GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

OPTIMIZACIÓN DE SISTEMAS RECOMENDADORES PARA LA PERSONALIZACIÓN DE LA EXPERIENCIA DE VISUALIZACIÓN DE PELÍCULAS

Autor: Inés Gómez Fortis

Director: Carlos Castro Rey

Madrid, julio de 2023

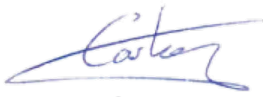
Inés Gómez Fortis, declara bajo su responsabilidad, que el Proyecto con título **Optimización de Sistemas Recomendadores para la personalización de la experiencia de visualización de películas** presentado en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2022/23 es de su autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.:  Fecha: 3 / 7 / 2023

Autoriza la entrega:

El Director del Proyecto

Carlos Castro Rey

Fdo.:  Fecha: 3 / 7 / 2023

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

OPTIMIZACIÓN DE SISTEMAS RECOMENDADORES PARA LA PERSONALIZACIÓN DE LA EXPERIENCIA DE VISUALIZACIÓN DE PELÍCULAS

Autor: Inés Gómez Fortis

Director: Carlos Castro Rey

Madrid, julio de 2023

OPTIMIZACIÓN DE SISTEMAS RECOMENDADORES PARA LA PERSONALIZACIÓN DE LA EXPERIENCIA DE VISUALIZACIÓN DE PELÍCULAS

Autor: Gómez Fortis, Inés
Director: Castro Rey, Carlos
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En la actualidad, muchas empresas han adoptado sistemas recomendadores basados en plataformas digitales para aprovechar la amplia cantidad de datos generados por la interacción en línea. Estos datos ofrecen una valiosa oportunidad para mejorar constantemente la personalización de las recomendaciones y los servicios ofrecidos. Como resultado, los sistemas recomendadores desempeñan un papel fundamental al facilitar a los usuarios el descubrimiento de nuevos productos, servicios o contenido relevante, mejorando así su experiencia y satisfacción.

Este proyecto tiene como objetivo comparar y evaluar las diversas técnicas y algoritmos de recomendación disponibles en la actualidad. Para lograrlo, se realiza un análisis exhaustivo de técnicas como el enfoque basado en contenido, el filtrado colaborativo, las técnicas de factorización de matrices y algunas técnicas de Deep Learning. Sin embargo, este trabajo va más allá de un análisis teórico de los modelos existentes, ya que se lleva a cabo la tarea de desarrollar los algoritmos desde cero. Este proceso implica la recolección y el preprocesamiento de datos, así como la construcción y evaluación de los modelos.

A través de la implementación de estos algoritmos, se busca identificar el mejor modelo de recomendación. Para ello, se establecen métricas offline que permiten medir y comparar el rendimiento de los diferentes modelos. Además, se realiza una simulación en un entorno de producción en vivo utilizando el framework Dash. Esta simulación demuestra cómo el sistema de recomendación puede operar de manera interactiva y en tiempo real, dado que los usuarios tienen la posibilidad de recibir recomendaciones personalizadas y relevantes mientras interactúan con la interfaz del sistema.

En resumen, este trabajo explora y compara las técnicas existentes en los sistemas recomendadores con el propósito de desarrollar un recomendador personalizado y llevarlo a un entorno de producción en vivo de manera que se demuestra la aplicabilidad práctica y la importancia de estos sistemas en la mejora de la experiencia del usuario.

Palabras clave: sistemas de recomendación, técnicas, arranque en frío, MovieLens, métricas de evaluación offline, entorno de producción

1. Introducción

Desde la aparición de Internet, el comercio electrónico ha experimentado un crecimiento significativo, impulsado por muchas empresas que han adoptado modelos de negocio basados en plataformas digitales. Estas empresas aprovechan la gran cantidad de datos generados por la interacción en línea para mejorar la interacción con los clientes y tomar decisiones estratégicas que impulsen el crecimiento empresarial.

Los sistemas recomendadores analizan los datos que los usuarios registrados en plataformas en línea y los clasifican en grupos según sus preferencias, ofreciendo recomendaciones personalizadas que mejoran la experiencia del usuario y aumentarn la satisfacción del cliente [1]. Cada vez más negocios optan por digitalizarse y aventurarse en el comercio en línea, invirtiendo en sistemas de recomendación [2]. Hoy en día se encuentran recomendadores en diversas industrias, desde el comercio electrónico hasta la música y el entretenimiento, y desempeñan un papel fundamental en el comercio en línea al personalizar el servicio. Como consecuencia, los sistemas recomendadores han ganado un prominente interés en la actualidad y se han convertido en el núcleo de muchas empresa que operan en Internet.

El objetivo de este trabajo es abordar el desafío de la recomendación, centrándose especialmente en el ámbito del contenido multimedia, en particular, las películas. Se busca analizar las diversas técnicas existentes con el fin de encontrar aquellas que brinden los mejores resultados, es decir, sean capaces de generar sugerencias que satisfagan a los clientes.

2. Metodología

Este trabajo representa un análisis exhaustivo de los sistemas de recomendación. A lo largo de este estudio, se han examinado meticulosamente diversas técnicas y procedimientos existentes, con el objetivo principal de desarrollar un modelo desde cero capaz de proporcionar recomendaciones de películas basadas en las preferencias de los usuarios.

El proyecto comienza con un período de familiarización y aprendizaje de los conceptos fundamentales relacionados con los sistemas de recomendación, las diferentes técnicas existentes, su implementación y sus aplicaciones. Posteriormente, se desarrollan una serie de algoritmos de recomendación de dificultad baja y media, como el sistema basado en contenido, utilizando el conjunto de datos de MovieLens. Este conjunto de datos es una base de datos popular y completa sobre películas y recomendaciones de usuarios, utilizada para analizar patrones de preferencia y desarrollar algoritmos de recomendación personalizada. Este enfoque permite adquirir experiencia en el desarrollo, entrenamiento y evaluación de sistemas de recomendación.

Es importante destacar que para el procesamiento y manejo eficiente de los datos utilizados en el proyecto se ha creado la clase MovieLens. Esta clase proporciona funcionalidades esenciales relacionadas con la carga de datos, la obtención de información detallada sobre películas y la generación de recomendaciones basadas en las preferencias de los usuarios. Al utilizar la clase MovieLens, se accede a una interfaz eficiente que facilita la obtención de información relevante, como detalles de películas y clasificaciones de usuarios, y permite generar recomendaciones personalizadas basadas en las preferencias individuales de cada usuario.

Una vez adquiridos los conocimientos necesarios sobre los recomendadores, se procede a desarrollar modelos más complejos empleando técnicas de Deep Learning. Cabe destacar que se ha tomado como referencia el curso *Building Recommender Systems with Machine Learning and AI* [3] para la elaboración de las diferentes técnicas de recomendación.

En total, se han desarrollado siete algoritmos de recomendación, incluyendo un sistema basado en contenido, dos filtros colaborativos (basado en usuarios y basado en elementos), dos técnicas de factorización de matrices (SVD y SVD++) y dos técnicas de Deep Learning, las RBM y el AutoRec.

Después de haber desarrollado todos los modelos, se procede a evaluar su rendimiento con el objetivo de analizar las ventajas e inconvenientes de cada una de las técnicas implementadas. Este análisis exhaustivo permite obtener una visión completa del rendimiento de los diferentes recomendadores en base a los criterios que deben cumplir las recomendaciones para satisfacer las necesidades y preferencias de los usuarios.

Finalmente, se ha desarrollado una aplicación que permite evaluar de manera práctica y dinámica el rendimiento del recomendador con las mejores recomendaciones en nuevos usuarios. Esta aplicación proporciona una experiencia interactiva que facilitará la evaluación y comprensión de las capacidades del sistema de recomendación implementado.

Todo el código desarrollado durante el proyecto se encuentra en el siguiente repositorio de GitHub : https://github.com/inesgfortis/RecommenderSys

3. Resultados

Con el objetivo de identificar las técnicas más destacadas en términos de precisión, relevancia, diversidad y novedad en la recomendación de películas, se han establecido una serie de métricas offline que permiten evaluar el desempeño de cada algoritmo en base a dichas métricas seleccionadas. Además, se pretende obtener una visión completa del rendimiento de cada algoritmo y realizar comparaciones significativas entre ellos. A continuación se presentan los resultados obtenidos en cada modelo:

Cuadro 1: Métricas de rendimiento de los modelos

Modelo	RMSE	MAE	MAP	MAR	Avg.NDCG	Coverage	User Cov.	Novelty
user-based	0.9030	0.6864	0.7299	0.4667	0.9510	0.0543	0.9967	4231
item-based	0.9172	0.6978	0.7430	0.4480	0.9518	0.0207	1	5854
SVD	0.8802	0.6762	0.7370	0.4480	0.9539	0.0384	0.9180	421
SVD++	0.8688	0.6642	0.7438	0.4495	0.9544	0.0345	0.9082	694
RBM	1.1620	0.9641	0.1630	0.0079	0.9360	0.0028	0	667
AutoRec	2.6319	2.3002	0.5348	0.1272	0.9334	0.0188	1	2701
Random	1.4320	1.1429	0.6299	0.2937	0.9312	0.0297	1	1846

El Cuadro 1 proporciona una visión general del rendimiento de diferentes modelos de recomendación en términos de diversas métricas de evaluación offline. Estas métricas han sido específicamente seleccionadas para la evaluación de los modelos, ya que ayudan a comprender cómo se desempeñan en áreas clave de la recomendación, como la precisión, relevancia, diversidad y novedad de las recomendaciones.

En general, se puede observar que los modelos SVD++, user-based y item-based obtienen buenos resultados en la mayoría de las métricas evaluadas. Estos modelos han logrado un equilibrio entre la precisión y la exhaustividad al recomendar elementos relevantes a los usuarios. Además, han demostrado una capacidad sólida para predecir las calificaciones de los usuarios, con valores bajos de RMSE (Root Mean Square Error) y MAE (Mean Absolute Error), lo que indica que son capaces de proporcionar recomendaciones precisas y útiles.

En cuanto a la capacidad para ofrecer recomendaciones diversas, los modelos user-based y item-based han mostrado una mayor cobertura, lo que significa que son capaces de recomendar una amplia gama de elementos a los usuarios. Esto sugiere que estos modelos pueden satisfacer las necesidades de diferentes usuarios al ofrecer opciones variadas.

En términos de novedad de las recomendaciones, el modelo item-based ha demostrado un buen desempeño al ofrecer recomendaciones más novedosas en comparación con otros modelos. Esto indica que el modelo es capaz de explorar nuevos elementos y presentar opciones interesantes a los usuarios. Por otro lado, los modelos SVD, RBM y SVD++ muestran valores más bajos de Novelty, lo que indica que sus recomendaciones se ajustan más a los gustos y preferencias previas del usuario.

Por otro lado, es importante destacar que los modelos RBM y AutoRec presentan un rendimiento inferior en varias métricas. Estos modelos podrían requerir mejoras o ajustes para alcanzar un rendimiento más sólido y efectivo en la generación de recomendaciones.

En base a los resultados obtenidos en las métricas offline seleccionadas, se puede concluir que el **modelo SVD++ muestra el mejor rendimiento general en comparación con los otros algoritmos**. Obtiene buenos resultados en términos de precisión (RMSE, MAE), calidad de las recomendaciones (MAP, Mean NDCG), capacidad de recuperación de elementos relevantes (MAR) y User Coverage. Sin embargo, es importante tener en cuenta que estos resultados se basan en métricas offline y pueden no reflejar completamente el rendimiento en un entorno en vivo. Por lo tanto, es recomendable realizar pruebas adicionales y considerar otros factores antes de seleccionar el modelo más adecuado para un sistema de recomendación de películas.

Hay que recordar que también se ha desarrollado un algoritmo basado en contenido que se centra únicamente en los atributos de las películas, en este caso los géneros, sin considerar las calificaciones de los usuarios. Debido a su naturaleza y características, la evaluación de este algoritmo difiere de la de los demás modelos y, por lo tanto, no se ha incluido en la tabla de métricas del estudio (Cuadro 1). Este algoritmo se evalúa en base a unas métricas específicas que permiten comparar las recomendaciones del algoritmo con las películas populares entre usuarios que han disfrutado de películas similares a las de referencia para la recomendación. Es un modelo muy sencillo, por lo que, aunque este enfoque pueda proporcionar recomendaciones iniciales, es probable que sea menos efectivo en comparación con otros algoritmos más sofisticados que tienen en cuenta una variedad de factores y características de las películas, así como las preferencias individuales de los usuarios.

Además, se ha realizado también un **análisis de las recomendaciones** para evaluar la efectividad de los diferentes algoritmos de recomendación considerando la diversidad en los gustos y experiencias de los usuarios. Para lograr esto, se han seleccionado dos usuarios con perfiles y preferencias diferentes, lo que implica que tienen gustos y experiencias distintas en el contexto del sistema de recomendación. En concreto, se han comparado las recomendaciones obtenidas de los modelos SVD++ y RBM.

