado y no permiten al usuario explorar nuevos intereses.

Filtrado Híbrido

Los sistemas de recomendación híbridos buscan fusionar las cualidades más destacadas de diversas técnicas de filtrado colaborativo, tales como aquellas basadas en usuarios, en elementos y en

modelos. Esto se realiza con el objetivo de superar las limitaciones propias de cada enfoque individual y mejorar la precisión de las recomendaciones.

Un sistema híbrido de recomendación busca optimizar la precisión de sus sugerencias mediante la combinación de diversos métodos. Para usuarios con gustos distintos, se puede emplear el filtrado colaborativo basado en usuarios, aprovechando la amplitud de sus preferencias. Por otro lado, cuando los gustos son más específicos, se recurre al filtrado colaborativo basado en contenido (elementos) para ofrecer recomendaciones que se ajusten con precisión a sus intereses particulares. Además, el filtrado basado en modelos se utiliza para predecir valoraciones en escenarios más complejos. Así, un sistema híbrido puede proporcionar recomendaciones más sólidas y precisas en una variedad de situaciones, compensando las limitaciones individuales de cada método.

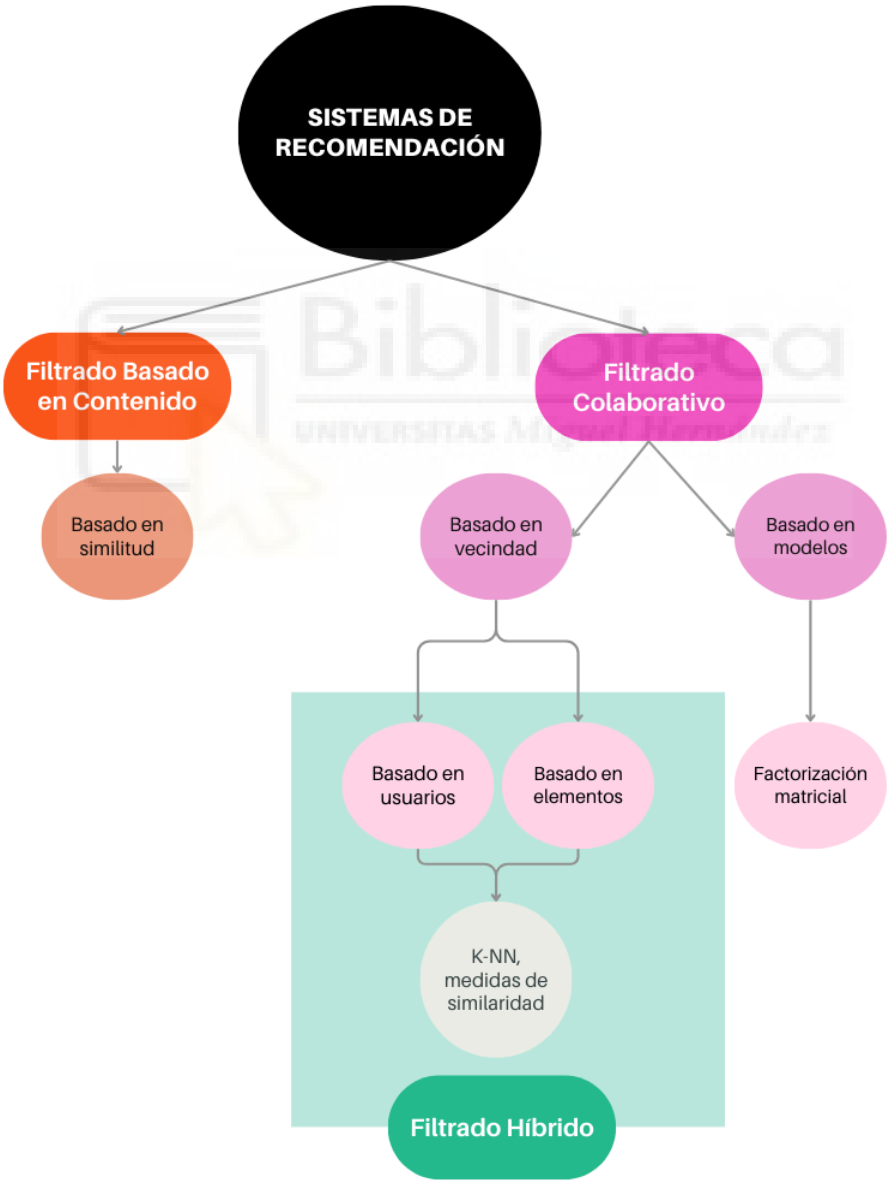

Figura 1. Jerarquía de los tipos de sistemas de recomendación.

Evaluación

La evaluación de los sistemas de recomendación es sumamente importante para garantizar la fiabilidad de éstos. Por lo general, para evaluar algoritmos de recomendación se divide la matriz de clasificaciones R. Primero se evalúa dividiendo los usuarios (filas) de R en dos conjuntos, $U_{train} \cup U_{test} = U$. Las filas de R que corresponden a los usuarios de entrenamiento $Utrain$se utilizan para aprender el modelo de recomendación. Por tanto, cada usuario ua pertc $u_a \cup U_{test}$ corresponde a un usuario activo. Antes de generar recomendaciones, se retienen algunos elementos del perfil r_{u_a} y se mide la correspondencia entre la calificación predicha y el valor retenido, o para los N algoritmos principales, si el usuario califica altamente los elementos en la lista recomendada. Finalmente se promedian las medidas de evaluación calculadas para los usuarios de la prueba.

Para determinar cómo dividir U en U_{train} y U_{test} podemos utilizar varios enfoques (Kohavi & Provost, 1998).

- *Splitting*. Se asigna aleatoriamente una proporción predefinida de los usuarios al conjunto de entrenamiento (U_{test}) y todos los demás al conjunto de prueba (U_{train}).
- *Bootstrap sampling*. Se toman muestras de U_{test} con reemplazo para crear el conjunto de entrenamiento (U_{train}) y después usar los usuarios que no están en el conjunto de entrenamiento como conjunto de prueba (U_{test}). La principal ventaja de este procedimiento es que para conjuntos de datos pequeños podemos crear conjuntos de entrenamiento más grandes y tener todavía usuarios para realizar pruebas.
- *k-fold cross-validation*. Se divide U en k conjuntos (llamados pliegues) del mismo tamaño aproximadamente. Después se evalúa k veces, siempre usando un pliegue para testear y todos los demás pliegues para inclinarnos. Los k resultados se pueden promediar. Este enfoque garantiza que cada usuario está al menos una vez en el conjunto de prueba y el promedio produce resultados y estimaciones de error más sólidos.

Evaluación de calificaciones previstas

Una forma típica de evaluar una predicción es calcular la desviación de la predicción del valor real. Esta es la base del error medio medio (MAE) (*Error Absoluto Medio*, 2022):

$$Error\ Absoluto\ Medio\ (MAE) = \frac{1}{|k|} \sum_{(j,l)\in k} \left| r_{jl} - \hat{r}_{jl} \right|$$

donde k es el conjunto de todos los pares usuario-elemento (j, l) para los cuales tenemos una calificación predicha $\hat{r}_{jl}$ y una calificación conocida r_{jl} que no se ha usado para aprender el modelo de recomendación.

Otra medida popular es el la raíz del error cuadrático medio (RMSE) (*Raíz del Error Cuadrático Medio*, 2020):

$$Raíz\ del\ Error\ Cuadrático\ Medio\ (RMSE) = \sqrt{\frac{\sum_{(J,L)\in K}(r_{jl}-\hat{r}_{jl})^2}{|K|}}$$

RMSE penaliza con mayor fuerza los errores más grandes que MAE y, por lo tanto, es adecuado para situaciones en las que los pequeños errores de predicción no son muy importantes.

Evaluación de las N recomendaciones principales

Los elementos en las listas de los N principales previstos y los elementos retenidos que le gustan al usuario para todos los usuarios de prueba de U_{test} se pueden agregar en la llamada matriz de confusión que se muestra en la Tabla 1 (Kohavi & Provost, 1998) que corresponde exactamente a los resultados de un experimento estadístico clásico. La matriz de confusión muestra cuántos de los elementos recomendados en las N listas principales (columna predicha positiva; d + b) fueron elementos retenidos y, por lo tanto, recomendaciones correctas (celda d) y cuántos fueron potencialmente incorrectos (celda b). La matriz también muestra cuántos de los elementos no recomendados (columna prevista negativa; a + c) deberían haberse recomendado realmente, ya que representan elementos retenidos (celda c).

Tabla 1. Matriz de confusión.

actual / predicción	Positivo	Negativo
Positivo	a	b
Negativo	c	d

De la matriz de confusión se pueden derivar varias medidas de desempeño. Para la tarea de minería de datos de un sistema de recomendación, el rendimiento de un algoritmo depende de su capacidad para aprender patrones significativos en el conjunto de datos. Las medidas de rendimiento utilizadas para evaluar estos algoritmos tienen su origen en el aprendizaje automático. Una medida comúnmente utilizada es la precisión (*accuracy*), la fracción de recomendaciones correctas hasta el total de recomendaciones posibles.

$$Accuracy = \frac{correct\ recommendations}{total\ possible\ recommendations} = \frac{a+d}{a+b+c+d}$$

Una medida de error común es el error absoluto medio (MAE, también llamado desviación absoluta media o MAD).

$$Desviación\ Absoluta\ Media\ (MAE) = \frac{1}{N}\sum_{i=1}^{N}\left|e_i\right| = \frac{b+c}{a+b+c+d}$$

donde $N = a + b + c + d$ es el número total de ítems que se pueden recomendar y e_i es el error absoluto de cada ítem. Dado que tratamos con datos 0-1, e_i solo puede ser cero (en las celdas a y d en la matriz de confusión) o uno (en las celdas b y c). Para los algoritmos de recomendación

de evaluación de datos de calificación, a menudo se utiliza la raíz del error cuadrático medio. Para datos 0-1, se reduce a la raíz cuadrada de MAE.
Los sistemas de recomendación ayudan a encontrar elementos de interés entre el conjunto de todos los elementos disponibles. Esto puede verse como una tarea de recuperación conocida por la recuperación de información. Por lo tanto, las medidas estándar de rendimiento de recuperación de información se utilizan con frecuencia para evaluar el rendimiento del recomendador. La precisión (*precision*) y el recuerdo (*recall*) son las medidas más conocidas utilizadas en la recuperación de información (Salton & McGill, 1983).

$$Precision = \frac{correctly\ recommended\ items}{total\ recommended\ items} = \frac{d}{b+d}$$

$$Recall = \frac{correctly\ recommended\ items}{total\ useful\ recommendations} = \frac{d}{c+d}$$

En ocasiones, se desconoce el número total de recomendaciones útiles necesarias para retirarlas, ya que sería necesario inspeccionar toda la colección. Sin embargo, en lugar del total de recomendaciones útiles reales, a menudo se utiliza el número total de recomendaciones útiles conocidas. La precisión y la recuperación son propiedades conflictivas; alta precisión significa baja recuperación y viceversa. Para encontrar un equilibrio óptimo entre precisión y recuperación se puede utilizar una medida de un solo valor como la medida E (*E-measure*) (van Rijsbergen C, 1979). El parámetro α controla el equilibrio entre precisión y recordar.

$$E - measure = \frac{1}{\alpha(1/Precision) + (1-\alpha)(1/Recall)}$$

Una medida popular de un solo valor es la medida F (*F-measure*). Se define como la media armónica de precisión y recuperación.

$$F - measure = \frac{2\ Precision\ Recall}{Precision + Recall} = \frac{2}{1/Precision + 1/Recall}$$

Otro método utilizado para comparar dos clasificadores con diferentes configuraciones de parámetros es la característica operativa del receptor (ROC). El método fue desarrollado para la detección de señales y se remonta al modelo de Swets (van Rijsbergen C, 1979). La curva ROC es una gráfica de la probabilidad de detección del sistema (también llamada sensibilidad o tasa de verdaderos positivos TPR) por la probabilidad de falsa alarma Una forma posible de comparar la eficiencia de dos sistemas es comparando el tamaño del área bajo la curva ROC, donde un área más grande indica un mejor rendimiento.

Sistemas de recomendación en R

R (Ihaka, 2024) es un lenguaje de programación orientado al análisis estadístico, surgido como una versión de código abierto del lenguaje de programación S. R nace en 1993 como un proyecto de investigación en los Laboratorios Bell por parte de Robert Gentleman y Ross Ihaka. En el año 1995 se consolidó como un proyecto de código abierto bajo la Licencia Pública General de GNU (sistema operativo de software libre).