Al analizar las recomendaciones y compararlas con las métricas de evaluación, **se observa una correspondencia entre el rendimiento de los modelos en las métricas y la adecuación de las recomendaciones a las preferencias del usuario**. El modelo que muestra un mejor rendimiento en las métricas (SVD++) también genera recomendaciones más acertadas y alineadas con las preferencias del usuario, mientras que el modelo con un desempeño inferior en las métricas (RBM) ofrece recomendaciones menos adecuadas en este caso. Esto respalda la conexión entre las métricas de evaluación y la calidad de las recomendaciones proporcionadas por los modelos.

Finalmente, para poner en práctica y comprobar en un entorno en vivo el rendimiento del modelo que ha mostrado un mejor rendimiento en las métricas offline, **se ha desarrollado un Dash interactivo** para evaluar de forma online los algoritmos de recomendación. Este Dash tiene como objetivo simular un entorno vivo en el cual los usuarios pueden experimentar la funcionalidad del sistema recomendador y recibir recomendaciones personalizadas en tiempo real.

4. Conclusiones

En conclusión, los sistemas de recomendación son herramientas poderosas que utilizan diversos algoritmos para ofrecer sugerencias personalizadas a los usuarios. Existen múltiples enfoques y algoritmos de recomendación, cada uno con sus ventajas e inconvenientes. No hay un "mejor"recomendador universal, ya que la elección del algoritmo depende del contexto, el problema que se desee abordar y los datos y tecnologías disponibles.

Si bien las técnicas de evaluación offline proporcionan una forma útil de medir el rendimiento de los algoritmos, es importante recordar que la verdadera prueba de un sistema de recomendación radica en la satisfacción del usuario final. Un recomendador se considera exitoso si los usuarios están contentos con las recomendaciones que reciben y si estas recomendaciones cumplen sus necesidades y expectativas.

Por lo tanto, es valioso evaluar los algoritmos de recomendación en entornos vivos, como a través de la implementación de un dashboard que simule un entorno de producción. Esto permite obtener retroalimentación directa de los usuarios finales y evaluar cómo se reciben y utilizan las recomendaciones en un entorno real.

En última instancia, el objetivo principal de un sistema de recomendación es mejorar la experiencia del usuario y ayudar a los usuarios a descubrir y encontrar contenido relevante de manera eficiente. Para lograr esto, es fundamental considerar las necesidades y preferencias de los usuarios y buscar constantemente mejorar y adaptar los algoritmos de recomendación para brindar una experiencia satisfactoria y personalizada.

5. Referencias

[1] Aggarwal CC, et al. Recommender systems. vol. 1. Springer; 2016.

[2] Lü L, Medo M, Yeung CH, Zhang YC, Zhang ZK, Zhou T. Recommender systems. Physics reports. 2012;519(1):1-49.

[3] Sundog Education by Frank Kane. Building Recommender Systems with Machine Learning and AI; 2022. Online. Available from: `https://www.udemy.com/course/building-recommender-systems-with-machine-learning-and-ai/`.

OPTIMIZATION OF RECOMMENDER SYSTEMS FOR THE PERSONALIZATION OF THE MOVIE VIEWING EXPERIENCE

Author: Gómez Fortis, Inés
Supervisor: Castro Rey, Carlos
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Currently, many companies have adopted recommender systems based on digital platforms to take advantage of the vast amount of data generated by online interactions. This data provides a valuable opportunity to continuously improve the personalization of recommendations and services offered. As a result, recommender systems play a crucial role in facilitating users' discovery of new products, services, or relevant content, thereby enhancing their experience and satisfaction.

This project aims to compare and evaluate various recommendation techniques and algorithms available today. To achieve this, a comprehensive analysis is conducted on techniques such as content-based approaches, collaborative filtering, matrix factorization techniques, and some deep learning techniques. However, this work goes beyond a theoretical analysis of existing models as it involves the task of developing the algorithms from scratch. This process entails data collection and preprocessing, as well as model construction and evaluation.

Through the implementation of these algorithms, the goal is to identify the best recommendation model. To do so, offline metrics are established that allow measuring and comparing the performance of different models. Additionally, a simulation is performed in a live production environment using the Dash framework. This simulation demonstrates how the recommender system can operate interactively and in real-time, as users have the ability to receive personalized and relevant recommendations while interacting with the system interface.

In summary, this work explores and compares existing techniques in recommender systems with the purpose of developing a personalized recommender and deploying it in a live production environment to demonstrate the practical applicability and importance of these systems in enhancing the user experience.

Key words: recommender systems, techniques, cold start problem, MovieLens, offline assessment metrics, production environment

1. Introduction

Since the emergence of the Internet, e-commerce has experienced significant growth, driven by many companies that have adopted business models based on digital platforms. These companies leverage the vast amount of data generated by online interaction to improve customer interaction and make strategic decisions that drive business growth.

Recommender systems analyze data from registered users on online platforms and classify them into groups based on their preferences, offering personalized recommendations that enhance the user experience and increase customer satisfaction [1]. More and more businesses are choosing to digitize and venture into online commerce, investing in recommender systems [2]. Nowadays, recommender systems can be found in various industries, from e-commerce to music and entertainment, and they play a crucial role in online commerce by personalizing the service. As a result, recommender systems have gained prominent interest and have become the core of many companies operating on the Internet.

The objective of this work is to address the challenge of recommendation, focusing specifically on the field of multimedia content, particularly movies. The aim is to analyze the various existing techniques in order to find those that provide the best results, i.e., capable of generating suggestions that satisfy customers.

2. Methodology

This work represents a comprehensive analysis of recommender systems. Throughout this study, various existing techniques and procedures have been meticulously examined with the main objective of developing a model from scratch capable of providing movie recommendations based on user preferences.

The project begins with a familiarization and learning period of the fundamental concepts related to recommender systems, the different existing techniques, their implementation, and their applications. Subsequently, a series of low and medium difficulty recommendation algorithms are developed, such as the content-based system, using the MovieLens dataset. This dataset is a popular and comprehensive database of movies and user recommendations, used to analyze preference patterns and develop personalized recommendation algorithms. This approach allows for gaining experience in the development, training, and evaluation of recommender systems.

It is important to note that for the efficient processing and handling of the data used in the project, the MovieLens class has been created. This class provides essential functionalities related to data loading, obtaining detailed information about movies, and generating recommendations based on user preferences. By using the MovieLens class, an efficient interface is accessed that facilitates obtaining relevant information, such as movie details and user ratings, and enables the generation of personalized recommendations based on individual user preferences.

Once the necessary knowledge about recommenders has been acquired, more complex models are developed using Deep Learning techniques. It is worth mentioning that the course *Building Recommender Systems with Machine Learning and AI* [3] has been used as a reference for the elaboration of the different recommendation techniques.

In total, seven recommendation algorithms have been developed, including a content-based, two collaborative filters (user-based and item-based), two matrix factorization techniques (SVD and SVD++), and two Deep Learning techniques, RBM and AutoRec.

After developing all the models, their performance is evaluated to analyze the advantages and disadvantages of each of the implemented techniques. This comprehensive analysis provides a complete view of the performance of the different recommenders based on the criteria that recommendations must meet to satisfy the needs and preferences of users.

Finally, an application has been developed that allows for practical and dynamic evaluation of the recommender's performance with the best recommendations for new users. This application provides an interactive experience that will facilitate the evaluation and understanding of the capabilities of the implemented recommender system.

All the code developed during the project can be found in the following GitHub repository: https://github.com/inesgfortis/RecommenderSys

3. Results

In order to identify the most prominent techniques in terms of precision, relevance, diversity, and novelty in movie recommendation, a series of offline metrics have been established to evaluate the performance of each algorithm based on these selected metrics. Additionally, the aim is to gain a comprehensive understanding of the performance of each algorithm and make meaningful comparisons between them. The following are the results obtained for each model:

Table 1: Model performance metrics

Model	RMSE	MAE	MAP	MAR	Avg.NDCG	Coverage	User Cov.	Novelty
user-based	0.9030	0.6864	0.7299	0.4667	0.9510	0.0543	0.9967	4231
item-based	0.9172	0.6978	0.7430	0.4480	0.9518	0.0207	1	5854
SVD	0.8802	0.6762	0.7370	0.4480	0.9539	0.0384	0.9180	421
SVD++	0.8688	0.6642	0.7438	0.4495	0.9544	0.0345	0.9082	694
RBM	1.1620	0.9641	0.1630	0.0079	0.9360	0.0028	0	667
AutoRec	2.6319	2.3002	0.5348	0.1272	0.9334	0.0188	1	2701
Random	1.4320	1.1429	0.6299	0.2937	0.9312	0.0297	1	1846

Table 1 provides an overview of the performance of different recommendation models in terms of various offline evaluation metrics. These metrics were specifically selected to evaluate the models as they help understand how they perform in key areas of recommendation, such as precision, relevance, diversity, and novelty of recommendations.

Overall, it can be observed that the SVD++ model, user-based, and item-based models achieve good results in most of the evaluated metrics. These models have achieved a balance between precision and recall by recommending relevant items to users. Additionally, they have demonstrated a strong ability to predict user ratings, with low values of RMSE (Root Mean Square Error) and MAE (Mean Absolute Error), indicating their capability to provide accurate and useful recommendations.

In terms of the ability to offer diverse recommendations, the user-based and item-based models have shown higher coverage, meaning they are capable of recommending a wide range of items to users. This suggests that these models can cater to the needs of different users by offering varied choices.

Regarding the novelty of recommendations, the item-based model has demonstrated good performance by offering more novel recommendations compared to other models. This indicates that the model is capable of exploring new items and presenting interesting options to users. On the other hand, the SVD, RBM, and SVD++ models show lower Novelty values, indicating that their recommendations align more closely with the user's prior tastes and preferences.

However, it is important to note that the RBM and AutoRec models exhibit lower performance in several metrics. These models may require improvements or adjustments to achieve a more robust and effective performance in generating recommendations.

Based on the results obtained in the selected offline metrics, it can be concluded that the SVD++ model shows the best overall performance compared to the other algorithms. It achieves good results in terms of precision (RMSE, MAE), recommendation quality (MAP, Mean NDCG), relevant item retrieval (MAR), and User Coverage. However, it is important to consider that these results are based on offline metrics and may not fully reflect the performance in a live environment. Therefore, additional testing and consideration of other factors are recommended before selecting the most suitable model for a movie recommendation system.

It is worth noting that a content-based algorithm has also been developed, focusing solely on movie attributes, in this case, genres, without considering user ratings. Due to its nature and characteristics, the evaluation of this algorithm differs from that of the other models and thus has not been included in the metrics table (Table 1). This algorithm is evaluated based on specific metrics that allow for comparing the algorithm's recommendations to movies popular among users who have enjoyed similar reference movies for recommendation. It is a simplistic model, so while this approach may provide initial recommendations, it is likely to be less effective compared to more sophisticated algorithms that take into account a variety of movie factors and characteristics, as well as individual user preferences.

Additionally, an analysis of the recommendations has also been conducted to evaluate the effectiveness of different recommendation algorithms considering the diversity in users' tastes and experiences. To achieve this, two users with different profiles and preferences have been selected, implying they have distinct tastes and experiences in the context of the recommendation system. Specifically, the recommendations obtained from the SVD++ and RBM models have been compared.

When analyzing the recommendations and comparing them with the evaluation metrics, there is a correspondence between the models' performance in the metrics and the suitability of the recommendations to the user's preferences. The model that performs better in the metrics (SVD++) also generates more accurate and aligned recommendations with the user's preferences, while the model with lower performance in the metrics (RBM) offers less suitable recommendations in this case. This supports the connection between evaluation metrics and the quality of recommendations provided by the models.

Finally, to put into practice and test the performance of the model that has shown the best performance in the offline metrics, an interactive Dash has been developed to evaluate the recommendation algorithms online. This Dash aims to simulate a live environment in which users can experience the functionality of the recommendation system and receive real-time personalized recommendations.

4. Conclusions

In conclusion, recommendation systems are powerful tools that use various algorithms to provide personalized suggestions to users. There are multiple approaches and recommendation algorithms, each with its own advantages and disadvantages. There is no "universal best"recommender, as the choice of algorithm depends on the context, the specific problem at hand, and the available data and technologies.

While offline evaluation techniques provide a useful way to measure the performance of algorithms, it is important to remember that the true test of a recommendation system lies in the satisfaction of the end user. A recommender is considered successful if users are happy with the recommendations they receive and if these recommendations meet their needs and expectations.

Therefore, it is valuable to evaluate recommendation algorithms in live environments, such as through the implementation of a dashboard that simulates a production environment. This allows for direct feedback from end users and evaluation of how recommendations are received and utilized in a real-world setting.

Ultimately, the main goal of a recommendation system is to enhance the user experience and help users discover and find relevant content efficiently. To achieve this, it is essential to consider the needs and preferences of users and continuously strive to improve and adapt recommendation algorithms to provide a satisfying and personalized experience.

5. References

[1] Aggarwal CC, et al. Recommender systems. vol. 1. Springer; 2016.

[2] Lü L, Medo M, Yeung CH, Zhang YC, Zhang ZK, Zhou T. Recommender systems. Physics reports. 2012;519(1):1-49.