Hoy en día, se utiliza como herramienta estadística en múltiples disciplinas, para el análisis estadístico, aprendizaje automático, minería de datos, econometría, investigación biomédica, bioinformática, modelos predictivos y la inferencia estadística. Su flexibilidad deriva de su carácter de lenguaje de código abierto, con una gran cantidad de librerías con funciones implementadas por sus usuarios y puestas a disposición de la comunidad.

El proceso general para desarrollar librerías y distribuirlas en R comienza con el desarrollo de funciones y algoritmos en R. El siguiente paso es generar una documentación clara y completa para que otros usuarios puedan comprender cómo se utilizan las funciones desarrolladas y cuál es el propósito de la librería. Después de los dos primeros pasos, se crea un paquete en R que incluye las funciones y la documentación, junto con todos los archivos necesarios (metadatos y ejemplos). A continuación, se realizan múltiples pruebas para garantizar que el paquete funcione correctamente y devuelva los resultados esperados en todos los escenarios planteados. Para finalizar el proceso, la librería se distribuye a través del repositorio de paquetes de R (CRAN), GitHub u otros repositorios conocidos entre la comunidad de R para que los usuarios interesados puedan instalarlo y utilizarlo fácilmente.

Puesto que los algoritmos que se utilizan en los sistemas de recomendación están basados en técnicas estadísticas multivariantes, es viable implementar en R sistemas de recomendación bajo los diferentes enfoques, básicamente utilizando las librerías específicas de análisis multivariantes como son *FactoMineR* (Husson, 2008), *caret* (Kuhn, 2008), *MASS* (Ripley, 2002), *factoextra* (Kassambara, 2020).

Se han desarrollado en R librerías específicas sobre la implementación de sistemas de recomendación, como son:

- *RecommenderLab (2022)* (Hahsler, 2022). Ofrece una amplia variedad de algoritmos de recomendación y proporciona un marco de trabajo para el desarrollo y evaluación de sistemas de recomendación, centrándose especialmente en el **Filtrado Colaborativo**. Fue desarrollada por Michael Hahsler junto con otros colaboradores.
  La principal característica de esta librería es facilitar a los usuarios una herramienta robusta y flexible para implementar sistemas de recomendación en R. Está diseñada para trabajar con conjuntos de datos que contienen información sobre la interacción entre usuarios y elementos, tales como películas, series, canciones o artículos.
  Su ejecución se lleva a cabo mediante el análisis de los patrones de comportamiento de los usuarios y la similitud entre elementos para generar recomendaciones personalizadas.
  En términos generales, la librería *RecommenderLab* funciona siguiendo los siguientes pasos:
  - Preparación y limpieza de datos, lo que implica la carga y la limpieza de los datos con información sobre usuarios, elementos y sus interacciones.
  - Selección del algoritmo. *RecommenderLab* ofrece varios algoritmos de recomendación, dependiendo de las características del conjunto de datos y los objetivos del proyecto.
  - Entrenamiento del modelo: Con los datos preparados, se entrena el modelo de recomendación utilizando el algoritmo seleccionado. Durante este proceso, el

modelo aprende a identificar patrones de comportamiento y similitudes entre usuarios y elementos.
- Creación de recomendaciones: Cuando el modelo ya está entrenado, se puede utilizar para generar recomendaciones personalizadas para usuarios específicos. Estas recomendaciones se basan en el análisis de patrones anteriormente identificados y se pueden ajustar según las preferencias individuales de cada usuario.

- *rectools (2022)* (Feldman & Tikhonovich, 2022). Ofrece una amplia gama de herramientas para implementar sistemas de recomendación avanzados. El paquete tiene implementado información de covariables de usuarios y artículos, además de preferencias de categoría de artículos.
  Rectools proporciona a los usuarios de R una solución integral para la implementación y evaluación de sistemas de recomendación. Desde modelos más utilizados, como ALS implícita, KNN implícita, LightFM, SVD y DSSM, hasta las más nuevas variaciones, como Diversidad, Novedad y Serendipia. La librería pretende ofrecer una amplia gama de opciones para adaptarse a las necesidades de recomendación que surjan.
  El paquete ofrece diversos modelos de recomendación, incluyendo el modelo ANOVA, Modelos de Factorización Matricial (NMF), Modelos de Coseno y más. Los usuarios pueden aprovechar la metodología de validación cruzada para evaluar la precisión de los modelos y comparar su rendimiento.
  Además, el paquete permite la personalización de los modelos así como la asignación de diferentes pesos a los componentes α y β en el Método de los Momentos (MM).

- *recosystem* (Qiu, 2023). Diseñada para la implementación de sistemas de recomendación, especialmente centrada en el **Filtrado Colaborativo**. Fue desarrollada por Liang Sun y Jiahui Liu, dos investigadores de la Universidad de Columbia. Su principal finalidad es proporcionar a los usuarios de R una herramienta eficiente y fácil de usar para generar recomendaciones basadas en comportamiento y las preferencias de los usuarios. Está orientada a resolver problemas de recomendación en música, películas, productos o servicios. El funcionamiento básico de *recosystem* es muy parecido al de la librería *RecommenderLab*:
  - Cálculo de similitudes y diferencias. Después de entrenar el algoritmo y haber modelado las interacciones, se calcula la similitud entre usuarios o entre elementos. La similitud puede calcularse utilizando diversas técnicas, como la similitud coseno o la correlación de Pearson.

También hay librerías que implementan sistemas de recomendación en otros lenguajes como Python. Estas librerías son:

- *rrecsys (2017)* (Çoba & Zanker, 2017). Tiene la característica de procesar conjuntos de datos de recomendación como entrada y genera predicciones de calificación y elementos recomendados.
  El funcionamiento del paquete *rrecsys* permite a los usuarios recomendar y predecir elementos utilizando algoritmos de recomendación con métodos no personalizados como el

promedio global, promedio del artículo, promedio de usuarios y el más popular. Además, también utiliza métodos de **Filtrado Colaborativo** como K-vecinos más cercanos (K-NN), SVD de Simon Funk, clasificación personalizada bayesiana (BPR) y mínimos cuadrados alternos ponderados (wALS). Igualmente, *rrecsys* proporciona una metodología de evaluación con métricas estándar para evaluar el rendimiento de los modelos de recomendación.

- *LKPY (2018)* (Ekstrand, 2018). Librería diseñada para llevar a cabo experimentos de sistemas de recomendación sin conexión a internet. Su diseño modular se basa en el ecosistema PyData, lo que facilita la implementación y prueba de diversos algoritmos y estrategias de evaluación.

- *Surprise (2020)* (Hug, 2020). Librería diseñada para construir y evaluar algoritmos de predicción de calificaciones en sistemas de recomendación. Su principal objetivo es predecir datos faltantes en un conjunto de datos de interacciones usuario - elemento.

RecommenderLab

En este proyecto nos centramos en la librería *RecommenderLab* de R para construir un sistema de recomendación eficiente y resolutivo. La elección de esta librería se basa en su versatilidad y la opción que te permite poder crear sistemas de recomendación desde cero.

En cuanto a la versatilidad, este paquete es compatible con diferentes tipos de datos, incluidos conjuntos de datos de clasificación (por ejemplo, calificaciones de 1 a 5 estrellas) y conjuntos de datos binarios (0 o 1) que representan interacciones. Además, su popularidad se refleja en la creación de varios paquetes adicionales en R, como *cmfrec* (Cortes, 2023), *crassmat* (Kunz, 2019), *recometrics* (Cortes, 2023), *recomenderlabBX* (Hahsler, 2022), *recomenderlabJester* (Hahsler, 2022) y *RMOA* (Wijffels, 2022), que extienden y aprovechan las funcionalidades de RecommenderLab para diversas aplicaciones de recomendación.

La librería ofrece distintos algoritmos, además de técnicas de evaluación, que vamos a describir a continuación.

Los algoritmos que se incluyen integrados en el paquete *RecommenderLab* son los siguientes:

- *User-based Collaborative Filtering* (UBCF): Este algoritmo identifica usuarios similares en función de sus interacciones pasadas y recomienda elementos que los usuarios similares han apreciado pero el usuario objetivo no ha visualizado todavía. Las recomendaciones están basadas en el método del aprendizaje automático K-Nearest Neighbors (K-NN).
  Se recomienda el uso de este algoritmo cuando se dispone de datos detallados de interacciones de usuarios con elementos. Es útil para sistemas donde la similitud entre usuarios es relevante y las preferencias de usuario son estables a lo largo del tiempo. Además, funciona bien en conjuntos de datos de tamaño moderado.
  En cuanto a las ventajas de este algoritmo:
  - Simple de implementar
  - No requiere información detallada sobre los elementos

Respecto a los inconvenientes del algoritmo:

- Problema de dispersión de datos para usuarios activos.
- Sensible a cambios en las preferencias de los usuarios.
- Problema de escalabilidad con grandes conjuntos de datos de usuarios y elementos.

- *Item-based Collaborative Filtering* (IBCF): Este algoritmo es similar al UBCF, pero en lugar de comparar usuarios, compara elementos. El algoritmo determina la similitud entre elementos y genera recomendaciones según los elementos más similares a los preferidos por el usuario.
  El uso de este algoritmo es recomendado cuando hay una variedad de elementos y se dispone de menor número de datos de usuario. Es útil para sistemas donde la similitud etre elementos es más estable que entre usuarios.
  En cuanto a las ventajas de este algoritmo:
  - Menos propenso a la dispersión de datos en comparación con UBCF.
  - Menos susceptible a cambios en las preferencias de los usuarios.
  - Puede manejar mejor grandes conjuntos de datos de usuarios y elementos.

  Respecto a los inconvenientes del algoritmo:
  - Puede requerir más recursos computacionales para calcular las similitudes entre elementos.
  - No es muy efectivo para usuarios nuevos o con pocas interacciones.

- *SVD with column-mean imputation* (SVD): Este algoritmo utiliza la descomposición de valores singulares (SVD) para reducir la dimensionalidad de la matriz de calificaciones de usuario-elemento. Además, se utiliza la media de las calificaciones de cada elemento para realizar predicciones basadas en la matriz de calificaciones de usuario-elemento.
  Se recomienda el uso de este algoritmo para conjuntos de datos grandes y dispersos. Es útil cuando se desean obtener recomendaciones precisas y la interpretabilidad no es una prioridad. Además, es adecuado cuando se plantea que pueda existir complejidad computacional.
  En cuanto a las ventajas de este algoritmo:
  - Puede manejar matrices y grandes conjuntos de datos.
  - Ofrece una buena precisión en la recomendación.

  Respecto a los inconvenientes del algoritmo:
  - Sensible a valores atípicos y ruido en los datos.
  - Puede no ser eficiente con conjuntos de datos muy grandes.

- *Funk SVD* (SVDF): Este algoritmo es una variante del SVD que utiliza la factorización de matrices para predecir las calificaciones de los elementos para los usuarios. Se basa en minimizar la diferencia entre las calificaciones reales y las predicciones mediante la descomposición de matrices.
  Este algoritmo está recomendado para el uso de grandes conjuntos de datos y dispersos donde también se busca la precisión. Es útil cuando se necesita una mejora sobre los métodos de SVD tradicionales.
  En cuanto a las ventajas de este algoritmo:

- Puede manejar conjuntos de datos grandes y dispersos.
- Tiende a proporcionar mejores resultados que el SVD tradicional.

Respecto a los inconvenientes del algoritmo:

- Sensible a valores atípicos y ruido en los datos.
- Puede ser computacionalmente costoso.

- *Alternating Least Squares* (ALS): Este algoritmo aborda el problema de factorización de matrices utilizando la técnica de optimización llamada Alternating Least Squares. Se alterna entre la optimización de filas y columnas de la matriz de calificaciones de usuario-elemento para encontrar las matrices de factores que mejor aproximan las calificaciones originales.
  Se recomienda su uso para grandes conjuntos de datos donde la escalabilidad es una preocupación. Es útil cuando se necesita manejar matrices dispersas y se desea una precisión razonable. Adecuando cuando se busca una solución eficiente y escalable.
  En cuanto a las ventajas de este algoritmo:
  - Eficiente para grandes conjuntos de datos.
  - Puede manejar matrices dispersas.

  Respecto a los inconvenientes del algoritmo:
  - Menos interpretable que otros métodos.

- *Matrix factorization with LIBMF* (LIBMF): Utiliza la librería LIBMF para realizar la factorización de matrices. Este algoritmo encuentra dos matrices de factores, para usuarios y para elementos, que minimizan la diferencia entre las calificaciones reales y las predicciones.
  Se recomienda su uso para grandes conjuntos de datos donde la escalabilidad es una preocupación. Es adecuado cuando se puede comprometer la interpretabilidad a cambio de un mejor rendimiento.
  En cuanto a las ventajas de este algoritmo:
  - Eficiente y escalable para grandes conjuntos de datos.
  - Puede manejar matrices dispersas.

  Respecto a los inconvenientes del algoritmo:
  - Menos interpretabilidad.
  - Requiere ajuste de hiperparámetros.

- *Association rule-based recommender* (AR): Este algoritmo utiliza reglas de asociación para encontrar relaciones entre los diferentes elementos en función de las interacciones de los usuarios. Más tarde, se utilizan estas reglas para hacer recomendaciones de elementos asociados.
  Se recomienda su uso cuando se desea descubrir patrones de asociación entre elementos. Es útil cuando se necesita una recomendación basada en la co-ocurrencia de elementos. También cabe añadir que es adecuado para sistemas donde no se tienen preferencias de usuario explícitas.
  En cuanto a las ventajas de este algoritmo:
  - Puede capturar patrones de asociación complejos entre elementos.
  - No requiere información detallada sobre los usuarios.

Respecto a los inconvenientes del algoritmo:

- Sensible a elementos populares y sesgos en los datos.
- Puede generar recomendaciones poco personalizadas.

- *Popular items* (POPULAR): Este algoritmo simplemente recomienda los elementos más populares o los elementos más vistos por los usuarios. No tiene en cuenta las preferencias individuales de los usuarios.
  Se recomienda su uso como base para los sistemas de recomendación. Es bastante útil cuando se necesita una solución simple y rápida. Además, es adecuado para sistemas donde las preferencias de usuario no son tan distintivas.
  En cuanto a las ventajas de este algoritmo:
  - Simple y rápido de implementar.
  - Útil como línea de base para comparaciones.

  Respecto a los inconvenientes del algoritmo:
  - No considera las preferencias individuales de los usuarios.
  - Genera recomendaciones poco personalizadas.

- *Randomly chosen items for comparison* (RANDOM): Este algoritmo selecciona elementos aleatoriamente para generar recomendaciones sin considerar las preferencias de los usuarios. Se utiliza principalmente como base para comparar el rendimiento de otros algoritmos.
  Se recomienda su uso como base para los sistemas de recomendación. Es útil para establecer un punto de referencia inicial para el rendimiento de otros algoritmos. Además, es adecuado cuando no se tienen expectativas específicas sobre el rendimiento del sistema.
  En cuanto a las ventajas de este algoritmo:
  - Simple y rápido de implementar.
  - Útil como línea de base para comparaciones.

  Respecto a los inconvenientes del algoritmo:
  - No proporciona recomendaciones útiles.
  - No considera las preferencias de los usuarios.

- *Re-recommend liked items* (RE-RECOMMEND): Este algoritmo recomienda elementos que el usuario ya ha valorado en el pasado. Es útil para mantener a los usuarios comprometidos mostrándoles elementos que anteriormente han sido de interés para ellos.
  Se recomienda su uso para mantener a los usuarios comprometidos enseñándole elementos que ya han consumido. Útil cuando se busca mejorar la satisfacción del usuario a corto plazo. Adecuado para sistemas donde se quiere reforzar la retroalimentación positiva de los usuarios.
  En cuanto a las ventajas de este algoritmo:
  - Mantiene a los usuarios comprometidos mostrando elementos que les gustaron anteriormente.
  - Puede mejorar la satisfacción del usuario.

  Respecto a los inconvenientes del algoritmo:
  - No proporciona diversidad en las recomendaciones.
  - Puede perder relevancia si las preferencias del usuario cambian con el tiempo.

- *Hybrid recommendations* (HybridRecommender): Este algoritmo combina múltiples métodos de recomendación anteriores, como UBCF, IBCF, SVD, etc., para mejorar la precisión y la variedad de las recomendaciones. Puede integrar diferentes tipos de enfoques para aprovechar sus fortalezas Se recomienda su uso cuando se necesita mejorar la precisión y la diversidad de las recomendaciones. Es realmente útil para combinar las fortalezas de diferentes algoritmos de recomendación. Además, es adecuado cuando se dispone de múltiples fuentes de datos y se busca una solución más completa y personalizada. individuales y mitigar sus debilidades.
  En cuanto a las ventajas de este algoritmo:
  - Combina las fortalezas de múltiples métodos de recomendación.
  - Puede mejorar la precisión y la diversidad de las recomendaciones.

  Respecto a los inconvenientes del algoritmo:
  - Requiere más esfuerzo de desarrollo e integración.
  - Puede ser complicado de optimizar.

En la Tabla 2 se presenta un resumen de la descripción de los distintos algoritmos integrados en RecommenderLab, su recomendación de uso y sus ventajas e inconvenientes:

Biblioteca
UNIVERSITAS Miguel Hernández

Tabla 2. Descripción y comparativa de los distintos algoritmos integrados en la librería RecommenderLab.

Algoritmo	**Descripción**	**Recomendación**	**Ventajas**	**Inconvenientes**
User-based Collaborative Filtering (UBCF)	Identifica usuarios similares en función de sus interacciones pasadas y recomienda elementos que los usuarios similares han apreciado pero el usuario objetivo no ha visualizado todavía. Basado en K-NN.	Con datos detallados de interacciones de usuarios con elementos. Útil cuando la similitud entre usuarios es relevante y las preferencias de usuario son estables a lo largo del tiempo. Funciona bien con datos de tamaño moderado	Simple de implementar. No requiere información detallada sobre los elementos.	Problema de dispersión de datos para usuarios activos. Sensible a cambios en las preferencias de los usuarios. Problema de escalabilidad con grandes conjuntos de datos de usuarios y elementos.
Item-based Collaborative Filtering (IBCF)	Similar al UBCF, pero comparando elementos. Determina la similitud entre elementos. Genera recomendaciones según los elementos más similares a los preferidos por el usuario.	Recomendado cuando hay variedad de elementos y tenemos pocos datos de usuarios. Útil para sistemas donde la similitud entre elementos es más estable que entre usuarios.	Menor dispersión de datos en comparación con UBCF. Menos susceptible a cambios en las preferencias de los usuarios. Maneja bien grandes conjuntos de datos.	Puede requerir más recursos computacionales para calcular las similitudes entre elementos. No es muy efectivo para usuarios nuevos o con pocas interacciones.
SVD with column-mean imputation (SVD)	Utiliza SVD para reducir la dimensionalidad de la matriz de calificaciones de usuario-elemento.	Recomendado para conjuntos de datos grandes y dispersos. Útil para obtener recomendaciones precisas.	Puede manejar matrices y grandes conjuntos de datos. Ofrece una buena precisión en la recomendación.	Sensible a valores atípicos y ruido en los datos. Puede no ser eficiente con conjuntos de datos grandes.

	Utiliza la media de las calificaciones de cada elemento para realizar predicciones basadas en la matriz de calificaciones de usuario-elemento.	La interpretabilidad no es una prioridad. Recomendable cuando pueda existir complejidad computacional.		
Funk SVD (SVDF)	Variante del SVD que utiliza la factorización de matrices para predecir las calificaciones de los elementos para los usuarios. Minimiza la diferencia entre las calificaciones reales y las predicciones mediante la descomposición de matrices.	Recomendado para el uso de grandes conjuntos de datos y dispersos. Se busca la precisión. Útil cuando se necesita una mejora sobre los métodos de SVD tradicionales.	Puede manejar conjuntos de datos grandes y dispersos. Tiende a proporcionar mejores resultados que el SVD tradicional.	Sensible a valores atípicos y ruido en los datos. Computacionalmente costoso.
Alternating Least Squares (ALS)	Aborda el problema de factorización de matrices utilizando la técnica de optimización Alternating Least Squares. Alterna entre la optimización de filas y columnas de la matriz de	Uso para grandes conjuntos de datos donde la escalabilidad es una preocupación. Útil cuando hay matrices dispersas y se desea una precisión razonable.	Eficiente para grandes conjuntos de datos. Puede manejar matrices dispersas.	Menos interpretable que los otros métodos.

	calificaciones de usuario - elemento para encontrar las matrices de factores que mejor aproximan las calificaciones originales.	Adecuado cuando se busca una solución eficiente y escalable.		
Matrix factorization with LIBMF (LIBMF)	Utiliza la librería LIBMF para realizar la factorización de matrices. Encuentra dos matrices de factores (usuarios y elementos) que minimizan la diferencia entre las calificaciones reales y las predicciones.	Uso recomendado para grandes conjuntos de datos donde la escalabilidad es una preocupación. Adecuado cuando se puede comprometer la interpretabilidad a cambio de un mejor rendimiento.	Eficiente y escalable para grandes conjuntos de datos. Puede manejar matrices dispersas.	Menos interpretable que los otros métodos. Requiere ajuste de hiperparámetros.
Association rule-based recommender (AR)	Utiliza reglas de asociación para encontrar relaciones entre los diferentes elementos en función de las interacciones de los usuarios.	Uso recomendado cuando se desea descubrir patrones de asociación entre elementos. Útil cuando se necesita una recomendación basada en la co-ocurrencia de elementos. Adecuado para sistemas donde no se tienen preferencias de usuario explícitas.	Puede capturar patrones de asociación complejos entre elementos. No requiere información detallada sobre los usuarios.	Sensible sesgos en los datos. Genera recomendaciones poco personalizadas.

Popular items (POPULAR)	Recomienda los elementos más populares o los elementos más vistos por los usuarios. No tiene en cuenta las preferencias individuales de los usuarios.	Uso recomendado como base para los sistemas de recomendación. Útil cuando se necesita una solución simple y rápida. Adecuado para sistemas donde las preferencias de usuario no son tan distintivas.	Simple y rápido de implementar. Útil como línea de base para comparaciones.	No considera las preferencias individuales de los usuarios. Genera recomendaciones poco personalizadas.
Randomly chosen items for comparison (RANDOM)	Selecciona elementos aleatoriamente para generar recomendaciones sin considerar las preferencias de los usuarios.	Uso recomendado como base para los sistemas de recomendación. Útil para establecer un punto de referencia inicial para el rendimiento de otros algoritmos. Adecuado cuando no se tienen expectativas específicas sobre el rendimiento del sistema.	Simple y rápido de implementar. Útil como línea de base para comparaciones.	No proporciona recomendaciones útiles. No considera las preferencias de los usuarios.
Re-recommend liked items (RERECOMMEND)	Recomienda elementos que el usuario ya ha valorado en el pasado. Útil para mantener a los usuarios comprometidos mostrándoles elementos	Uso recomendado para mantener a los usuarios comprometidos enseñándole elementos que ya han consumido.	Mantiene a los usuarios comprometidos mostrando elementos que les gustaron anteriormente. Puede mejorar la satisfacción del usuario.	No proporciona diversidad en las recomendaciones. Puede perder relevancia si las preferencias del usuario cambian con el tiempo.

	que anteriormente han sido de interés para ellos.	Útil cuando se busca mejorar la satisfacción del usuario a corto plazo. Adecuado para sistemas donde se quiere reforzar la retroalimentación positiva de los usuarios.		
Hybrid recommendations (HybridRecommender)	Combina métodos de recomendación anteriores, como UBCF, IBCF, SVD, etc., para mejorar la precisión y la variedad de las recomendaciones. Integra diferentes tipos de enfoques para aprovechar sus fortalezas individuales y mitigar sus debilidades.	Uso recomendado cuando se necesita mejorar la precisión y la diversidad de las recomendaciones. Útil para combinar las fortalezas de diferentes algoritmos de recomendación. Adecuado cuando se dispone de múltiples fuentes de datos y se busca una solución más completa y personalizada.	Combina las fortalezas de múltiples métodos de recomendación. Puede mejorar la precisión y la diversidad de las recomendaciones.	Requiere más esfuerzo de desarrollo e integración. Puede llegar a ser complicado de optimizar.