[3] Sundog Education by Frank Kane. Building Recommender Systems with Machine Learning and AI; 2022. Online. Available from: `https://www.udemy.com/course/building-recommender-systems-with-machine-learning-and-ai/`.

A mis padres, a mis amigos y a todos
los que han estado a mi lado a lo largo de estos años.

Por ayudarme a seguir adelante
y recordarme que la actitud es la clave de todo.

Índice general

Índice de figuras

Índice de cuadros

Acrónimos

ETS	Escuela Técnica Superior
ICAI	Insitituto Católico de Artes e Industrias
MAE	Error Absoluto Medio
RMSE	Raíz del Error Cuadrático Medio
MAP	Mean Average Precision
MAR	Mean Average Recal
CG	Cumulative Gain
DCG	Discounted Cumulative Gain
IDCG	Ideal Discounted Cumulative Gain
NDCG	Normalized Discounted Cumulative Gain
KNN	K-Nearest Neighbors
TF-IDF	Term Frequency-Inverse Document Frequency
RBM	Restricted Boltzmann Machines
SVD	Singular Value Decomposition
CNN	Redes Neuronales Convolucionales
RNN	Redes Neuronales Recurrentes
DNN	Redes Neuronales Profundas
FM	Factorization Machines
NLP	Natural Language Processing
ODS	Objetivos de Desarrollo Sostenible
ONU	Organización de las Naciones Unidas

Capítulo 1

Introducción

Desde la aparición de Internet, muchas empresas han optado por utilizar las nuevas tecnologías como base de su negocio. Esto se refleja en el crecimiento del comercio electrónico, también conocido como *e-commerce*, que está experimentando una expansión significativa.

En la actualidad, muchas empresas han adoptado modelos de negocio basados en plataformas digitales, aprovechando la vasta cantidad de datos generados por la interacción en línea. Estos datos ofrecen una oportunidad invaluable para mejorar continuamente la interacción con los clientes y los servicios ofrecidos. Mediante el análisis de estos datos, las empresas pueden tomar decisiones estratégicas que impulsan la mejora de la experiencia del usuario y fomentan el crecimiento empresarial. El uso efectivo de esta información no solo permite identificar oportunidades de mejora, sino que también impulsa la optimización de la empresa en su conjunto.

Esta es precisamente la idea detrás de un sistema recomendador: organizar la información disponible para convertir los datos sobre los usuarios y sus preferencias en predicciones de sus posibles gustos e intereses futuros, con el objetivo de obtener el máximo beneficio posible [1].

Un sistema recomendador consiste en un sistema capaz de analizar los datos de los usuarios registrados en una plataforma en línea o sitio web que ofrece un servicio determinado, y clasificar a estos usuarios en grupos según sus preferencias. El objetivo es ofrecer recomendaciones personalizadas que mejoren la experiencia del usuario.

Estos sistemas son aplicables a una amplia gama de industrias. La mayoría de los sitios web de comercio electrónico ahora ofrecen diversas formas de recomendación, desde mostrar los artículos más populares hasta sugerir productos del mismo fabricante, utilizando técnicas sofisticadas de extracción de datos [2]. El Cuadro 1.1 muestra algunos ejemplos de aplicaciones actuales de los sistemas recomendadores.

Los sistemas recomendadores desempeñan un papel fundamental en el comercio en línea, ya que permiten satisfacer las necesidades de los clientes al optimizar los recursos de la empresa a través de la personalización del servicio. Esta personalización tiende a aumentar la satisfacción del cliente con el sitio web, lo que a su vez aumenta la probabilidad de que el cliente consuma productos y crea una base sólida y leal de clientes [3, 4].

Cuadro 1.1: Sitios populares que utilizan sistemas de recomendación

Página Web	**Objeto de recomendación**
Amazon	Productos
Netflix	Películas y series
Uber Eats	Comida
LinkedIn	Personas y empleos
Spotify	Canciones
YouTube	Videos

Como resultado de todo esto, los sistemas de recomendación se han convertido en el núcleo de muchas empresas que ofrecen productos o servicios a través de Internet.

Capítulo 2

Estado del Arte

Como se ha señalado previamente, los sistemas recomendadores surgen con el objetivo de mejorar la experiencia del usuario al proporcionar un servicio personalizado que beneficie tanto al cliente como a la empresa proveedora de servicios. Por lo tanto, un sistema recomendador eficiente puede resultar un factor diferencial frente a la competencia dentro de una misma industria.

2.1. Características de los Recomendadores

De acuerdo con [1], un sistema recomendador eficiente se puede caracterizar por alcanzar los siguientes objetivos fundamentales:

a. **Relevancia**: El modelo debe ser capaz de satisfacer las necesidades del usuario utilizando la menor cantidad de recursos disponibles en el menor tiempo posible.

   Se considera que el sistema recomendador es exitoso si un cliente ingresa a Amazon en busca de un producto y puede encontrarlo entre las primeras opciones del menú. Por el contrario, si el cliente tiene que buscar entre diferentes pestañas para encontrar lo que desea, el modelo ha fallado al no personalizar correctamente la experiencia del usuario.

b. **Novedad**: Otra característica deseable es ofrecer productos o servicios novedosos o desconocidos para el cliente, pero que sean de su agrado.

   Resulta fácil recomendar un *best seller* con muchas valoraciones positivas, pero el desafío radica en sugerir un libro desconocido para el usuario y que pueda interesarle. Además, las recomendaciones basadas únicamente en tendencias y gustos populares reducen la diversidad de opciones, lo cual es valorado negativamente.

c. **Sorpresa**: Es importante tener la capacidad de sorprender o innovar, es decir, el sistema debe ser capaz de establecer conexiones entre los gustos del usuario.

   Un ejemplo de esto sería alguien que disfruta de la comida exótica. El modelo reconoce que este cliente disfruta de la comida en restaurantes japoneses y peruanos, entre otros, pero no tiene información sobre los restaurantes de comida india. La sorpresa radica en que el sistema interprete que a este cliente le gusta la comida exótica y le recomiende con éxito un restaurante indio.

d. **Variedad**: Es imprescindible proporcionar al usuario una amplia gama de opciones para elegir, de acuerdo a sus preferencias. Cuanta más variedad de elección exista, más probable será que el usuario quede satisfecho.

2.2. Principales desafíos

A la hora de elaborar un sistema de recomendación existen varios desafíos que representan un peligro para el uso y rendimiento de los algoritmos. De acuerdo con [2], los principales retos son los siguientes:

1. **Falta de información**. En la actualidad, nos enfrentamos al problema de la falta de datos en los sistemas de recomendación. Con la amplia disponibilidad de elementos (por ejemplo, en un servicio de streaming con una gran variedad de series y películas), es común que haya poca o ninguna superposición entre los elementos evaluados por dos usuarios. Incluso cuando el promedio de evaluaciones por usuario-elemento es alto, estas evaluaciones se distribuyen de manera desigual entre los usuarios y los elementos, siguiendo generalmente una distribución de ley de potencias [5] o una distribución de Weibull [6]. Esto significa que la mayoría de los usuarios y elementos solo han recibido o expresado unas pocas calificaciones. Por lo tanto, para desarrollar algoritmos de recomendación efectivos, es crucial considerar esta ausencia de datos y sus implicaciones [7].

2. **Escalabilidad**. La escalabilidad es un reto importante en los sistemas de recomendación, especialmente en sitios con grandes volúmenes de datos de usuarios y elementos. Aunque estos datos suelen ser dispersos, es crucial considerar el costo computacional al desarrollar algoritmos de recomendación. Se busca encontrar algoritmos que sean eficientes en términos de tiempo y recursos, y que puedan ser fácilmente paralelizados. Otra estrategia es utilizar versiones incrementales de los algoritmos, donde las recomendaciones se ajustan de manera incremental a medida que llegan nuevos datos, en lugar de recalcularlas globalmente [8].

3. **Arranque en frío**. Cuando se incorporan nuevos usuarios al sistema, surge el desafío de la falta de información para generar recomendaciones personalizadas. Para abordar este problema, se utilizan técnicas de recomendación híbridas que combinan contenido y datos colaborativos [9]. Además, se puede recopilar información básica de los usuarios, como edad, ubicación y preferencias de género, para mejorar el proceso de recomendación. Otra estrategia consiste en identificar a los usuarios a través de servicios web, rastreando sus actividades en diferentes sitios de comercio electrónico, para ofrecer recomendaciones basadas en su historial.

4. **Diversidad vs. precisión**. En la tarea de recomendación, existe un dilema entre la precisión y la diversidad. Por lo general, recomendar elementos populares y altamente calificados es más efectivo para satisfacer los gustos de un usuario en particular. Sin embargo, este tipo de recomendación tiene poco valor, ya que los objetos populares son fácilmente accesibles sin la ayuda de un sistema de recomendación [2]. Por lo tanto, una lista de recomendaciones de calidad debe incluir elementos menos evidentes, que los usuarios difícilmente descubrirían por sí mismos [10]. Para abordar este problema, se emplean enfoques que buscan mejorar directamente la diversidad de las recomendaciones y se recurre a métodos de recomendación híbridos que combinan diferentes técnicas [11].

5. **Vulnerabilidad a ataques**. Debido a la importancia de los recomendadores en aplicaciones de comercio electrónico (ver ejemplos en el Cuadro 1.1), son objetivo de ataques maliciosos [12]. Para contrarrestar esta amenaza, existen diversas herramientas, como el bloqueo de evaluaciones maliciosas y el uso de técnicas avanzadas

de recomendación resistente [13]. Sin embargo, proteger los sistemas de recomendación contra ataques no es una tarea fácil, ya que los atacantes también desarrollan estrategias más sofisticadas a medida que evolucionan las medidas de prevención.

6. **Temporalidad**. Aunque los usuarios reales tienen intereses con escalas temporales muy diversas (por ejemplo, intereses a corto plazo relacionados a un viaje planificado e intereses a largo plazo relacionados con el lugar de residencia o las preferencias políticas),la mayoría de los algoritmos de recomendación no consideran las marcas de tiempo asociadas a las evaluaciones [2]. Actualmente, se está investigando cómo degradar adecuadamente el valor de las opiniones más antiguas y cómo identificar patrones temporales comunes en las evaluaciones de los usuarios y la relevancia de los elementos [14].

7. **Evaluación de recomendaciones**. Al evaluar las recomendaciones generadas por un algoritmo, existen diversas métricas disponibles (Capítulo 4). Sin embargo, es crucial seleccionar aquellas que se ajusten mejor a los objetivos deseados. Asimismo, comparar diferentes algoritmos de recomendación puede resultar problemático debido a que pueden abordar tareas distintas. Además, la experiencia y satisfacción del usuario desempeñan un papel fundamental para determinar la calidad y utilidad de las recomendaciones, aunque medir esto en una evaluación en tiempo real presenta desafíos [2].

8. **Interfaz de usuario**. Por último, la transparencia en las recomendaciones es fundamental para favorecer la aceptación por parte de los usuarios, quienes valoran comprender el motivo detrás de cada recomendación recibida [15]. Además, al enfrentar una extensa lista de elementos potencialmente interesantes, es esencial presentarla de manera clara y permitir una navegación sencilla para explorar las distintas recomendaciones generadas mediante enfoques diversos.

En definitiva, el desarrollo de sistemas de recomendación enfrenta varios desafíos que pueden afectar su uso y rendimiento. Entre los principales desafíos se encuentran la falta de información, la escalabilidad, el arranque en frío, el equilibrio entre la diversidad y la precisión, la vulnerabilidad a ataques, la temporalidad, la evaluación de las recomendaciones y la interfaz de usuario. Estos desafíos requieren enfoques y soluciones específicas para garantizar recomendaciones efectivas y satisfactorias. Además, es importante considerar la transparencia en las recomendaciones y proporcionar una interfaz de usuario intuitiva y fácil de usar.

2.3. Clasificación

En el diseño de los algoritmos de recomendación, el sistema utilizado para el seguimiento de las valoraciones desempeña un papel crucial. Estas valoraciones suelen ser expresadas en una escala que indica el nivel específico de agrado o desagrado hacia el artículo en cuestión.

Para brindar un producto o servicio personalizado, es fundamental contar con una amplia cantidad de datos que nos permitan comprender las características principales, gustos y preferencias de los usuarios, con el objetivo de clasificarlos de manera precisa. Este conocimiento detallado de las preferencias del cliente se obtiene a través de la recopilación y análisis de las valoraciones proporcionadas por los usuarios. De esta manera, al tener

acceso a un gran volumen de datos y utilizar un sistema de seguimiento de valoraciones, se logra comprender en profundidad las preferencias de los usuarios, lo que permite ofrecer recomendaciones y productos personalizados de manera más efectiva.

Existen dos enfoques para obtener estos datos y conocer las preferencias de los clientes: el enfoque explícito y el enfoque implícito [16].