Resultados

Análisis Exploratorio

Gracias a los quantiles podemos obtener la información de que el 25% de las películas tienen menos de 7 visualizaciones, la mediana (50%) indica que el 50% de las películas tienen 27 visualizaciones o menos, otro 25% tiene más de 80 visualizaciones y el valor máximo (100%) indica que la película con más visualizaciones tiene 583 visualizaciones.

El 25% de los usuarios ha valorado menos de 32 películas, la mediana (50%) indica que el 50% de los usuarios ha valorado 64 películas o menos, otro 25% de los usuarios ha valorado 147.5 películas y el valor máximo (100%) indica que el usuario que más ha valorado, ha valorado 735 películas.

Este análisis nos proporciona una idea general sobre la distribución de las visualizaciones entre las películas de MovieLense y sobre la distribución de las valoraciones entre los usuarios.

Un **mapa de calor** es una representación gráfica de los datos en la que la intensidad de cada celda interpreta la magnitud de los valores, facilitando la identificación de patrones, tendencias y anomalías en los datos.

Cada celda tiene un color en una escala de negro-gris cuya intensidad refleja puntuaciones más altas. Los grises más claros pueden representar puntuaciones más bajas, mientras que los negros más oscuros representan puntuaciones más altas.

Los patrones de filas más oscuras que otras pueden significar que algunos usuarios dan puntuaciones más altas a todas las películas.

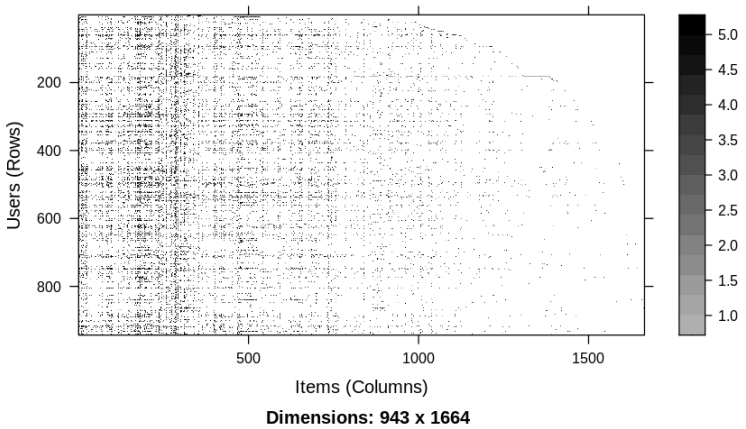

Figura 2. Mapa de calor con valoraciones de las películas por los usuarios en la base de datos MovieLense

Las filas de la matriz representan los usuarios, y las columnas representan las películas. El valor en la intersección de una fila y una columna es la puntuación que el usuario dio a esa película.

Las celdas de color negro oscuro representan puntuaciones altas, indicando que esas películas fueron altamente valoradas por esos usuarios.

Las celdas de color gris claro representan puntuaciones bajas o la ausencia de valoraciones, indicando que esas películas no fueron valoradas o fueron mal valoradas por esos usuarios.

Al observar la distribución de la escala de grises, se pueden identificar rápidamente patrones, como grupos de usuarios que valoran de manera similar o películas que tienden a recibir puntuaciones altas o bajas.

Explorando los valores de clasificación, podemos conocer que las puntuaciones registradas son números enteros del rango 0 a 5, donde 0 representa ausencia de valoración y del 1 al 5 representa la escala de valoración posible.

El porcentaje de películas no valoradas es del 93.67%. En la Figura 3 se muestra la distribución de valoraciones proporcionada por los usuarios para el 6,33% restante. En ella podemos ver que la puntuación más frecuente es la 4 para un 34.15% de las películas valoradas, seguida de la 3 y la 5 con unas frecuencias del 27.17% y 21.21% respectivamente.

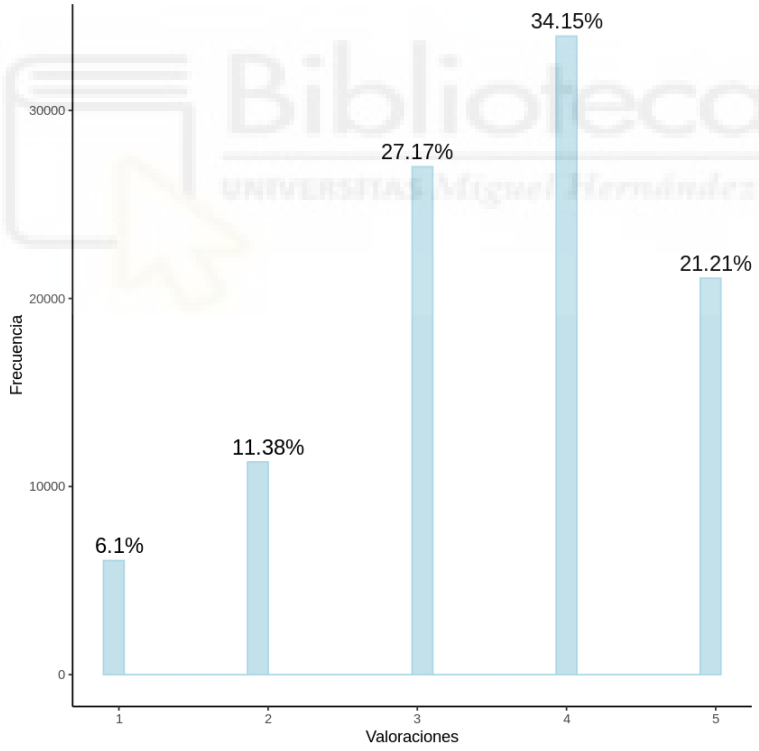

Figura 3. Distribución de las valoraciones proporcionadas en las películas que han sido valoradas por los usuarios.

En la Figura 4 mostramos el número de visualizaciones de las cinco películas mejor valoradas por los usuarios, ordenadas de arriba a abajo en el eje vertical.Observamos que Star Wars es, además de la mejor valorada, la más visualizada de todas con 583 visualizaciones, la siguiente mejor valorada es Contact con 509 visualizaciones y la tercera mejor valorada es Fargo con 508 visualizaciones.

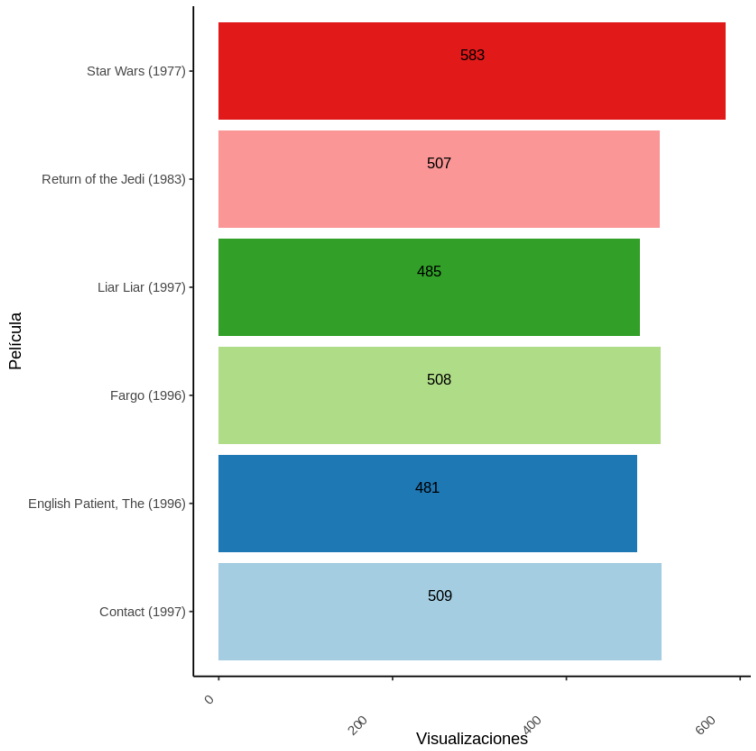

Figura 4. Número de visualizaciones de las películas mejor valoradas por los usuarios.

En la Figura 5 mostramos el recuento de películas clasificadas en cada uno de los géneros registrados en la base de datos. Apreciamos que los géneros drama y comedia son los más frecuentes con un total de 716 y 502 películas respectivamente, del total de 943 películas (algunas películas pertenecen a varios géneros simultáneamente).

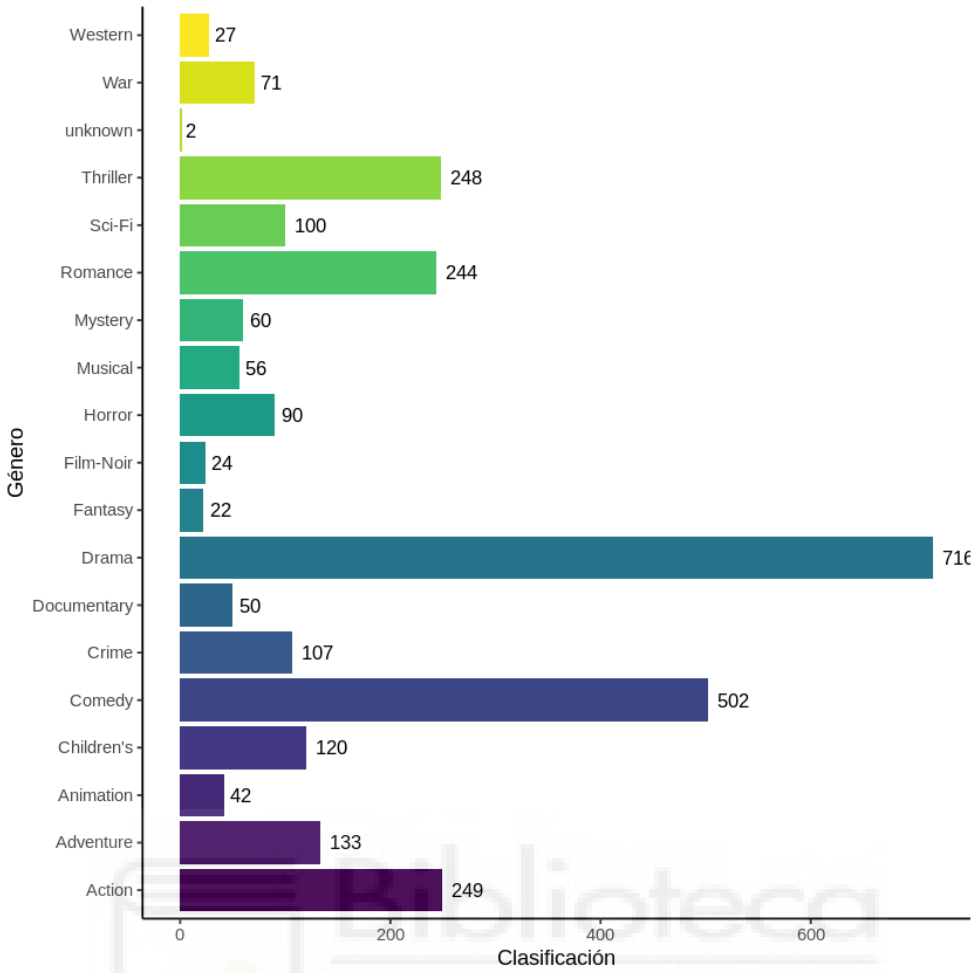

Figura 5. Recuento de películas clasificadas en cada género.

Respecto a la descripción de los usuarios en la base de datos, visualizamos en la Figura 6 que un 29% de ellos son mujeres y un 71% restante son hombres.

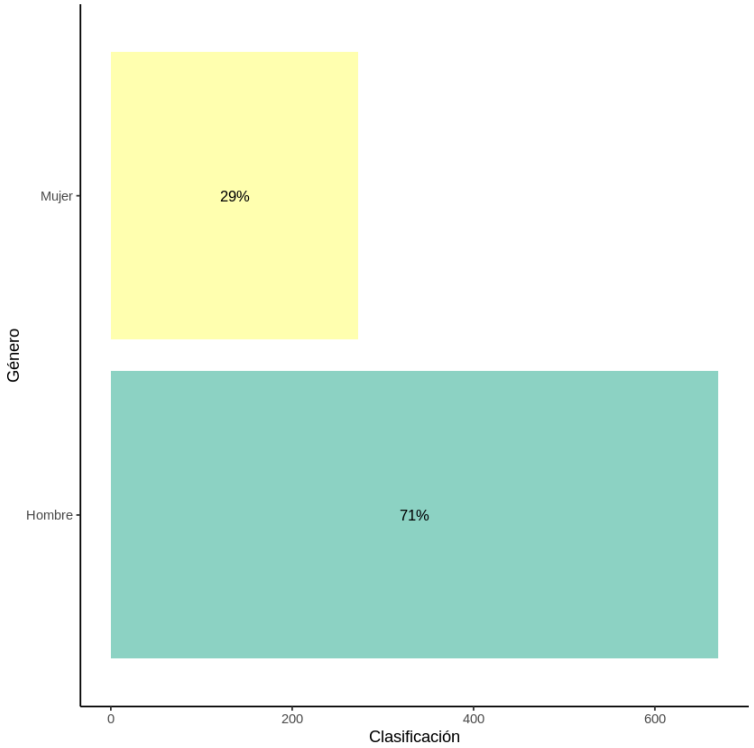

Figura 6. Distribución de los usuarios por sexo.