- **Enfoque explícito**. En el enfoque explícito, se recopilan datos directamente proporcionados por los usuarios, como reseñas, calificaciones o encuestas que expresan sus preferencias de manera explícita. Por ejemplo, se utilizan sistemas de valoración de cinco estrellas o de "me gusta/no me gusta". Estos datos brindan información valiosa sobre las preferencias y gustos de los usuarios, ya que son expresados de manera consciente y deliberada. Sin embargo, este enfoque tiene la limitación de requerir la participación activa de los usuarios, algo que no todos están dispuestos a hacer. La falta de suficiente información puede resultar en recomendaciones simples y poco efectivas. Además, surge el desafío de los estándares de calificación, ya que diferentes usuarios pueden interpretar de manera diferente una misma puntuación.

- **Enfoque implícito**. Por otro lado, en el enfoque implícito se recopilan datos de forma indirecta, mediante la observación y análisis del comportamiento de los usuarios, interpretando sus acciones como indicios de interés o desinterés. Esto puede incluir el seguimiento de la navegación en un sitio web mediante clics, el registro de compras anteriores, la interacción en redes sociales y otras acciones que brindan información sobre las preferencias de los usuarios, aunque no sean expresadas de manera explícita. Sin embargo, este enfoque también presenta desafíos. Por ejemplo, para el caso de los clics, es necesario considerar la posibilidad de acciones accidentales por parte del usuario. Además, el hecho de que un cliente haya adquirido o consumido un producto no garantiza su satisfacción con el mismo.

En resumen, el acceso a un amplio volumen de datos utilizando tanto el enfoque explícito como el enfoque implícito nos permite obtener un conocimiento profundo de las características, gustos y preferencias de los usuarios. Esta información es fundamental para ofrecer productos y servicios personalizados de manera efectiva y satisfactoria.

A continuación, pasaremos a discutir las diferentes técnicas de recomendación existentes, las cuales se pueden clasificar según el tipo de información necesaria para categorizar a los usuarios. Algunos de los principales modelos de recomendación son el sistema basado en contenido, el filtrado colaborativo, el filtro basado en conocimientos y el modelo de recomendación híbrido.

2.3.1. Sistema basado en contenido

Los algoritmos basados en contenido son una clase de modelos de recomendación que utilizan información detallada sobre los elementos a recomendar. A diferencia de otros enfoques, no se requiere información de actividad pasada de los usuarios, ya que las recomendaciones se generan a partir de atributos comunes entre los productos disponibles, independientes de las preferencias de comportamiento de los usuarios [17]. Por ejemplo, en el caso de las películas, un atributo común podría ser el género asociado a ellas.

Una ventaja notable de estos algoritmos es que permiten recomendar fácilmente nuevos productos con características similares a los *best-sellers* o aquellos que tienen potencial para ser populares en el futuro. Esto es posible sin necesidad de un histórico de consumo, como en el caso de recomendar el nuevo libro de un autor reconocido a aquellos lectores que hayan leído previamente obras del mismo autor o les gusten los libros con temáticas similares.

La información utilizada por estos algoritmos se actualiza de forma continua a medida que se agregan nuevos atributos a los productos disponibles. Cuantos más atributos estén disponibles, más precisa y específica será la recomendación. Sin embargo, en situaciones en las que no se dispone de suficiente información, es decir, cuando no hay elementos con los que comparar, la personalización de la experiencia del usuario puede verse limitada [16].

Un inconveniente de estos modelos es que, en algunos casos, las recomendaciones pueden resultar obvias o pueden entrar en un bucle debido a la falta de variedad de artículos con atributos similares, hecho que puede afectar a la diversidad de las recomendaciones y la satisfacción del usuario [17].

Los sistemas de recomendación basados en contenido se tratan en la sección 5.1.

2.3.2. Filtrado colaborativo

Supone un sistema muy popular basado en los gustos de una comunidad para dar recomendaciones. La idea básica de los métodos de filtrado colaborativo consiste en generar recomendaciones utilizando la información de las interacciones pasadas entre usuarios y elementos. El objetivo es identificar a otros usuarios con gustos similares y utilizar sus opiniones y elecciones para hacer recomendaciones al usuario objetivo [1].

Por ejemplo, consideremos dos usuarios con gustos muy parecidos. Si las valoraciones que ambos han especificado son muy similares, entonces su similitud puede ser identificada por el algoritmo subyacente. En tales casos, es muy probable que las valoraciones en las que solo uno de ellos haya especificado un valor también sean similares. Esta similitud puede utilizarse para hacer inferencias sobre valores incompletamente especificados. La mayoría de los modelos de filtrado se centran en aprovechar las correlaciones entre elementos o entre usuarios para el proceso de predicción [1].

Existen dos tipos principales de filtrado colaborativo: el basado en usuarios y el basado en elementos.

a. **Filtro basado en usuarios**. Trata de relacionar usuarios a través de actitudes de compra similares [17]. Normalmente esta recomendación tiene lugar entre usuarios con historial de compra parecidos, y suele aparecer mediante frases del estilo “otros usuarios también compraron el producto x”.

b. **Filtro basado en elementos**. Consiste en relacionar productos con características comunes que puedan contener patrones de compra o consumo. También, se utilizan las valoraciones del propio usuario sobre estos artículos similares para realizar una predicción de valoración [1]. Es probable que estos métodos ofrezcan recomendaciones más relevantes, pero es menos probable que produzcan recomendaciones diversas.

El principal problema de estos algoritmos es que necesitan datos de actividad pasada para formalizar las recomendaciones [17]. Esto supone varios inconvenientes: el primero aparece en el momento de entrenar al modelo, ya que sin suficientes datos no se pueden encontrar patrones de comportamiento con facilidad. El siguiente surge cuando aparecen nuevos clientes de los que no se dispone de un historial de compra (arranque en frío descrito en la sección 2.2). Por último, está el caso de un nuevo producto cuando se pone a la venta y aún no ha sido valorado.

Los sistemas de recomendación de filtrado colaborativo se tratan en la sección 5.2.

2.3.3. Filtro basado en conocimientos

Un recomendador basado en conocimiento es un tipo de modelo de recomendación que utiliza información y reglas específicas del dominio para generar recomendaciones. A diferencia del filtrado colaborativo, que se basa en datos de interacciones pasadas entre usuarios y elementos, el recomendador basado en conocimiento se apoya en un conjunto de reglas y conocimientos previos para realizar sus recomendaciones, algo que implica tener un conocimiento detallado sobre los elementos a recomendar y las preferencias de los usuarios.

El sistema de recomendación utiliza este conocimiento para realizar inferencias y deducciones lógicas, determinando qué elementos podrían ser relevantes o de interés para un usuario en particular. En este tipo de algoritmos, cada usuario especifica sus necesidades mediante filtros, lo que permite al sistema generar recomendaciones personalizadas que se ajusten a los requisitos establecidos por el usuario [1].

Este modelo resulta especialmente útil para recomendar productos o servicios poco habituales, donde la disponibilidad de datos históricos es limitada [17]. Por ejemplo, en la compra de una casa, no es común contar con un registro de compras frecuente debido a la naturaleza infrecuente de esta transacción. En estos casos, se utiliza un modelo de recomendación basado en filtros, donde el usuario especifica criterios como el número de habitaciones, el presupuesto y la ubicación deseada. De esta manera, se generan recomendaciones que cumplen con los criterios establecidos, acotando el rango de búsqueda del cliente.

Un problema que puede presentar este tipo de algoritmos es que las recomendaciones pueden parecer obvias, ya que se muestran al usuario precisamente lo que ha especificado mediante los filtros. Sin embargo, en este contexto, la importancia radica en que el usuario busca algo concreto y específico [17].

2.3.4. Modelo de recomendación híbrido

Incluso para un algoritmo de recomendación eficiente, resulta desafiante satisfacer las diversas necesidades de sus usuarios heterogéneos [2]. Los modelos de recomendación híbridos superan esta problemática al combinar múltiples enfoques y técnicas de recomendación para proporcionar recomendaciones más precisas y efectivas [18, 19]. Estos modelos aprovechan las fortalezas de diferentes métodos de recomendación y los integran en un sistema unificado. Al combinar diversos enfoques, se pueden superar las limitaciones individuales de cada técnica y ofrecer recomendaciones personalizadas y adaptadas a las necesidades de los usuarios.

Una de las aplicaciones más significativas de los algoritmos de recomendación híbridos es resolver el desafío del arranque en frío (ver sección 2.2), mediante la combinación de datos colaborativos y de contenido. De esta manera, incluso un nuevo objeto que nunca ha sido valorado antes puede recibir recomendaciones [20]. De manera similar, un nuevo usuario que no ha realizado ninguna valoración previa puede recibir algunas recomendaciones. Además, dado que estos algoritmos combinan diferentes enfoques de recomendación, también tienen el potencial de mejorar la diversidad de las recomendaciones [11].

2.4. El problema de recomendar

En secciones anteriores, se ha subrayado la relevancia de contar con datos fiables y válidos para llevar a cabo una clasificación precisa de los clientes en distintos grupos en base a sus preferencias. Sin embargo, debido a la complejidad inherente a la recopilación y análisis de datos, esta tarea puede presentar desafíos significativos. En este sentido, se distinguen dos estrategias fundamentales para abordar de manera efectiva el problema de las recomendaciones, las cuales están determinadas por la interpretación y el uso adecuado de los datos disponibles.

2.4.1. Predicciones

Este primer enfoque consiste en predecir el valor de rating de una combinación usuario-artículo, en este caso, usuario-película. Se supone que se dispone de datos de entrenamiento que indican las preferencias de los usuarios por los artículos [1].

El problema se formula mediante una matriz conocida como *Utility Matrix*. Esta matriz consta de m filas que representan a los usuarios y n columnas que representan a los artículos. Cada celda de la matriz contiene la puntuación asignada por los usuarios a los respectivos productos (ver Figura 2.1). Durante el entrenamiento, se utilizan los valores específicos (observados) para ajustar el modelo, mientras que los valores ausentes (no observados) se predicen mediante este modelo de entrenamiento [1].

	I_1	I_2	I_3	I_4	I_5
	4			3	2
		2			
	1	4			
			5	2	

Figura 2.1: Matriz combinación usuario-película para $m = 4$ y $n = 5$

En la Figura 2.1 se muestra un ejemplo de una matriz de combinaciones usuario-película, donde se tienen cuatro usuarios y cinco películas (*items*). Los valores en las celdas representan las calificaciones dadas por los usuarios a las respectivas películas. Por ejemplo, el usuario uno ha dado una calificación de 4/5 a la primera película. Las celdas sin valor indican combinaciones no observadas.

Es importante tener en cuenta que en la matriz de recomendación, muchas celdas no tendrán valores debido a que los usuarios no suelen consumir y calificar todos los elementos disponibles. Por lo tanto, el objetivo de este método es predecir los valores desconocidos, estimando la actitud que un usuario podría tener hacia esos elementos no calificados [17].

La evaluación de un recomendador de este tipo se basa en su capacidad para hacer predicciones precisas y proporcionar recomendaciones acertadas.

2.4.2. Ranking

En la práctica, no es necesario realizar predicciones de las valoraciones de los usuarios para artículos específicos con el objetivo de proporcionar recomendaciones. Resulta más común recomendar los k mejores artículos para un usuario en particular, o determinar los k mejores usuarios a los que dirigirse para un artículo específico. Este problema se conoce como el problema de recomendación top-k, y se basa en la formulación de clasificación del problema de recomendación. Aunque la determinación de los artículos top-k es más frecuente, los métodos utilizados en ambos casos son análogos [1].

El problema de recomendación top-k se puede describir de manera intuitiva como la selección de k elementos de un conjunto n dado. Estos k elementos son aquellos que el algoritmo predice como los más interesantes para un usuario determinado, donde k representa el número de elementos seleccionados.

Un ejemplo de esto se encuentra en la plataforma web de Airbnb[1]. A partir de unos filtros determinados por el usuario como el precio deseado, número de camas y localización, se muestran el top k de elementos que correspondan con lo que el usuario desea encontrar (ver Figura 2.2).

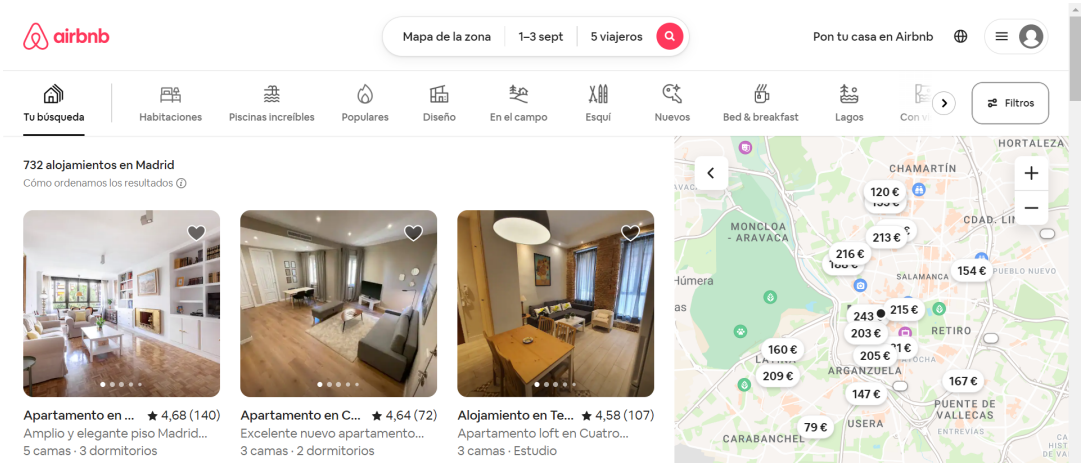


Figura 2.2: Aplicación del recomendador de Airbnb

[1] Airbnb: `https://www.airbnb.es/`

Otro ejemplo es el algoritmo de recomendación de Zara[2]. Cuando un usuario realiza una búsqueda de un producto obtiene un menú con una serie de elementos que el algortimo entiende que pueden resultar interesantes para el usuario (ver Figura 2.3).