En la Figura 7 se muestra la distribución por edades de los usuarios y se aprecia que la mayoría de los hombres están entre los 20 y los 40 años, pero las mujeres se reparten de modo homogéneo entre los 20 y los 60 años.

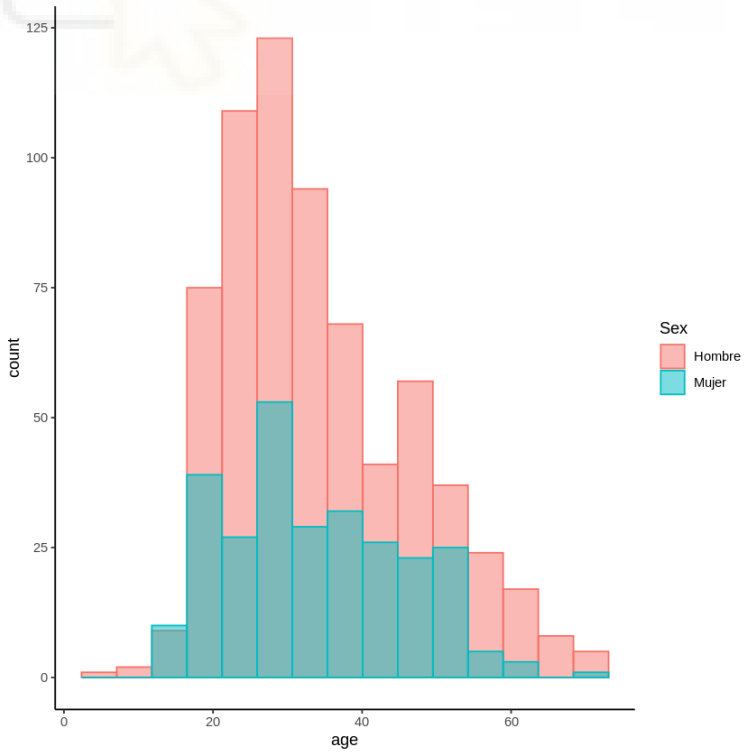

Figura 7. Distribución de las edades de los usuarios por sexo.

Aplicación de RecommenderLab

Una vez hemos realizado la carga de los datos con el comando data(MovieLense) disponible en la librería RecommenderLab, podemos visualizar las valoraciones con la sintaxis:

```
movie_ratings <- as(MovieLense, "realRatingMatrix")
ratings <- as(movie_ratings, "data.frame")
```

que nos devuelve un data.frame de 99392 filas y 3 columnas con las variables user, item y rating, que identifican respectivamente al usuario con un numeral, la película con su título y la valoración en una escala de enteros entre el 1 y el 5.

Para acceder a todos los algoritmos disponibles en el paquete RecommenderLab, junto con una breve explicación de su funcionamiento, podemos ejecutar el comando:

```
recommenderRegistry$get_entries(dataType = "realRatingMatrix")
```

A continuación, creamos un recomendador con el algoritmo que deseemos con el método Recommender(), indicando la muestra de datos que ha de utilizar para el ajuste y el algoritmo deseado con el argumento method.

A modo de ejemplo, generamos recomendaciones en función de la popularidad de las películas, con el algoritmo POPULAR y utilizando los primeros 941 usuarios en el conjunto de datos de MovieLense:

```
ML_POPULAR <- Recommender(MovieLense[1:941], method = "POPULAR")
```

El comando getModel(ML_POPULAR) proporciona los resultados del algoritmo de recomendación. Para obtener la recomendación para un nuevo usuario no utilizado en el ajuste del modelo, usamos el método predict() indicando las filas de los usuarios a los que queremos recomendar y el número de recomendaciones que queremos hacer.

```
recom <- predict(ML_POPULAR,MovieLense[941:943], n=5)
```

Para mostrar las recomendaciones en formato de lista utilizamos:

```
as(recom, "list")
```

que proporciona los siguientes resultados:

$`941`: 'Star Wars (1977)', 'Godfather, The (1972)', 'Fargo (1996)', 'Raiders of the Lost Ark (1981)', 'Silence of the Lambs, The (1991)'

$`942`: 'Godfather, The (1972)', 'Fargo (1996)', 'Silence of the Lambs, The (1991)', 'Shawshank Redemption, The (1994)', 'Return of the Jedi (1983)'
$`943`: 'Titanic (1997)', 'L.A. Confidential (1997)', 'Casablanca (1942)', 'One Flew Over the Cuckoo\'s Nest (1975)', 'Amadeus (1984)'

Una vez tenemos las recomendaciones descritas, podemos mostrar un número menor de ellas con el comando bestN():

```
recom3 <- bestN(recom, n=3)
as(recom3, "list")
```

Muchos algoritmos de recomendación también pueden predecir calificaciones, que se consiguen con el método predict() y el argumento type=”ratings”.

```
recom <- predict(ML_POPULAR, MovieLense[941:943], type = "ratings")
as(recom, "matrix")[,1:5]
```

Que proporciona una visualización como la que se muestra en la Tabla 3.

Tabla 3. Valoraciones predichas por el sistema de recomendación para tres usuarios sobre cinco películas.

A matrix: 3 × 5 of type dbl

	Toy Story (1995)	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)	Copycat (1995)
941	NA	3.778187	3.639979	4.016017	3.840134
942	4.559542	3.992473	3.854265	4.230303	4.054420
943	3.710516	NA	3.005239	3.381277	3.205394

Las calificaciones previstas se devuelven como un objeto de realRatingMatrix. La predicción contiene NA para los elementos calificados por los usuarios activos. En el ejemplo, mostramos las calificaciones previstas para los primeros 5 elementos para los 3 usuarios activos.

```
recom <- predict(ML_POPULAR, MovieLense[941:943], type = "ratingMatrix")
as(recom, "matrix")[,1:5]
```

En la Tabla 4 mostramos la matriz de calificación completa que incluye las calificaciones originales del usuario.

Tabla 4. Valoraciones predichas por el sistema de recomendación para tres usuarios sobre cinco películas.

A matrix: 3 × 5 of type dbl

	Toy Story (1995)	GoldenEye (1995)	Four Rooms (1995)	Get Shorty (1995)	Copycat (1995)
941	4.345256	3.778187	3.639979	4.016017	3.840134
942	4.559542	3.992473	3.854265	4.230303	4.054420
943	3.710516	3.143447	3.005239	3.381277	3.205394

Para evaluar un sistema de recomendación habitualmente realizamos un ajuste con una muestra de entrenamiento y lo testamos con una muestra test aplicando el método evaluationScheme(). En esta base de datos, vamos a utilizar los primeros 849 usuarios como muestra de entrenamiento (con el argumento train = 0.9 porque representan el 90% de los usuarios) y los restantes 94 como muestra test (10%). También podemos elegir el número de películas que destinamos al entrenamiento con el argumento given, o utilizarlas todas con la especificación given = -1. Asimismo, se puede especificar con el argumento goodRating la valoración mínima con la cual se va a hacer una recomendación. Probamos esta función con las especificaciones adicionales given=15 y goodRating=5.

```
e <- evaluationScheme(MovieLense, method = "split", train = 0.9, given=15,
goodRating=5)
```

Para comparar el filtrado colaborativo basado en usuarios (UBCF) y el basado en elementos (IBCF), utilizamos el siguiente código:

```
r1 <- Recommender(getData(e, "train"), "UBCF")
r2 <- Recommender(getData(e, "train"), "IBCF")

p1 <- predict(r1, getData(e, "known"), type="ratings")
p2 <- predict(r2, getData(e, "known"), type="ratings")
```

```
error <- rbind(
  UBCF = calcPredictionAccuracy(p1, getData(e, "unknown")),
  IBCF = calcPredictionAccuracy(p2, getData(e, "unknown"))
)
```

que proporciona la comparativa al modo de la Tabla 5:

Tabla 5. Comparativa de diferentes tipos de filtrados para un algoritmo.

A matrix: 2 × 3 of type dbl

	RMSE	MSE	MAE
UBCF	1.208667	1.460875	0.9561954
IBCF	1.302470	1.696429	1.0357143

En este ejemplo, el filtrado colaborativo basado en el usuario (UBCF) produce un error de predicción menor.

Para la evaluación del algoritmo de recomendación TopN creamos un esquema de validación cruzada:

```
scheme <- evaluationScheme(MovieLense, method="cross", k=4, given=3, goodRating=5)
```

A continuación utilizamos el esquema de evaluación creado para evaluar el método de recomendación popular. Evaluamos las listas de recomendaciones top 1, top 3, top 5, top 10, top 15 y top 20.

El resultado es un objeto de la clase EvaluationResult que contiene varias matrices de confusión. getConfusionMatrix() devolverá las matrices de confusión de las cuatro ejecuciones como una lista.

Analizamos el primer elemento de la lista que representa la primera de las cuatro ejecuciones:

```
results <- evaluate(scheme, method="POPULAR", type = "topNList", n=c(1,3,5,10,15,20))
getConfusionMatrix(results)[[1]]
```

En la Tabla 6 tenemos una matriz de confusión representada por seis filas, una para cada lista de N principales que se han utilizado para la evaluación, n es el número de recomendaciones por lista.

TP, FP, FN y TN son las entradas de verdaderos positivos, falsos positivos, falsos negativos y verdaderos negativos en la matriz de confusión. Las columnas restantes contienen rendimiento precalculado.

Tabla 6. Matriz de confusión.

A matrix: 6 × 10 of type dbl

TP	FP	FN	TN	N	precision	recall	TPR	FPR	n
0.2983193	0.4747899	19.67647	1640.697	1661.147	0.3858696	0.01708078	0.01708078	0.0002889721	1
0.6680672	1.6512605	19.30672	1639.521	1661.147	0.2880435	0.03595465	0.03595465	0.0010060732	3
1.0168067	2.8487395	18.95798	1638.324	1661.147	0.2630435	0.04985912	0.04985912	0.0017353552	5
1.8193277	5.9117647	18.15546	1635.261	1661.147	0.2353261	0.08974208	0.08974208	0.0036015679	10
2.5546218	9.0420168	17.42017	1632.130	1661.147	0.2202899	0.12650707	0.12650707	0.0055105026	15
3.1092437	12.3529412	16.86555	1628.819	1661.147	0.2010870	0.14892451	0.14892451	0.0075293173	20

En el Gráfico X se muestran los resultados de la evaluación mediante la curva ROC que traza la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR).

```
plot(results, annotate=TRUE)
```

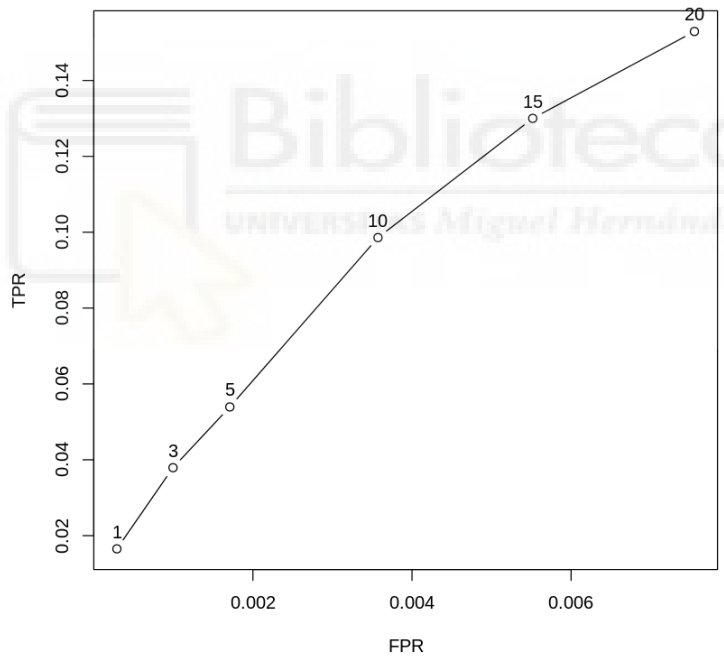

Figura 8. Curva ROC

Biblioteca
UNIVERSIDAD Miguel Hernández

Al graficar la curva de ROC, utilizamos "prec/rec" como segundo argumento para conseguir un gráfico de recuperación de precisión.

```
plot(results, "prec/rec", annotate=TRUE)
```