Figura 2.3: Recomendador de Zara

Según [16], existen dos procedimientos para abordar este tipo de algoritmos. Las técnicas posibles son las siguientes:

a. **Considerar únicamente los productos asociados a los gustos del usuario**. En este enfoque, se seleccionan los productos que se ajustan a los gustos del usuario y se presentan ordenados según las predicciones de las puntuaciones que el usuario daría a esos productos. Estas predicciones son invisibles para el usuario, ya que se considera que no son relevantes. Sin embargo, las opiniones y valoraciones de otros usuarios sobre las recomendaciones pueden resultar interesantes. Además, se filtran los contenidos según las posibles preferencias del usuario o restricciones específicas (por ejemplo, la edad), eliminando películas que el usuario ya haya visto o que puedan resultar ofensivas.

b. **Realizar una predicción de la puntuación de todos los productos disponibles en el catálogo**. Consiste en una predicción de la puntuación que el usuario daría a todos los productos del catálogo y luego seleccionar los N mejores valorados según el modelo. Sin embargo, esta estrategia no es eficiente, ya que no aprovecha todos los recursos disponibles y supone un coste computacional significativo cuando el catálogo contiene un gran número de productos.

[2] Zara: `https://www.zara.com/es/`

Capítulo 3

Definición del Trabajo

3.1. Motivación

Los sistemas recomendadores han ganado un prominente interés en la actualidad. Cada vez más negocios optan por digitalizarse y aventurarse en el comercio en línea, invirtiendo en sistemas de recomendación [2]. Estos sistemas se encuentran presentes en una amplia variedad de industrias, como el comercio minorista, la música, el contenido multimedia, las búsquedas en la web, los productos y la comida, entre otros (ver ejemplos en el Cuadro 1.1), con el objetivo de mejorar la experiencia de los usuarios.

Un ejemplo notable es Netflix[1], que personaliza el menú principal de su plataforma en función del usuario que inicia sesión [21], ofreciendo contenido de películas y series de televisión acorde a sus preferencias (Figura 3.1). Incluso las imágenes descriptivas de los productos pueden variar en función del usuario (Figura 3.2).

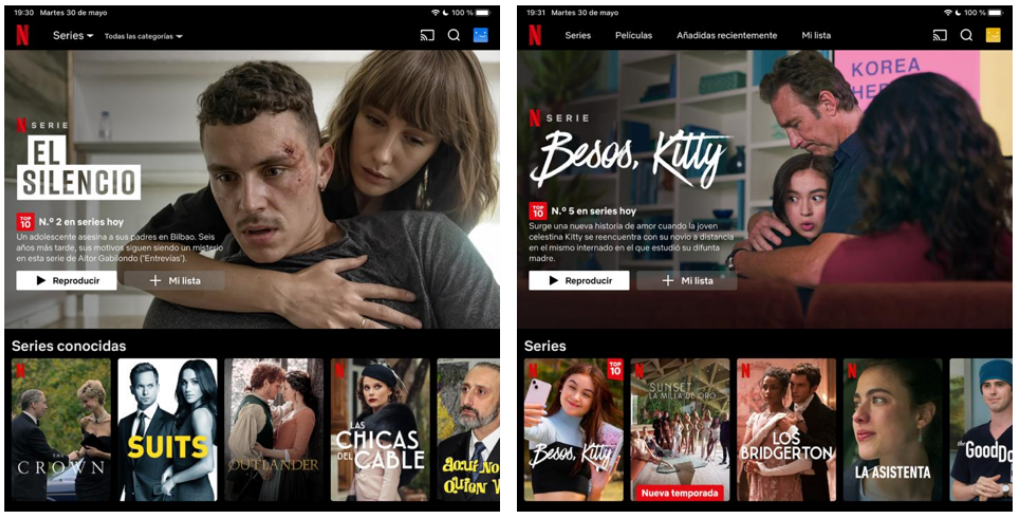


Figura 3.1: Ejemplo de adaptación del menú principal en función del usuario

[1]Netflix: `https://www.netflix.com`

La Figura 3.1 presenta una comparación entre dos menús de Netflix que varían en función del usuario. En la imagen de la izquierda, se muestra el menú principal para el usuario A, mientras que en la imagen de la derecha se muestra el menú principal para el usuario B. Es evidente que ambos menús son distintos, ya que se presentan sugerencias personalizadas y adaptadas a los gustos y preferencias de cada usuario.

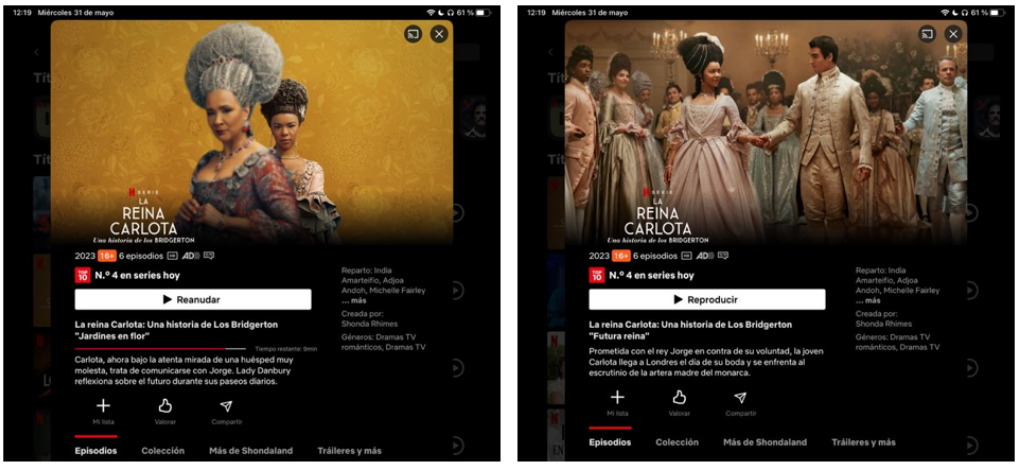


Figura 3.2: Ejemplo de adaptación de las imágenes en función del usuario

La Figura 3.2 muestra cómo la plataforma adapta la imagen de portada de la serie *La Reina Carlota en función del usuario*. En la imagen de la izquierda, diseñada específicamente para el usuario A, se presentan los personajes principales, sin hacer alusión a ninguna escena concreta de la serie. Por otro lado, en la imagen de la derecha, creada para el usuario B, se enfatiza el romance y los aspectos históricos con una imagen más elegante y romántica. Esta adaptación personalizada de las imágenes de portada refleja cómo Netflix busca captar la atención y satisfacer los intereses de cada usuario, ofreciendo una experiencia única y atractiva para cada uno.

Además, Netflix permite a los usuarios evaluar los productos de su catálogo mediante un sistema de clasificación de 5 estrellas. Esta información se utiliza para comprender mejor el perfil del cliente y poder ofrecer recomendaciones más precisas basadas en sus preferencias y visualizaciones anteriores. Los datos de los perfiles también se almacenan para analizar patrones de comportamiento.

El objetivo de este trabajo es abordar el desafío de la recomendación, centrándose especialmente en el ámbito del contenido multimedia, en particular, las películas. Se busca analizar las diversas técnicas existentes con el fin de encontrar aquellas que brinden los mejores resultados, es decir, sean capaces de generar sugerencias que satisfagan a los clientes.

3.2. Objetivos del proyecto

Este proyecto representa un análisis exhaustivo de los sistemas de recomendación. A lo largo de este estudio se examinarán meticulosamente las diversas técnicas y procedimientos existentes, con el objetivo principal de desarrollar un modelo desde cero capaz de proporcionar recomendaciones de películas basadas en las preferencias de los usuarios.

Además, otro de los objetivos fundamentales de este trabajo consiste en abordar los desafíos inherentes a los modelos de recomendación existentes y proponer soluciones para superarlos. Se presta especial atención a la problemática de la escasez de información, tanto en lo que respecta a los usuarios nuevos como a las películas recién estrenadas que carecen de valoraciones. Por otra parte, se llevará a cabo un estudio exhaustivo e implementación de diversas reglas de asociación de contenido con el fin de llevar la personalización a un nivel superior.

Por último, se tiene la intención de desarrollar una plataforma que permita verificar el correcto funcionamiento del sistema recomendador a través de la creación y evaluación de nuevos perfiles de usuarios.

3.3. Metodología

En esta sección, se presenta el enfoque principal del trabajo, que se centra en la creación y evaluación de un modelo de recomendación personalizado para nuevos usuarios. Se emplearán diversas técnicas de recomendación, como el filtrado colaborativo y el basado en contenido, con el objetivo de proporcionar recomendaciones precisas y relevantes. Además, se describirán en detalle los distintos modelos y procedimientos desarrollados en el proyecto, así como las diferentes fases necesarias para garantizar un desarrollo efectivo del mismo.

En primer lugar, es importante destacar que el desarrollo técnico del proyecto, específicamente la implementación de diversos algoritmos de recomendación, se llevará a cabo utilizando el lenguaje de programación Python[2]. La elección de este lenguaje se basa en su reconocida combinación de facilidad de uso, disponibilidad de una amplia gama de bibliotecas especializadas, versatilidad, una comunidad activa y su capacidad de integración con otras tecnologías. Estas características hacen de Python un lenguaje popular y apropiado para el desarrollo de algoritmos de recomendación.

Dicho esto, el proyecto comienza con un período de familiarización y aprendizaje de los conceptos fundamentales relacionados con los sistemas de recomendación, las diferentes técnicas existentes, su implementación y sus aplicaciones. Posteriormente, se desarrollan una serie de algoritmos de recomendación de dificultad baja y media (por ejemplo, sistema basado en contenido) utilizando el conjunto de datos de MovieLens[3], una base de datos popular y completa sobre películas y recomendaciones de usuarios, utilizada para analizar patrones de preferencia y desarrollar algoritmos de recomendación personalizada. Este enfoque permite adquirir experiencia en el desarrollo, entrenamiento y evaluación de sistemas de recomendación.

[2]Python: `https://www.python.org`
[3]MovieLens: `https://movielens.org/`

Es importante destacar que para el procesamiento y manejo eficiente de los datos utilizados en el proyecto se ha creado la clase MovieLens. Esta clase proporciona funcionalidades esenciales relacionadas con la carga de datos, la obtención de información detallada sobre películas y la generación de recomendaciones basadas en las preferencias de los usuarios. Al utilizar la clase MovieLens, se accede a una interfaz eficiente que facilita la obtención de información relevante, como detalles de películas y clasificaciones de usuarios, y permite generar recomendaciones personalizadas basadas en las preferencias individuales de cada usuario.

Una vez que se han adquirido los conocimientos necesarios sobre los recomendadores, se procede a desarrollar modelos más complejos empleando técnicas de *Deep Learning*. Cabe destacar que para la elaboración de las diferentes técnicas de recomendación se toma como referencia el curso *Building Recommender Systems with Machine Learning and AI* [16].

Tras la finalización del desarrollo de todos los modelos, se procede a evaluar su rendimiento con el objetivo de analizar las ventajas e inconvenientes de cada una de las técnicas implementadas. Este análisis exhaustivo permite obtener una visión completa del rendimiento de los diferentes recomendadores.

Finalmente, se lleva a cabo el desarrollo de una aplicación que permita evaluar de manera práctica y dinámica el rendimiento del recomendador con las mejores recomendaciones en nuevos usuarios. Esta aplicación proporcionará una experiencia interactiva que facilitará la evaluación y comprensión de las capacidades del sistema de recomendación implementado.

Todo el código desarrollado durante el proyecto se encuentra en el siguiente repositorio de GitHub : https://github.com/inesgfortis/RecommenderSys

3.4. Planificación

La planificación implementada en el desarrollo del proyecto se ha estructurado de la siguiente manera (Figura 3.3):

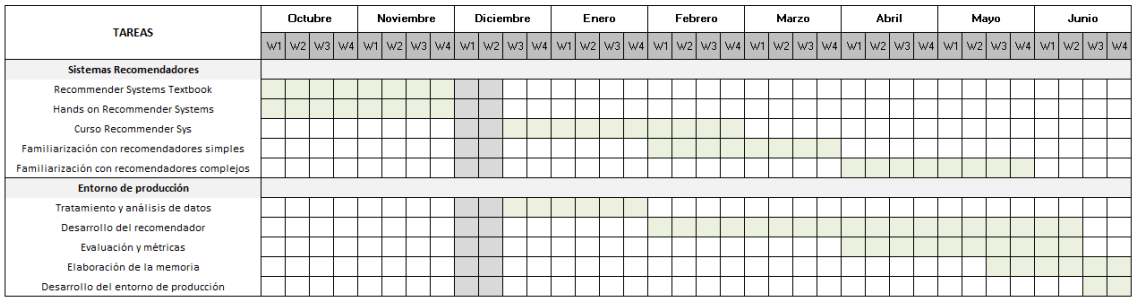


Figura 3.3: Planificación del proyecto

Capítulo 4

Métricas de evaluación

Los sistemas de recomendación comparten varias similitudes conceptuales con el problema de los modelos de clasificación y regresión. En este tipo de problemas se trata de predecir la variable *target* a partir una serie de variables características. En el caso de los sistemas recomendadores se busca predecir las entradas no observadas a partir de información disponible en la matriz de objetos, de forma que el problema de recomendar puede considerarse una generalización del problema de la clasificación [1].

En el contexto específico de este proyecto, el objetivo sería recomendar a un usuario objetivo i aquellas películas que mejor se ajusten a sus preferencias. Esto implica que el sistema de recomendación ordenará todas las películas no consumidos por el usuario i y recomendará aquellas que mejor se adapanten a sus preferencias y necesidades. Las recomendaciones dependerán del tipo de algortimo empleado.

Con el fin de evaluar los algoritmos de recomendación, la base de datos se divide en tres conjuntos de datos: 1. el conjunto de entrenamiento (*train*), 2. el conjunto de prueba (*test*), y 3. el *anti-testset*. Para ello se utiliza la librería Surprise [22].

El conjunto de train-test es una división de los datos utilizada para entrenar y evaluar modelos de recomendación. Se trata de separar el conjunto de datos original en dos conjuntos distintos.

El conjunto de train contiene una parte de los datos originales y generalmente es la porción más grande del conjunto. Su propósito es entrenar el modelo de recomendación mediante la identificación de patrones y relaciones en los datos, lo que permite generar recomendaciones personalizadas.

El conjunto de test está compuesto por datos que no se utilizaron durante el entrenamiento. Este conjunto se emplea para simular situaciones reales en las que el modelo debe realizar recomendaciones basadas en datos desconocidos. Su principal función es evaluar el rendimiento del modelo entrenado y medir su capacidad para hacer recomendaciones precisas y relevantes en escenarios no vistos previamente.

Por otro lado, el conjunto anti-testset se emplea en la evaluación de sistemas de recomendación con el propósito de evitar la selección aleatoria de pares usuario-ítem que puedan introducir sesgos en los resultados de la evaluación.

En contraposición a la selección aleatoria de usuarios y elementos para el conjunto de prueba, un antitest set se selecciona de manera rigurosa y sistemática, con el fin de asegurar que no se incluyan datos que puedan inducir a una evaluación engañosa. Esta selección cuidadosa busca excluir, por ejemplo, aquellos usuarios y elementos con un número limitado de calificaciones o con una correlación atípica respecto a otros usuarios o elementos en el conjunto de datos.

La utilización de un antitest set en la evaluación de sistemas de recomendación se fundamenta en la necesidad de garantizar la objetividad y la validez de los resultados. Al excluir de manera deliberada ciertos usuarios y elementos, se busca evitar la introducción de sesgos que puedan distorsionar la efectividad y la fiabilidad de los algoritmos evaluados.

En esta sección examinaremos algunas de las principales técnicas *offline* para la evaluación de los sistemas de recomendación. Es importante tener en cuenta que la elección de métricas específicas para evaluar la eficacia y el rendimiento de las recomendaciones depende de los objetivos del sistema en cuestión. Cada métrica se selecciona con base en los criterios que deben cumplir las recomendaciones para satisfacer las necesidades y preferencias de los usuarios.

Sin embargo, vale la pena destacar que, en última instancia, la evaluación final de cualquier sistema de recomendación se basa en el juicio de sus usuarios. Aunque las métricas proporcionan una medida objetiva, la experiencia y la satisfacción del usuario desempeñan un papel fundamental en determinar la calidad y la utilidad de las recomendaciones.

A continuación, se describen las métricas escogidas en este proyecto para medir la calidad de las recomendaciones de los diferentes algoritmos que se van a desarrollar de acuerdo con las características propias de un buen recomendador descritas en la sección 2.1.

En el Apéndice B se muestra la implementación de las métricas en Python.

4.1. Métricas de precisión

En el campo de los sistemas recomendadores, es necesario considerar métricas de evaluación que permitan medir la precisión y el desempeño de los modelos. Estas métricas juegan un papel fundamental a la hora de evaluar la calidad de las recomendaciones generadas y proporcionan información cuantitativa sobre la efectividad de los algoritmos utilizados.

Las métricas de precisión se enfocan en medir la exactitud de las recomendaciones en comparación con los datos reales, lo que ayuda a determinar la capacidad del sistema para predecir los gustos y preferencias de los usuarios de manera precisa.

Dos de las métricas más comúnmente utilizadas en la evaluación de sistemas recomendadores son el Error Absoluto Medio (MAE) y el Error Cuadrático Medio (RMSE). Estas métricas miden la discrepancia entre las valoraciones predichas por el sistema y las valoraciones reales proporcionadas por los usuarios.

El MAE calcula el promedio de las diferencias absolutas entre las valoraciones predichas y las valoraciones reales. Proporciona una medida directa del error promedio, sin tener en cuenta la dirección de las diferencias. La fórmula para el MAE es la siguiente:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|r_i - \hat{r}_i| \tag{4.1}$$

Por otro lado, el RMSE es la raíz cuadrada del promedio de las diferencias al cuadrado entre las valoraciones predichas y las valoraciones reales. Esta métrica penaliza de manera más significativa los errores más grandes, ya que los eleva al cuadrado antes de calcular la media. Se define como

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2} \tag{4.2}$$

Ambas métricas son ampliamente utilizadas en la evaluación de sistemas recomendadores debido a su simplicidad y capacidad para proporcionar una medida objetiva del desempeño. Sin embargo, es importante considerar que existen otras métricas y enfoques que pueden adaptarse mejor a diferentes contextos y necesidades específicas, dado que en el context de las recomendaciones no aporta valor lograr una precisión excepcional en la predicción de una película valorada con un 1 si esta no se recomendaría en ningún caso. Del mismo modo, la diferencia entre predecir una película con una valoración de 5 como 4.8 o 4.3 no es significativa si la decisión final acaba siendo la misma. Es evidente que contar con una precisión sólida en las predicciones de las valoraciones es relevante, pero en última instancia, lo fundamental es que las recomendaciones ofrecidas agreguen valor.

Con este objetivo en mente, es primordial identificar las N mejores recomendaciones del conjunto de prueba en evaluación y verificar cuáles de ellas resultan verdaderamente relevantes. Este proceso permite evaluar la capacidad del modelo de recomendación para sugerir elementos que satisfagan las necesidades y preferencias del usuario, y, en última instancia, brinden un valor significativo. Para ello se emplea el conjunto *anti-testset* y se examinan otro tipo de métricas.

4.2. Métricas de relevancia

Las métricas de relevancia desempeñan un papel crucial en la evaluación de la calidad de las recomendaciones generadas, dado que se centran en medir la capacidad del sistema para proporcionar recomendaciones relevantes y útiles a los usuarios.

Dos métricas ampliamente utilizadas en este contexto son *Precision* y *Recall* dado un número de recomendaciones k. Estas métricas permiten evaluar el equilibrio entre la precisión de las recomendaciones y la capacidad del sistema para recuperar ítems relevantes.

La precisión se refiere a la proporción de ítems recomendados que son relevantes para el usuario. Es decir, mide la exactitud de las recomendaciones al indicar cuántas de las sugerencias realizadas son realmente útiles para el usuario. Una alta precisión implica que las recomendaciones proporcionadas son altamente relevantes y se ajustan a los intereses del usuario.

Por otro lado, el recall se refiere a la proporción de ítems relevantes que son efectivamente recomendados por el sistema. Mide la capacidad del sistema para recuperar ítems importantes y evitar perder recomendaciones valiosas. Un alto recall indica que el sistema es capaz de identificar y presentar ítems relevantes en una mayor medida.

Para definir las métricas mencionadas, es necesario considerar los siguientes aspectos:

- Se requiere establecer un umbral mínimo para diferenciar entre lo que se considera como positivo y lo que no lo es.
- Denominaremos una valoración de un usuario como **Relevante** si ha sido puntuada igual o por encima del umbral establecido.
- Una recomendación se considerará **Valorada** dentro del conjunto de las mejores K recomendaciones si la predicción del modelo es mayor o igual al umbral definido.

Bajo estas condiciones, se definen las métricas de Precisión@K (ecuación 4.3) y Recall@K (ecuación 4.4) de la siguiente manera:

$$\text{Precision@K} = \frac{|S(K) \cap G|}{|S(K)|} \tag{4.3}$$

Donde:

- Precision@K es la métrica de precisión en el nivel de corte K de la lista recomendada.
- $|S(K) \cap G|$ representa el tamaño del conjunto de elementos relevantes en la lista recomendada de tamaño K que también están en el conjunto verdadero de elementos relevantes G.
- $|S(K)|$ es el tamaño de la lista recomendada de tamaño K.

La ecuación 4.3 mide la proporción de elementos relevantes entre las recomendaciones realizadas en comparación con el número total de elementos recomendados. Esto brinda una indicación de la calidad y relevancia de las recomendaciones, ya que cuanto mayor sea la proporción de elementos relevantes en la lista recomendada, mayor será la precisión.

$$\text{Recall@K} = \frac{|S(K) \cap G|}{|G|} \tag{4.4}$$

Donde $|G|$ es el tamaño del conjunto verdadero de elementos relevantes.

La ecuación 4.4 mide la proporción de elementos relevantes que se recuperan correctamente de la lista recomendada en relación con el conjunto verdadero de elementos relevantes. Esta métrica proporciona información sobre qué tan efectivamente se han recuperado los elementos relevantes. Un recall más alto indica que se recuperaron más elementos relevantes de manera adecuada en comparación con el tamaño total del conjunto verdadero.

Es importante destacar que estas métricas devuelven un valor de precisión y recall para cada usuario en el conjunto de test y en cada modelo evaluado. Para obtener un valor agregado, se definen las siguientes métricas:

- **Mean Average Precision at K (MAP@K)**: corresponde con la media de todas las precisiones obtenidas en el paso anterior.
- **Mean Average Recall at K (MAR@K)**: equivale a la media de todos los recalls obtenidos en el paso anterior.

Estos valores pueden compararse con la precisión y el recall resultantes de la recomendación aleatoria, lo que da lugar a mejoras de precisión y como se definen en [11].

Estas métricas son complementarias y brindan una visión más completa de la calidad de las recomendaciones. No obstante, es importante considerar que la precisión y el recall están inversamente relacionados, lo que implica que mejorar uno puede afectar al otro. Por lo tanto, es fundamental encontrar un equilibrio adecuado según las necesidades y preferencias de los usuarios.

Otro factor relevante en los sistemas recomendadores es la posición de las recomendaciones, ya que se busca resaltar las mejores sugerencias de los usuarios en las primeras posiciones. Para medir esto, se utiliza la métrica de Ganancia Acumulada (*Cumulative Gain* o CG) de las recomendaciones. Esta medida evalúa la utilidad acumulada de las recomendaciones en función de las calificaciones de los elementos y su posición en la lista recomendada . Al sumar las calificaciones reales de las recomendaciones anteriores, se obtiene un valor acumulativo que refleja la calidad y efectividad de las recomendaciones [23]. En otras palabras, la CG considera tanto la relevancia de los elementos recomendados como el efecto de la posición relativa en la utilidad acumulada. Una puntuación más alta de CG indica que las recomendaciones han logrado proporcionar una mayor utilidad en las primeras posiciones de la lista recomendada, lo cual es deseable en sistemas recomendadores que buscan ofrecer las opciones más relevantes y atractivas a los usuarios. Matemáticamente, se expresa de la siguiente manera:

$$\text{CG@K} = \sum_{i=1}^{k} \text{Rel}(i) \times \text{Weight}(i) \tag{4.5}$$

Donde:

- CG@K representa el Cumulative Gain en el nivel de corte K de la lista recomendada.
- Rel(i) es la relevancia del elemento recomendado en la posición i. Corresponde con el rating asignado por el usuario al elemento recomendado en la posición i.
- Weight(i) es el peso o factor de ponderación asignado al elemento recomendado en la posición i.