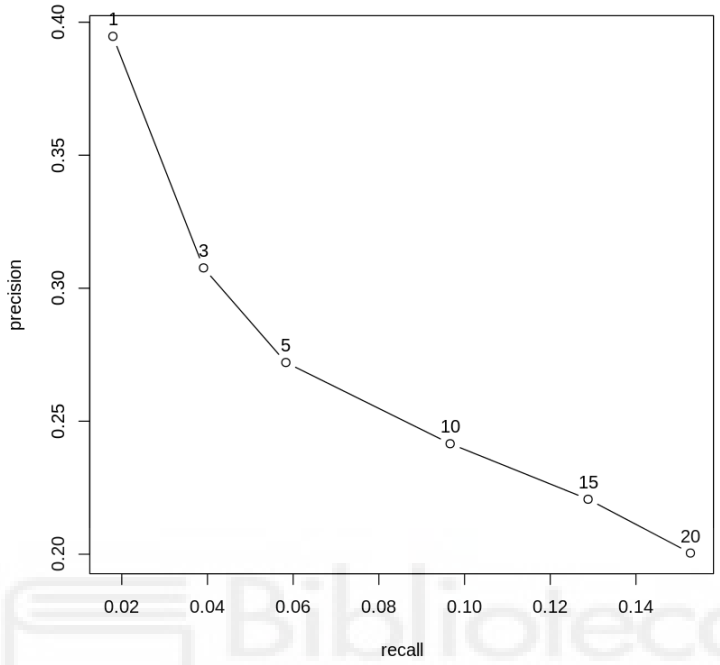

Figura 9. Gráfico de recuperación de precisión.

Una de las principales funciones de RecommenderLab es la comparación de varios algoritmos de recomendación. Para hacer efectiva dicha comparación se utiliza la función evaluate().

A continuación utilizamos el esquema de evaluación creado anteriormente para comparar cinco algoritmos de recomendación: RANDOM, POPULAR, UBCF, IBCF y SVD:

```
set.seed(2016)
scheme <- evaluationScheme(MovieLense, method = "split", train = 0.9, k=1,
given=5, goodRating = 5)
algorithms <- list(
  "random items" = list(name="RANDOM", param=NULL),
  "popular items" = list(name="POPULAR", param=NULL),
  "user-based CF" = list(name="UBCF", param=list(nn=50)),
  "item-based CF" = list(name="IBCF", param=list(k=50)),
  "SVD approximation" = list(name="SVD", param=list(k = 50))
)
results <- evaluate(scheme, algorithms, type="topNList",
n=c(1,3,5,10,15,20))
```

El resultado es un objeto de clase evaluationResultList para los cinco algoritmos recomendadores:

```
List of evaluation results for 5 recommenders:

$`random items`
Evaluation results for 1 folds/samples using method 'RANDOM'.

$`popular items`
Evaluation results for 1 folds/samples using method 'POPULAR'.

$`user-based CF`
Evaluation results for 1 folds/samples using method 'UBCF'.

$`item-based CF`
Evaluation results for 1 folds/samples using method 'IBCF'.

$`SVD approximation`
Evaluation results for 1 folds/samples using method 'SVD'.
```

Graficamos de nuevo la curva ROC en la Figura 10 y el gráfico de recuperación de precisión en la Figura 11. Destacamos que POPULAR ITEMS tiene una tasa de falsos positivos significativamente más elevada que el resto de algoritmos.

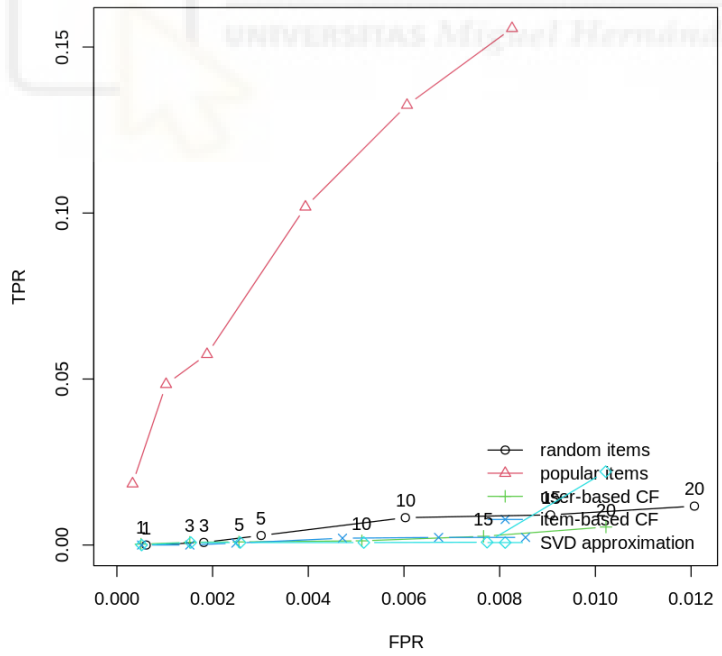

Figura 10. Curva ROC.

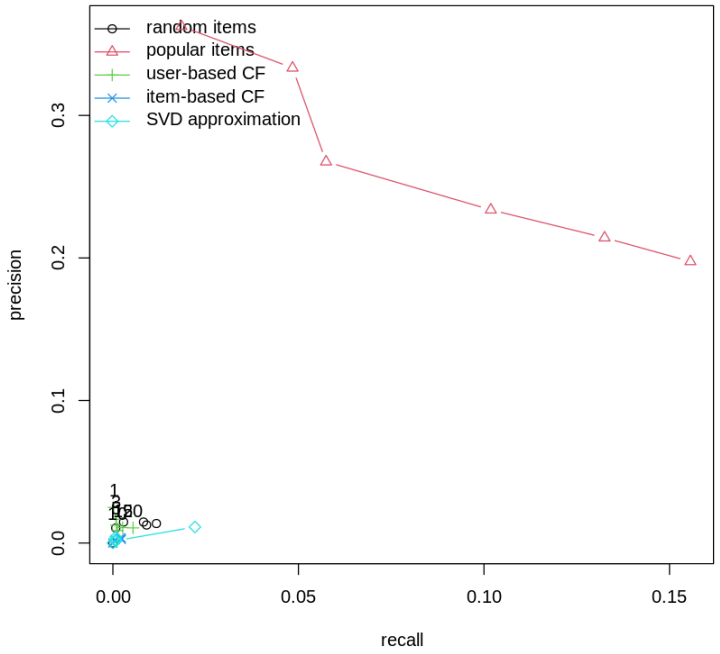

Figura 11. Gráfico de recuperación de precisión.

En la Figura 12 se muestra un diagrama de barras con la raíz del error cuadrático medio (RMSE), el error cuadrático medio (MSE) y el error medio cuadrático (MAE) de los cinco algoritmos.

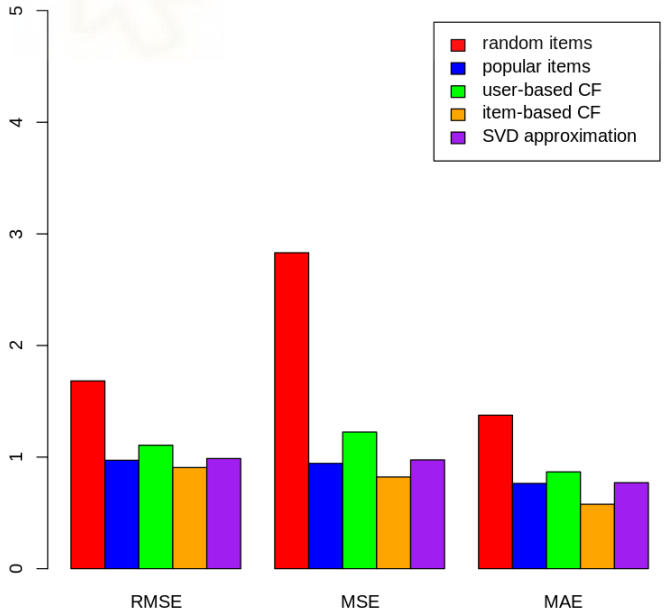

Figura 12. Diagrama de barras comparando errores.

El algoritmo RANDOM ITEMS es el que mayor número de error tiene en comparación con todos los demás algoritmos.

Finalmente, a modo de comparación, comprobaremos cómo se comparan los algoritmos con menos información. Para ello convertiremos el conjunto de datos en datos binarios (0-1)

```
MovieLense_binary <- binarize(MovieLense, minRating=5)
MovieLense_binary <- MovieLense_binary[rowCounts(MovieLense_binary)>20]
scheme_binary <- evaluationScheme(MovieLense_binary, method="split",
train=.9, k=1, given=3)
results_binary <- evaluate(scheme_binary, algorithms, type = "topNList",
n=c(1,3,5,10,15,20))
```

Debemos tener en cuenta que SVD no implementa un método para datos binarios y, por tanto, se omite.

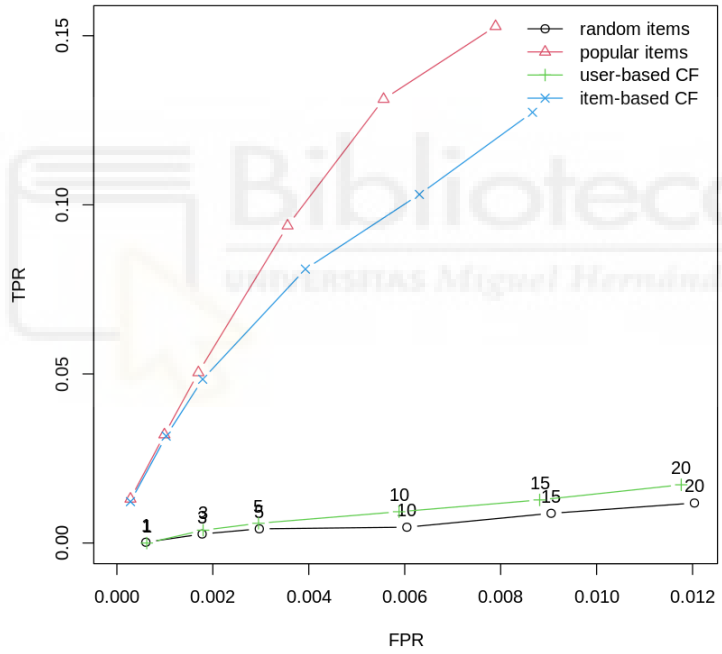

Figura 13. Curva ROC

En la Figura 13 podemos observar que al tener menos información, el rendimiento del filtrado colaborativo basado en elementos (IBCF) es el que más sufre y el recomendador RANDOM ITEM realiza recomendaciones igual de correctas como el filtrado colaborativo basado en usuarios (UBCF).
Obtenemos pues, que el mejor algoritmo de recomendación sobre esta base de datos es el filtrado colaborativo basado en usuarios (UBCF) porque tiene una tasa de falsos positivos prácticamente nula y, en comparación a los demás algoritmos, en la comparación de errores tiene unos valores aceptables.

Conclusión

Hemos cumplido con éxito los objetivos propuestos, haciendo un exhaustiva revisión sobre los fundamentos, tipologías y software disponible en la actualidad, tanto en el lenguaje de programación R como Python, para implementar sistemas de recomendación desde sus diferentes perspectivas: basados en contenido, en usuarios e híbridos. Hemos presentado en particular el funcionamiento de la librería RecommenderLab en R, cuya algoritmia y aplicación hemos desarrollado sobre la popular base de datos MovieLens. La comparación entre los resultados de recomendación obtenidos al aplicar diversos algoritmos también la hemos resuelto con las métricas habituales, obteniendo que el filtrado colaborativo basado en usuarios (UBCF) es el mejor algoritmo de recomendación sobre la base de datos MovieLense. Como conclusión general y final, podemos decir que la librería RecommenderLab proporciona herramientas efectivas para desarrollar sistemas de recomendación precisos y robustos, optimizando la experiencia del usuario.