Sin embargo, esta métrica no considera el orden de las recomendaciones. Para abordar este aspecto, se introduce el concepto de Ganancia Acumulada Descontada (*Discounted Cumulative Gain* o DCG), que divide cada calificación por un factor que penaliza de manera más significativa a medida que aumenta la posición k [23]. La fórmula del DCG es la siguiente:

$$\text{DCG@K} = \sum_{i=1}^{k} \frac{\text{Rel}(i)}{\log_2(i+1)} \tag{4.6}$$

En esta métrica, se busca que el modelo asigne las calificaciones reales más altas a las primeras posiciones, por lo que cada calificación se divide por un factor creciente. Sin embargo, dado que el DCG no está acotado, es necesario compararlo con un valor de referencia. Para esto, se utiliza el DCG ideal (*Ideal Discounted Cumulative Gain* o IDCG), que representa la mejor ordenación posible de las calificaciones. La fórmula del iDCG es la siguiente:

$$\text{IDCG@K} = \sum_{i=1}^{k} \frac{\text{Rel}_{\text{ideal}}(i)}{\log_2(i+1)} \tag{4.7}$$

Donde $\text{Rel}_{\text{ideal}}(i)$ corresponde con la relevancia ideal del elemento recomendado en la posición i y se refiere a la mejor ordenación posible de los elementos según los ratings de los usuarios.

La relación entre el DCG y el iDCG se utiliza para determinar qué tan cerca se encuentra el DCG del mejor orden posible. Esta relación se conoce como Ganancia Acumulada Descontada Normalizada (*Normalized Discounted Cumulative Gain* o NDCG), y se calcula dividiendo el DCG entre el iDCG:

$$\text{NDCG@K} = \frac{\text{DCG}}{\text{IDCG}} \tag{4.8}$$

El valor resultante del NDCG está en el rango de 0 a 1, donde 1 indica que el modelo ha ordenado las recomendaciones de manera óptima. Es importante mencionar que existen variantes más agresivas para penalizar los errores de posición en las fórmulas mencionadas. Además, a medida que aumenta el valor de k, se penaliza en mayor medida.

Por otro lado, es fundamental tener en cuenta que el NDCG no mide la calidad de las recomendaciones en sí, sino que evalúa la capacidad del modelo para ordenar las recomendaciones correctamente. Una vez que las recomendaciones están establecidas, esta métrica proporciona una medida de qué tan bien se han ordenado, siendo 1 el valor máximo posible.

Otro aspecto relevante es que los valores resultantes del NDCG suelen ser cercanos a 1, debido a la escala de calificación utilizada en el dataset empleado *MovieLens* (0 a 5) y al hecho de que las recomendaciones son relativamente buenas. Sin embargo, si se empleara una escala de acierto/error con valores binarios $0, 1$, las diferencias en la ordenación serían más notables.

4.3. Otras métricas de interés

Incluso un objeto relevante recomendado con éxito tiene poco valor para un usuario cuando es conocido por la mayoría de personas. A fin de complementar las métricas de precisión y relevancia presentadas anteriormente, es crucial considerar la importancia de la diversidad y la novedad en las recomendaciones [10, 11, 24]. Estas métricas de búsqueda ofrecen una perspectiva adicional para evaluar la calidad y el impacto de los sistemas recomendadores.

Incluso cuando se logra una alta precisión en un sistema de recomendación, con frecuencia existe la posibilidad de que no se puedan recomendar ciertos elementos a una proporción determinada de usuarios, o que no se pueda recomendar nunca una cierta proporción de elementos. Esta limitación se conoce como cobertura, *Coverage* [1].

La cobertura mide la capacidad de un sistema recomendador para abarcar un amplio rango de elementos o productos en sus recomendaciones. Se enfoca en garantizar que no se descuiden opciones relevantes y valiosas durante el proceso de recomendación.Una cobertura baja indica que el algoritmo sólo puede acceder y recomendar un número reducido de objetos distintos (normalmente los más populares), lo que suele traducirse en recomendaciones poco variadas. Por el contrario, los algoritmos con una cobertura alta tienen más probabilidades de ofrecer recomendaciones diversas [25].

En términos matemáticos, la fórmula de la métrica de cobertura se expresa de la siguiente manera:

$$\text{Coverage} = \frac{\text{Número de elementos recomendados}}{\text{Número total de elementos en el conjunto de datos}} \tag{4.9}$$

Una variante de la cobertura, es la cobertura por usuario o *User Coverage*. Este valor indica el porcentaje de usuarios que han recibido, al menos, una recomendación donde el modelo entiende que es buena, es decir, una recomendación con una puntuación superior a un *threshold* dado.

La cobertura de los sistemas de recomendación se ve afectada por la dispersión de las matrices de valoración. Por ejemplo, si una matriz de valoración contiene una única entrada por cada fila y cada columna, prácticamente ningún algoritmo puede generar recomendaciones significativas. Sin embargo, diferentes sistemas de recomendación exhiben niveles variables de propensión para ofrecer cobertura. En la práctica, los sistemas a menudo logran una cobertura del 100 % mediante la utilización de valores predeterminados para las valoraciones que no pueden ser predichas. Estos valores por defecto suelen ser la media de todas las valoraciones de un usuario para un elemento cuando no se puede predecir la valoración para una combinación específica de usuario y elemento [1].

Por lo tanto, es fundamental considerar siempre el equilibrio entre precisión y cobertura en el proceso de evaluación de los sistemas de recomendación, dado que se espera que un buen método de recomendación tenga una precisión y una cobertura elevadas [2].

Por último, vamos a definir una métrica adicional que permite medir la popularidad de las recomendaciones en un sistema recomendador. Esta métrica se conoce como *Novelty* (novedad). Para ello, es necesario establecer el concepto de popularidad dentro del conjunto de películas consideradas. En este contexto, la popularidad de una película se define en función del número de valoraciones que ha recibido.

La métrica **Novelty** se define como el valor promedio del ranking de las recomendaciones ofrecidas a un usuario, expresado de la siguiente manera:

$$\text{Novelty} = \frac{\sum_{u \in U} \sum_{i \in R_u} Rank(i)}{\sum_{u \in U} |R_u|} \tag{4.10}$$

Donde:

- U representa el conjunto de usuarios a los que se les ha realizado al menos una recomendación.
- R_u denota el conjunto de recomendaciones para el usuario u.
- $|R_u|$ indica el número total de recomendaciones realizadas al usuario u.
- $Rank(i)$ representa el ranking de la película i, que varía desde 1 hasta el total de películas en el conjunto considerado.

La novedad se centra en evaluar la capacidad del sistema recomendador para presentar sugerencias sorprendentes y diferentes al usuario. Su objetivo es medir en qué medida las recomendaciones incluyen elementos poco comunes o desconocidos para el usuario. La novedad es especialmente relevante para evitar la monotonía y el aburrimiento, ya que ofrece la oportunidad de descubrir nuevas opciones y ampliar los horizontes del usuario.

Estas métricas de diversidad y novedad proporcionan una visión más completa de la calidad de las recomendaciones, considerando no solo la relevancia y precisión, sino también la capacidad del sistema para presentar opciones diversas y novedosas. Al incorporar estas métricas en la evaluación de los sistemas recomendadores, se promueve una experiencia más enriquecedora y personalizada para los usuarios, fomentando la exploración y el descubrimiento de nuevos contenidos y productos.

Capítulo 5

Técnicas de recomendación

Hasta ahora, se ha explorado la importancia de los sistemas de recomendación en la actualidad y cómo pueden mejorar la experiencia del usuario al proporcionar sugerencias personalizadas. Sin embargo, es crucial comprender que no hay un enfoque único para generar recomendaciones sino que existen diferentes algoritmos y técnicas que se utilizan para este propósito.

En esta sección, exploraremos una selección de los algoritmos de recomendación más populares y ampliamente utilizados en la actualidad, pero también abordaremos la dificultad progresiva que cada uno de ellos implica. Cada algoritmo presenta sus propias características y complejidades, lo que los hace apropiados para diferentes escenarios y dominios de aplicación. No solo nos limitaremos a describir los algoritmos más frecuentes, sino que también nos adentraremos en los desafíos y retos que cada uno plantea.

A medida que avanzamos en este capítulo, nos sumergiremos en un análisis detallado de cada algoritmo, explorando en profundidad su funcionamiento, los datos que requieren y las ventajas y dificultades que conllevan. El objetivo es proporcionar una visión completa de las principales técnicas de recomendación existentes, junto con una comprensión sólida de su aplicabilidad y potencial. Además, destacaremos algunos ejemplos y casos de uso relevantes para cada algoritmo, con el fin de ilustrar cómo se aplican en situaciones reales y cómo pueden generar recomendaciones útiles y precisas, a pesar de los desafíos inherentes a cada método.

5.1. Sistema basado en contenido

En los sistemas de recomendación basados en contenido, las recomendaciones se generan utilizando los atributos descriptivos de los artículos. En este contexto, el término **contenido** se refiere a las características y descripciones asociadas a cada artículo, que son utilizadas para establecer la similitud entre ellos. El objetivo de estos algoritmos es ofrecer recomendaciones personalizadas cuando no se dispone de valoraciones de usuarios, evitando así el uso del filtrado colaborativo.

Para ello, se emplean las descripciones de los artículos como datos de entrenamiento para crear un modelo de clasificación o regresión específico para cada usuario. Estos modelos se utilizan para predecir las preferencias del usuario en función de las características de los artículos, sin depender de las valoraciones o comportamientos de compra [1].

Es importante destacar que este proyecto se ha centrado exclusivamente en los atributos asociados a las películas, omitiendo deliberadamente las valoraciones de los usuarios. Esta elección nos permite evaluar la efectividad de un algoritmo que relaciona las películas únicamente en función de sus características. Por ejemplo, podemos medir la precisión de las recomendaciones para películas de género Acción sin considerar la opinión de los usuarios sobre esas películas.

Al desarrollar algoritmos de recomendación basados en contenido, es fundamental realizar un procesamiento previo de los datos para extraer de manera adecuada las características más informativas. En nuestro caso, al analizar los datos de MovieLens, identificamos dos atributos clave: el año de producción y los géneros asociados a las películas. Sin embargo, esta información no estaba directamente disponible en la base de datos, por lo que ha sido necesario llevar a cabo un procesamiento del conjunto de datos de MovieLens con el objetivo de preparar las variables en un formato adecuado y útil para nuestro sistema de recomendación.

En primer lugar se ha realizado una transformación en la columna *genres*, que originalmente contiene géneros presentados en forma de cadena de caracteres. Mediante el uso de la función *split*, se divide cada cadena de géneros en una lista de elementos, lo que permite representar de manera independiente cada uno de los géneros asociados a cada película. Posteriormente, se realiza la extracción del año de lanzamiento de las películas, el cual se encuentra entre paréntesis al final de columna *title*. Mediante el empleo de expresiones regulares, se identifican los años y se registran en una nueva columna denominada *year*. Simultáneamente, se actualiza la columna *title*, eliminando la información del año y los espacios en blanco innecesarios, de modo que solo se conservara el nombre de la película. Adicionalmente, se realiza una reorganización de las columnas del DataFrame, estableciendo el orden *movieId, title, year y genres* para facilitar la manipulación y posterior análisis de los datos. Finalmente, se guarda el DataFrame procesado en un nuevo archivo CSV para disponer de los datos preparados y listos para su utilización en el recomendador basado en contenido.

Hay que mencionar que a pesar de haber realizado el procesamiento de ambas variables en este trabajo se ha desarrollado un algoritmo de recomendación basado en contenido centrado exclusivamente en los géneros de las películas. En un principio, se consideró incluir también el año de la película como atributo relevante. Sin embargo, durante la etapa de desarrollo y pruebas, se observó que la inclusión del año afectaba negativamente las recomendaciones, ya que otorgaba un peso excesivo a este atributo y se omitían recomendaciones lógicas. Un ejemplo ilustrativo de esta situación fue el caso de sugerir películas relacionadas con *Toy Story*, donde curiosamente no se incluía *Toy Story 2* como una recomendación obvia. Esta omisión contradecía la lógica de la recomendación, ya que *Toy Story 2* es una secuela directa de la película original.

Como resultado de este análisis, se tomó la decisión de no considerar el año de la película como un factor determinante en el algoritmo de recomendación. De esta manera, se evitó distorsionar las recomendaciones y se buscó mantener una coherencia lógica en las sugerencias proporcionadas. Por lo tanto, el algoritmo final se basó únicamente en los géneros de las películas, permitiendo generar recomendaciones más precisas y acordes a las preferencias del usuario, sin que el año de la película afectara negativamente la calidad de las recomendaciones.

Con los datos preparados de acuerdo a esta decisión, se procede al desarrollo del algoritmo basado en la técnica de *K-Nearest Neighbors (KNN)*. KNN es una técnica de Machine Learning que se fundamenta en la premisa de que elementos similares tienden a compartir características similares, lo cual resulta fundamental en la búsqueda de puntos cercanos en un espacio de atributos. En el contexto de los sistemas de recomendación basados en contenido, KNN busca establecer la similitud entre dos películas en función de sus atributos comunes, concretamente a partir de los géneros asociados.

Para ello, se utiliza el concepto de distancia, donde dos películas serán consideradas similares si la distancia entre ellas es pequeña, lo que indica que comparten géneros en común. Así, KNN permite identificar las películas más cercanas en términos de sus características, facilitando la generación de recomendaciones personalizadas y relevantes.