A continuación, mostramos, de un modo esquematizado, los diferentes aspectos, contenidos y aplicaciones desarrolladas en este TFG.

1. **Introducir y definir los sistemas de recomendación, sus tipologías y fundamentos.**
    a. Introducción y evolución de los sistemas de recomendación:
        i. Contextualización. Comenzamos con ejemplos de decisiones cotidianas para resaltar la importancia de la toma de decisiones en nuestra vida diaria.
        ii. Evolución histórica. La descripción de la evolución histórica de los sistemas de recomendación desde la década de 1970 hasta la actualidad, incluyendo hitos importantes y contribuciones de investigadores clave (Elaine Rich, Jussi Karlgren, GroupLens, etc.), ofrece una perspectiva completa de su desarrollo y avances tecnológicos.
    b. Definición de sistemas de recomendación:
        i. Concepto general. Explicamos que un sistema de recomendación es una herramienta que ayuda a los usuarios en la toma de decisiones y reduce las opciones disponibles mientras destaca las más relevantes.
        ii. Fundamentos teóricos. La referencia a la historia y las técnicas fundamentales utilizadas en los sistemas de recomendación (como el filtrado colaborativo y basado en contenido) ayuda a comprender los principios.
    c. Aplicaciones actuales:
        i. Importancia a día de hoy. Explicamos cómo los sistemas de recomendación han transformado nuestra interacción con la información, destacando su papel en plataformas como Amazon, Netflix y YouTube.
        ii. Beneficios. Estos sistemas mejoran la experiencia del usuario, aumentan la satisfacción del cliente y fomentan la fidelidad a la marca, lo cual subraya su valor estratégico en el mundo empresarial.
    d. Tipos de filtrado en sistemas de recomendación:
        i. Filtrado basado en contenido. Explicamos cómo este enfoque utiliza similitudes entre elementos para recomendar contenidos similares a los que el usuario ha mostrado interés previamente.

ii. Filtrado colaborativo. Describimos cómo se basa en las preferencias de usuarios con comportamientos similares para hacer recomendaciones.
iii. Filtrado híbrido. Definimos cómo la combinación de los enfoques anteriores puede superar las limitaciones individuales de cada método.

e. Fundamentos técnicos:
   i. Técnicas y Algoritmos: Describimos técnicas y algoritmos específicos, como el uso de la Similitud de Jaccard, Distancia del Coseno Ajustada, y el algoritmo de K-Nearest Neighbours (K-NN).

2. Revisar los sistemas de recomendación disponibles e implementados en R.

3. Describir en detalle las funciones disponibles en la librería RecommenderLab de R.

a. Implementación de sistemas de recomendación en R:
   i. Librerías específicas de recomendación. Identificamos y describimos varias librerías desarrolladas en R para sistemas de recomendación, destacando RecommenderLab, rectools, recosystem y rrecsys. Cada una se presenta con detalles sobre sus funcionalidades y características.

b. Descripción detallada de librerías y algoritmos:
   i. RecommenderLab. Desarrollamos una descripción extendida de esta librería, incluyendo su funcionalidad, pasos de implementación, y tipos de datos con los que es compatible.
   ii. Algoritmos en RecommenderLab: Enumeramos y describimos en detalle los algoritmos integrados en RecommenderLab, entre ellos está User-based Collaborative Filtering (UBCF), Item-based Collaborative Filtering (IBCF), SVD, Funk SVD, ALS, LIBMF, AR, Popular items, RANDOM, Re-recommend liked items y HybridRecommender.

c. Comparación con Otras Librerías:
   i. Librerías en otros lenguajes: Además de las librerías en R, mencionamos más librerías de sistemas de recomendación implementadas en Python, como LKPY y Surprise.

4. Describir en detalle las funciones disponibles en dicha librería y el modo de implementarlas sobre una base de datos para conseguir un sistema de recomendación, evaluarlo y compararlo con otros.

5. Describir la base de datos MovieLens, disponible en la librería RecommenderLab.

6. Aplicar todas las funciones descritas para la implementación, evaluación y comparación de sistemas de recomendación.

a. Carga de datos. Cargamos la base de datos MovieLense y convertimos los datos en un data.frame.
b. Exploración de datos. Proporcionamos un análisis exploratorio de los datos.
c. Algoritmos de recomendación. Explicamos los algoritmos de recomendación que vamos a utilizar. Ellos son el filtrado colaborativo basado en usuarios (UBCF), basado en ítems (IBCF), popularidad y descomposición en valores singulares (SVD). Permite crear modelos de recomendación y generar predicciones para nuevos usuarios o elementos.

d. Evaluación de modelos. Evaluamos y comparamos múltiples algoritmos de recomendación con las métricas anteriormente descritas.
e. Visualización. Visualizamos las conclusiones de la evaluación del modelo con gráficos como el de la curva ROC y gráficos de recuperación de precisión para evaluar el rendimiento de los modelos.

Biblioteca
UNIVERSITAS Miguel Hernández

Bibliografía

(n.d.). R libraries for recommender systems · GitHub. https://gist.github.com/talegari/77c90db326b4848368287e53b1a18e8d

(n.d.). Wikipedia. https://developers.google.com/machine-learning/recommendation/overview/candidate-generation?hl=es-419

ACM Recommender Systems. (2007, Octubre 19-20). *RecSys 2007 (Minnesota)*. RecSys. https://recsys.acm.org/recsys07/

Adomavicius, G. (2005, Junio). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. https://doi.org/10.1109%2FTKDE.2005.99

Alegría, E. (2020, 06 29). *Sistemas Recomendadores con RecommenderLab*. Retrieved 11 23, 2023, from https://rpubs.com/elias_alegria/intro_RecommenderLab

Bagnato, J. I. (2019, 08 27). *Sistemas de Recomendación*. Sistemas de Recomendación. https://www.aprendemachinelearning.com/sistemas-de-recomendacion/

Beel, J., & et al. (2016). aper recommender systems: a literature survey. *International Journal on Digital Libraries*, (17), 305 - 338.

Beel, J., & Langer, S. (2013, Octubre 12). A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, 7 - 14. https://dl.acm.org/doi/10.1145/2532508.2532511

Çoba, L., & Zanker, M. (2017, June). *Rrrecsys library*. CRAN. https://hdl.handle.net/10863/4864

Cortes, D. (2023, Febrero 19). *recometrics: Evaluation Metrics for Implicit-Feedback Recommender Systems*. CRAN. https://cran.r-project.org/web/packages/recometrics/index.html

Cortes, D. (2023, December 9). cmfrec: Collective Matrix Factorization for Recommender Systems. https://cran.r-project.org/web/packages/cmfrec/index.html

Descomposición en valores singulares. (2024, marzo 4). Wikipedia. https://es.wikipedia.org/wiki/Descomposici%C3%B3n_en_valores_singulares

Ekstrand, M. D. (2018). *The LKPY Package for Recommender Systems Experiments*. Computer Science Faculty Publications and Presentations. https://scholarworks.boisestate.edu/cs_facpubs/147/

Error Absoluto Medio. (2022, septiembre 27). Wikipedia. https://es.wikipedia.org/wiki/Error_absoluto_medio

Error Cuadrático Medio. (2023, noviembre 7). Wikipedia. https://es.wikipedia.org/w/index.php?title=Error_cuadr%C3%A1tico_medio&oldid=155143824

Feldman, E., & Tikhonovich, D. (2022, Marzo). *RecTools: RecTools - library to build Recommendation Systems easier and faster than ever before*. GitHub. https://github.com/MobileTeleSystems/RecTools

Fernández Jauregui, A. (n.d, n.d n.d). *Cómo programar un sistema de recomendación en R - Ander Fernández*. Cómo programar un sistema de recomendación en R - Ander Fernández. https://anderfernandez.com/blog/programar-sistema-recomendacion-r/

Google for Developers. (2022, 09 27). *Machine Learning | Sistemas de Recomendación*. Wikipedia. https://developers.google.com/machine-learning/recommendation/overview/candidate-generation?hl=es-419

Hahsler, M. (n.d.). *RecommenderLab: An R Framework for Developing and Testing Recommendation Algorithms*. CRAN. Retrieved November 22, 2023, from https://cran.r-project.org/web/packages/RecommenderLab/vignettes/RecommenderLab.pdf

Hahsler, M. (2022, May 24). *RecommenderLab: An R Framework for Developing and Testing Recommendation Algorithms*. arXiv. https://arxiv.org/abs/2205.12371

Hahsler, M. (2022, Mayo 31). *RecommenderLabBX: Book-Crossing Dataset (BX) for 'RecommenderLab'*. CRAN. https://cran.r-project.org/web/packages/RecommenderLabBX/index.html

Hahsler, M. (2022, Mayo 31). *RecommenderLabJester: Jester Dataset for 'RecommenderLab'*. CRAN. https://cran.r-project.org/web/packages/RecommenderLabJester/index.html

Herlocker, J. L. (2004, Enero 1). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, *22*(1), 5–53. https://dl.acm.org/doi/10.1145/963770.963772

History of recommender systems: overview of information filtering solutions. (n.d.). Onespire Ltd. https://onespire.net/history-of-recommender-systems/

Hug, N. (2020, Marzo 2). Surprise: A Python library for recommender systems. *Journal of Open Source Software*, *5*(52). 10.21105/joss.02174

Husson, F. (2023, October 12). *Package FactoMineR*. CRAN. Retrieved February 26, 2024, from https://cran.r-project.org/web/packages/FactoMineR/index.html

Husson, F. J. (2008). *actoMineR: A Package for Multivariate Analysis*. Journal of Statistical Software. https://cran.r-project.org/web/packages/FactoMineR/index.html

Ihaka, R. (n.d.). *R (lenguaje de programación)*. Wikipedia. Retrieved February, 2024, from https://es.wikipedia.org/wiki/R_(lenguaje_de_programaci%C3%B3n)

Ihaka, R. (2024, marzo 23). *R (lenguaje de programación)*. Wikipedia. https://es.wikipedia.org/wiki/R_(lenguaje_de_programaci%C3%B3n)

Karlgren, J. (2017, October 1). *A digital bookshelf: original work on recommender systems*. Jussi Karlgren.

https://jussikarlgren.wordpress.com/2017/10/01/a-digital-bookshelf-original-work-on-recommender-systems/

Kassambara, A. (2020). *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. CRAN. https://cran.r-project.org/web/packages/factoextra/index.html

Khoshgoftaar, T. M., & Xiaoyuan Su. (2009). *A Survey of Collaborative Filtering Techniques*. Hindawi. https://www.hindawi.com/journals/aai/2009/421425/

Kohavi, R., & Provost, F. (1998). Glossary of Terms. *Machine Learning*, *30*(2-3), 271–274.

Kuhn, M. (2008). *Building Predictive Models in R Using the caret Package*. Journal of Statistical Software. https://cran.r-project.org/web/packages/caret/index.html

Kuhn, M. (2023, March 21). *Package caret*. CRAN. Retrieved February 26, 2024, from https://cran.r-project.org/web/packages/caret/index.html

Kunz, N. (2019, Julio 2). *crassmat: Conditional Random Sampling Sparse Matrices*. CRAN. https://cran.r-project.org/web/packages/crassmat/index.html

K vecinos más próximos. (2023, septiembre 11). Wikipedia. https://es.wikipedia.org/wiki/K_vecinos_m%C3%A1s_pr%C3%B3ximos

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems*, *74*, 12-32. https://doi.org/10.1016/j.dss.2015.03.008.

mhahsler/RecommenderLab: RecommenderLab - Lab for Developing and Testing Recommender Algorithms - R package. (n.d.). GitHub. https://github.com/mhahsler/RecommenderLab

Montaner, M. (2003, Junio). A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review.*, *19*, 285–330. https://link.springer.com/article/10.1023/A:1022850703159

Pepa, S. M. (2014, Mayo). *Suite de algoritmos de recomendación en aplicaciones reales*. Biblos-e Archivo.

https://repositorio.uam.es/bitstream/handle/10486/660903/marina_pepa_sofia_tfg.pdf?sequence=1

Portugal, I., Alencar, P., & Cowan, D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, *97*, 205-227. https://doi.org/10.1016/j.eswa.2017.12.020

P . Resnick, N. Iacovou, M. Suchak, P. Bergstrom, & J. Riedl. (1994). "Grouplens: an open architecture for collaborative filtering of netnews". *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 175–186.

Qiu, Y. (2023, May 5). *recosystem: Recommender System using Matrix Factorization*. recosystem: Recommender System using Matrix Factorization. https://cran.r-project.org/web/packages/recosystem/index.html

Raíz del Error Cuadrático Medio. (2020, mayo 6). Wikipedia. Retrieved April 20, 2024, from https://es.wikipedia.org/wiki/Ra%C3%ADz_del_error_cuadr%C3%A1tico_medio

Real, R., & Vargas, J. M. (1996, Septiembre). The Probabilistic Basis of Jaccard's Index of Similarity. 380-385. https://www.researchgate.net/profile/Raimundo_Real/publication/239604848_The_Probabilistic_Basis_of_Jaccard's_Index_of_Similarity/links/0c9605268d8ff04ab1000000.pdf

RecommenderLab package. (2021, February 26). RDocumentation. https://www.rdocumentation.org/packages/RecommenderLab/versions/0.2-7

Recommender system. (2024, Abril 24). Wikipedia. https://en.wikipedia.org/wiki/Recommender_system

Rich, E. (1979). User modeling via stereotypes. *Cognitive science*, *4*, 329 - 354.

Ripley, B. (2002). *Modern Applied Statistics with S*. CRAN. https://cran.r-project.org/web/packages/MASS/index.html

Ripley, B. (2024, January 13). *Package MASS*. CRAN. Retrieved February 26, 2024, from https://cran.r-project.org/web/packages/MASS/index.html

Salton, G., & McGill, M. (1983). Introduction to Modern Information Retrieval.

Sangüesa Pérez, D. (2019). *Evaluación comparativa de diferentes métodos para sistemas de recomendación aplicados a la movilidad urbana y su posible inclusión en la cadena de negocio*. https://core.ac.uk/download/pdf/290002244.pdf

Similitud coseno. (2022, diciembre 14). Wikipedia. https://es.wikipedia.org/wiki/Similitud_coseno

Sistemas de recomendación: ¿qué son y cómo funcionan? (2023, May 23). Máster en Inteligencia Artificial, Fundamentos y Aplicaciones - UPM. https://mfaia.ws.fi.upm.es/sistemas-de-recomendacion-personalizando-la-experiencia-del-usuario-en-la-era-digital/

Sohail, S. S., Ali, R., & Siddiqui,, J. (2017). Classifications of recommender systems: A review. *ournal of Engineering Science & Technology Review*, *10*(4). https://www.academia.edu/download/68334157/Classifications_of_Recommender_Systems.pdf

Valdiviezo, P. M. (n.d.). *Sistema recomendador híbrido basado en modelos probabilísticos*. Archivo Digital UPM. https://oa.upm.es/57250/1/PRISCILA_MARISELA_VALDIVIEZO_DIAZ_2.pdf

van Rijsbergen C, C. (1979). Information retrieval.

Vig, J., Sen, S., & Riedl, J. (2012). The Tag Genome: Encoding Community Knowledge to Support Novel Interaction. *ACM Trans. Interact.* https://doi.org/10.1145/2362394.2362395

Vineela, A., G. Lavanya Devi, Nelaturi, N., & Yadav, G. D. (2024, March 5). *A Comprehensive Study and Evaluation of Recommender Systems*. SpringerLink. https://link.springer.com/chapter/10.1007/978-981-15-3828-5_5

Wijffels, J. (2022, July 17). *Package RMOA*. CRAN.

https://cran.r-project.org/web/packages/RMOA/index.html

Biblioteca
UNIVERSITAS Miguel Hernández

Anexos

```
# Análisis exploratorio.

if(!"recommenderlab" %in% rownames(installed.packages())){
  install.packages("recommenderlab")
}
library(recommenderlab)
library(ggplot2)
library(tidyverse)

data("MovieLense")
class(MovieLense); MovieLense

slotNames(MovieLense)
class(MovieLense@data)

movies <- quantile(colCounts(MovieLense), c(0.25, 0.5, 0.75, 1)); movies

users <- quantile(rowCounts(MovieLense), c(0.25, 0.5, 0.75, 1)); users

image(MovieLense)

vector_ratings <- as.vector(MovieLense@data)
unique(vector_ratings)

table_ratings <- table(vector_ratings); table_ratings

ratings_0 <- (table_ratings[1] / sum(table_ratings))*100
cat("Teniendo en cuenta que el porcentaje de valoraciones inexistentes es de ", ratings_0, "%, eliminamos el valor 0 que representa valores omitidos.")

vector_ratings <- vector_ratings[vector_ratings != 0]
vector_ratings_porc <- table(vector_ratings); vector_ratings_porc

ratings_1 <- (vector_ratings_porc[1] / sum(vector_ratings_porc))*100; ratings_1
ratings_2 <- (vector_ratings_porc[2] / sum(vector_ratings_porc))*100; ratings_2
ratings_3 <- (vector_ratings_porc[3] / sum(vector_ratings_porc))*100; ratings_3
ratings_4 <- (vector_ratings_porc[4] / sum(vector_ratings_porc))*100; ratings_4
ratings_5 <- (vector_ratings_porc[5] / sum(vector_ratings_porc))*100; ratings_5

library(ggplot2)
df <- data.frame(ratings = vector_ratings)
df_table <- as.data.frame(table(df$ratings))
```

Biblioteca Universitas Miguel Hernández

```
names(df_table) <- c("ratings", "frequency")
df_table$percentage <- (df_table$frequency / sum(df_table$frequency)) * 100

qplot(vector_ratings,
      fill = I("lightblue"),
      color = I("lightblue"),
      alpha = I(0.7),
      xlab = "Valoraciones",
      ylab = "Frecuencia") + theme_classic() +
        geom_text(data = df_table,
          aes(x = as.numeric(ratings), y = frequency, label = paste0(round(percentage, 2), "%")),
          vjust = -0.5, size = 5, color = "black")

views_per_movie <- colCounts(MovieLense)
table_views <- data.frame(
  movie = names(views_per_movie),
  views = views_per_movie)
table_views <- table_views[order(table_views$views, decreasing = TRUE), ]
#head(table_views)

install.packages("RColorBrewer")
library(RColorBrewer)
table_views <- within(table_views, {
  percentage <- views
})

ggplot(table_views[1:6, ], aes(x = movie, y = views, fill = movie)) +
  geom_bar(stat = "identity", size = 0.7) +
    geom_text(aes(label = paste0(round(percentage, 1))), vjust = -1, size = 3.5, color = "black",
position = position_stack(vjust = 0.5)) +
  #ggtitle("Número de visualizaciones de las mejores películas") +
  labs(y = "Visualizaciones", x = "Película") +
  coord_flip() +
  scale_fill_brewer(palette = "Paired") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, hjust = 3),
        legend.position = "none")

average_ratings <- data.frame(colMeans(MovieLense))
colnames(average_ratings) = "Rate"

install.packages("viridis")
library(viridis)
```

Biblioteca
UNIVERSITAS Miguel Hernández

```
Conteo <- data.frame(Genre = names(apply(MovieLenseMeta[, 4:22], 2, sum)),
                     Count = apply(MovieLenseMeta[, 4:22], 2, sum))
ggplot(Conteo, aes(x = Genre, y = Count, fill = Genre)) +
  geom_bar(stat = "identity", size = 0.7) +
    geom_text(aes(label = Count), hjust = -0.3, size = 3.5, color = "black") +
  #ggtitle("Géneros que se han clasificado") +
  labs(y = "Clasificación", x = "Género") +
  coord_flip() +
  scale_fill_viridis_d() +
  theme_classic() +
  theme(legend.position = "none")

MovieLenseUser$Sex <- ifelse(MovieLenseUser[, 3] == "M", "Hombre",
                             ifelse(MovieLenseUser[, 3] == "F", "Mujer", MovieLenseUser[, 3]))
Conteo <- data.frame(Sex = names(table(MovieLenseUser$Sex)),
                     Count = as.numeric(table(MovieLenseUser$Sex)))
Conteo$Percentage <- (Conteo$Count / sum(Conteo$Count)) * 100
Conteo$Sex <- factor(Conteo$Sex, levels = Conteo$Sex)
ggplot(Conteo, aes(x = Sex, y = Count, fill = Sex)) +
  geom_bar(stat = "identity", size = 0.7) +
   geom_text(aes(label = paste0(round(Percentage, 1), "%")), vjust = 0.5, size = 3.5, color = "black",
position = position_stack(vjust = 0.5)) +
 # ggtitle("Clasificación por Género") +
  labs(y = "Clasificación", x = "Género") +
  coord_flip() +
  scale_fill_brewer(palette = "Set3") +
  theme_classic() +
  theme(legend.position = "none")

head(MovieLenseUser)

ggplot(MovieLenseUser, aes(x=age, color=Sex, fill=Sex)) +
  geom_histogram(alpha=0.5, bins=15, position="identity") +
  theme_classic()

# Creamos el sistema de recomendación.

movie_ratings <- as(MovieLense, "realRatingMatrix")
ratings <- as(movie_ratings, "data.frame")

str(ratings)
head(ratings)
```

```
recommenderRegistry$get_entries(dataType = "realRatingMatrix")

ML_POPULAR <- Recommender(MovieLense[1:941], method = "POPULAR"); ML_POPULAR

names(getModel(ML_POPULAR))

getModel(ML_POPULAR)$topN

recom <- predict(ML_POPULAR,MovieLense[941:943], n=5); recom

as(recom, "list")

recom3 <- bestN(recom, n=3); recom3

as(recom3, "list")

recom <- predict(ML_POPULAR, MovieLense[941:943], type = "ratings")
as(recom, "matrix")[,1:5]

recom <- predict(ML_POPULAR, MovieLense[941:943], type = "ratingMatrix"); recom

as(recom, "matrix")[,1:5]

e <- evaluationScheme(MovieLense, method = "split", train = 0.9, given=15, goodRating=5); e

r1 <- Recommender(getData(e, "train"), "UBCF"); r1

r2 <- Recommender(getData(e, "train"), "IBCF"); r2

p1 <- predict(r1, getData(e, "known"), type="ratings"); p1

p2 <- predict(r2, getData(e, "known"), type="ratings"); p2

error <- rbind(
  UBCF = calcPredictionAccuracy(p1, getData(e, "unknown")),
  IBCF = calcPredictionAccuracy(p2, getData(e, "unknown"))
); error

scheme <- evaluationScheme(MovieLense, method="cross", k=4, given=3, goodRating=5); scheme

results <- evaluate(scheme, method="POPULAR", type = "topNList", n=c(1,3,5,10,15,20)); results

getConfusionMatrix(results)[[1]]
```

```
avg(results)

plot(results, annotate=TRUE)

plot(results, "prec/rec", annotate=TRUE)

set.seed(2016)
scheme <- evaluationScheme(MovieLense, method = "split", train = 0.9, k=1, given=5, goodRating
= 5)
scheme

algorithms <- list(
  "random items" = list(name="RANDOM", param=NULL),
  "popular items" = list(name="POPULAR", param=NULL),
  "user-based CF" = list(name="UBCF", param=list(nn=50)),
  "item-based CF" = list(name="IBCF", param=list(k=50)),
  "SVD approximation" = list(name="SVD", param=list(k = 50))
)

results <- evaluate(scheme, algorithms, type="topNList", n=c(1,3,5,10,15,20)); results

names(results)

results[["user-based CF"]]

plot(results, annotate=c(1,3), legend="bottomright")

plot(results, "prec/rec", annotate=3, legend="topleft")
results <- evaluate(scheme, algorithms, type="ratings"); results

colors <- c("red", "blue", "green", "orange", "purple")
plot(results, ylim=c(0, 5), col=colors, pch=19)

MovieLense_binary <- binarize(MovieLense, minRating=5)
MovieLense_binary <- MovieLense_binary[rowCounts(MovieLense_binary)>20]
MovieLense_binary

scheme_binary <- evaluationScheme(MovieLense_binary, method="split", train=.9, k=1, given=3);
scheme_binary

results_binary <- evaluate(scheme_binary, algorithms, type = "topNList", n=c(1,3,5,10,15,20))

plot(results_binary, annotate=c(1,3), legend="topright")
```

Biblioteca
UNIVERSITAS Miguel Hernández