Para lograrlo, utilizamos la función de similitud del coseno, que es una medida popular y adecuada para comparar la similitud entre dos vectores en un espacio multidimensional [26]. La función coseno calcula la similitud entre dos películas al considerar el producto escalar entre los vectores que representan las películas en función de sus géneros. Esta función se define de la siguiente manera:

$$\text{CosSim(p,q)} = \frac{\sum_{i=1}^{n} p_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \cdot \sqrt{\sum_{i=1}^{n} q_i^2}} \tag{5.1}$$

En la ecuación 5.1, p y q representan los vectores que se asocian a las películas en función de sus géneros, y n la dimensión del espacio de atributos. La función coseno devuelve un valor entre -1 y 1, donde 1 indica una similitud máxima entre las películas y -1 indica una similitud mínima.

Al aplicar KNN con la función de similitud del coseno, identificamos las películas más cercanas en términos de sus características, lo que nos permite **generar recomendaciones personalizadas y relevantes sin depender de las valoraciones de los usuarios**. Este enfoque nos brinda recomendaciones precisas y acordes a las preferencias del usuario, basadas en la similitud de los géneros de las películas.

Hay que destacar que la medida del coseno se define utilizando frecuencias normalizadas mediante el uso de la ponderación TF-IDF. El término significa *Term Frequency-Inverse Document Frequency* (Frecuencia del Término-Frecuencia Inversa del Documento) [27], y se utiliza para analizar y representar características textuales de los artículos y evaluar así la relevancia de un término en un documento en comparación con una colección de documentos. La representación TF-IDF asigna un valor numérico a cada término en un documento, reflejando su importancia en el contexto de la colección de documentos. Esto se logra considerando dos factores: la frecuencia del término en el documento (TF) y la frecuencia inversa del término en la colección de documentos (IDF). La fórmula de TF-IDF para un término t en un documento d se calcula de la siguiente manera:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t) \tag{5.2}$$

Donde:

- *TF(t, d)* representa la frecuencia del término t en el documento d. Se puede calcular utilizando diferentes esquemas de ponderación, como la frecuencia bruta del término o la frecuencia relativa normalizada.

- *IDF(t)* es la frecuencia inversa del término t en la colección de documentos. Se calcula como el logaritmo del cociente entre el número total de documentos en la colección y el número de documentos que contienen el término t. Esto ayuda a penalizar los términos que aparecen en muchos documentos y, por lo tanto, tienen menos relevancia para la representación del contenido.

Al utilizar TF-IDF Vectorizer en un recomendador basado en contenido, se transforma el conjunto de características textuales de las películas en vectores TF-IDF. Una vez que se ha calculado la representación TF-IDF para cada película en función de sus características textuales, como en este caso, los géneros, se utiliza la medida de similitud del coseno para comparar las representaciones de las películas y determinar su similitud. Las películas más similares en términos de características textuales, como los géneros, se consideran como posibles recomendaciones para un usuario. Finalmente, para generar recomendaciones, se utiliza la información de los vecinos más cercanos.

Este enfoque tiene la ventaja de no depender de las valoraciones de los usuarios, lo que lo hace útil en situaciones donde no hay suficiente información de retroalimentación de los usuarios o para nuevos usuarios sin historial de valoraciones. Sin embargo, es importante tener en cuenta que este enfoque puede llevar a recomendaciones sesgadas y puede no capturar preferencias individuales más específicas de los usuarios.

Por último, es importante mencionar que debido a la estructura del dataset de MovieLens utilizado, las recomendaciones generadas por los algoritmos de recomendación basados en contenido pueden resultar relativamente simples. Esto se debe a que la única característica relevante presente en los datos es la información sobre los géneros de las películas. No obstante, se podrían conseguir recomendaciones más elaboradas y precisas, si se contasen con atributos adicionales, como el director de la película, los actores principales o incluso un embedding que represente la sinopsis de la película. Estos atributos podrían brindar una mayor profundidad y diversidad a las recomendaciones, permitiendo considerar aspectos más detallados y específicos de las películas.

Al incorporar información adicional, los algoritmos de recomendación basados en contenido podrían capturar mejor los gustos y preferencias de los usuarios, y ofrecer recomendaciones más personalizadas y enriquecedoras, lo que permitiría superar las limitaciones de la base de datos actual y brindar una experiencia de recomendación más satisfactoria y diversa para los usuarios de MovieLens y otros sistemas similares.

El algoritmo de recomendación basado en contenido desarrollado está disponible en el siguiente repositorio: https://github.com/inesgfortis/RecommenderSys/contentBased

5.2. Filtrado colaborativo

Según [1], los métodos de filtrado colaborativo se pueden considerar como generalizaciones de los problemas de clasificación y regresión. En estos problemas, la variable dependiente o *target* puede corresponder a un atributo con valores perdidos, mientras que las demás columnas se tratan como características independientes. El problema del filtrado colaborativo amplía este marco, permitiendo que cualquier columna, no solo la variable de clase, tenga valores perdidos.

Además, en el problema de recomendación, no se establece una distinción clara entre las variables de clase y las variables de características, ya que cada característica cumple el doble papel de variable dependiente e independiente. Esta distinción solo existe en el problema de clasificación, donde los valores faltantes se limitan a una columna especial. Además, en el filtrado colaborativo, no se distingue entre filas de entrenamiento y de prueba, ya que cualquier fila puede contener valores faltantes. Por lo que se habla de entradas de entrenamiento y de prueba en lugar de filas de entrenamiento y de prueba.

De esta forma, **el filtrado colaborativo es una generalización de los modelos de clasificación y regresión en el que la predicción se realiza por entradas en lugar de por filas** (ver Figura 5.1).

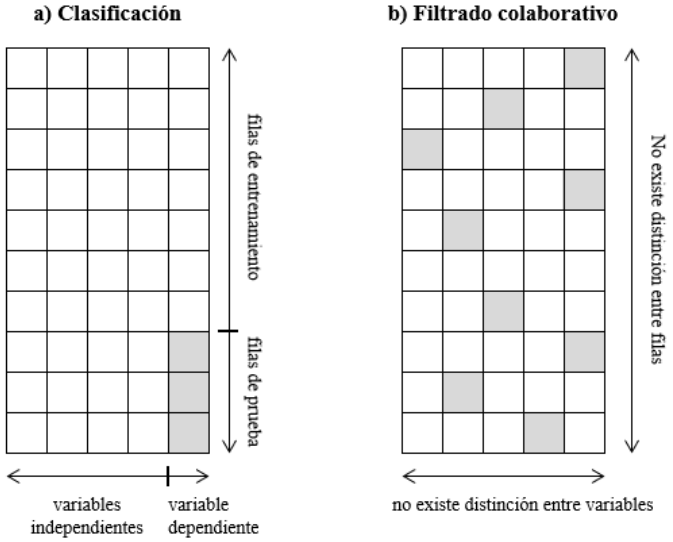


Figura 5.1: Comparación del problema de clasificación tradicional con el filtrado colaborativo. Las entradas sombreadas son datos que faltan y deben predecirse.

El filtrado colaborativo es un proceso en el que se utilizan las opiniones de otras personas para evaluar y filtrar elementos [28]. Por ejemplo, en un grupo de amigos, si la mayoría de ellos disfrutó de una película, aquellos que no la hayan visto podrían considerarla como una opción atractiva. Del mismo modo, si muchos de ellos consideran que una película es de baja calidad, es probable que otros decidan no invertir su tiempo en verla. Además, es posible notar que ciertos miembros del grupo recomiendan películas que se alinean con sus gustos personales, mientras que otros tienen una tendencia a recomendar obras de menor calidad o incluso recomendaciones indiscriminadas. Con el tiempo, cada miembro del grupo aprende a discernir qué opiniones tomar en cuenta y cómo aplicarlas para determinar la calidad de un artículo.

La idea central del filtrado colaborativo es procesar las opiniones en tiempo real y obtener una visión personalizada de un artículo. En lugar de basarse únicamente en la opinión de una comunidad amplia, se busca identificar las opiniones más relevantes para un usuario o grupo de usuarios en particular. De esta manera, el objetivo es desarrollar una visión verdaderamente personalizada de ese artículo, teniendo en cuenta las opiniones más apropiadas y significativas para el usuario o grupo en cuestión. Con el filtrado colaborativo, se busca aprovechar la diversidad de opiniones y las preferencias individuales para brindar recomendaciones más precisas y adaptadas a cada usuario.

Dado que las recomendaciones se basan en opiniones, en este tipo de algoritmos resulta de vital importancia disponer de valoraciones de usuarios. En este caso, al estar trabajando con el conjunto de datos de MovieLens, los usuarios califican las películas empleando el sistema de valoración estrellas, donde 1 es "Horrible" y 5 corresponde con "Peliculón", de forma que estas puntuaciones se pueden utilizar para recomendar otras películas que puedan interesar al usuario, predecir la valoración que ese usuario podría hacer de una película o realizar otras tareas.

En la última década, los algoritmos de filtrado colaborativo han experimentado una evolución significativa, pasando de ser algoritmos de investigación que intuitivamente captan las preferencias de los usuarios, a algoritmos que cumplen con los requisitos de rendimiento de las grandes aplicaciones comerciales [28]. En esta sección, exploraremos los algoritmos de filtrado colaborativo más conocidos, como el basado en vecinos más cercanos (*k-Nearest-Neighbors*), el cual se divide en dos enfoques: basado en usuarios (user-based) y basado en elementos (item-based).

5.2.1. Basado en usuarios

Como se mencionó en la sección 2.3.2, en los algoritmos de filtrado colaborativo basados en usuarios las predicciones se generan utilizando las valoraciones de usuarios similares, a los cuales nos referimos como vecinos. Si un usuario v es similar a un usuario u, decimos que v es vecino de u. Los algoritmos basados en usuarios hacen predicciones para un elemento específico, examinando las valoraciones que los usuarios en el vecindario de referencia han dado a ese mismo elemento [28]. Esto se ilustra en la Figura 5.2, donde se muestra un ejemplo de recomendaciones para un usuario concreto en función de otro que aparentemente comparte sus gustos.

Figura 5.2: Ejemplo de filtrado colaborativo basado en usuarios

En la Figura 5.2, se pueden observar dos usuarios que han visto y disfrutado de las películas *"Tienes un e-mail"* y *"Sleepless in Seattle"*, lo que sugiere que tienen gustos similares. Además, notamos que la usuaria ha visto la película "Joe contra el volcán". Esta película se recomienda al otro usuario, ya que se entiende que si a usuarios con gustos similares les ha gustado, es probable que también le guste al otro usuario.

Para desarrollar este algoritmo, seguimos los pasos descritos en [16]. En primer lugar, se crea una matriz que contiene la información de las valoraciones de las películas por parte de los usuarios. En esta matriz, las filas representan a los usuarios, las columnas representan a las películas, y el valor de cada celda corresponde a la valoración que el usuario ha dado a esa película. Si un usuario no ha valorado una película, la celda correspondiente se deja vacía (ver Cuadro 5.1).

Cuadro 5.1: Matriz usuario-película-rating

	Indiana Jones	Star Wars	Toy Story	Casablanca
usuario 1	4	5		
usuario 2	1		5	
usuario 3				1

En este proceso, es necesario calcular las similitudes entre los usuarios. Para ello, se define un diccionario llamado *sim_options* que incluye los siguientes parámetros:

- *name*: indica el tipo de medida de similitud que se utilizará para calcular las similitudes entre usuarios. En este caso, se utiliza el coeficiente de correlación de Pearson, el cual se define en la ecuación 5.3. Otra opción común es la similitud coseno, previamente definido (Ecuación 5.1). La razón por la cual se ha seleccionado el coeficiente de correlación de Pearson es su capacidad para considerar la escala y el rango de las valoraciones de los usuarios. Esto significa que se tiene en cuenta si los usuarios tienen diferentes niveles de calibración en sus valoraciones. Por ejemplo, si un usuario tiende a ser más generoso en sus valoraciones en comparación con otro usuario, el coeficiente de correlación de Pearson puede capturar esta diferencia y proporcionar una similitud más precisa en el cálculo de las recomendaciones.

- *user_based*: determina si se calcularán las similitudes entre usuarios (user-based) o entre elementos (item-based). En este caso, se establece en True para calcular las similitudes entre usuarios.

- *min_support*: especifica el número mínimo de elementos en común requeridos entre usuarios para considerar su similitud. Se establece en 5, lo que significa que solo se considerarán aquellos usuarios que tengan al menos 5 elementos en común. Esto se hace para evitar considerar usuarios con muy poca información en común.

El coeficiente de correlación de Pearson se define de la siguiente manera:

$$\text{Pearson Correlation}(u,v) = \frac{\sum_{i \in CR_{u,v}} (R_{u,i} - \overline{R_u})(R_{v,i} - \overline{R_v})}{\sqrt{\sum_{i \in CR_{u,v}} (R_{u,i} - \overline{R_u})^2} \sqrt{\sum_{i \in CR_{u,v}} (R_{v,i} - \overline{R_v})^2}} \tag{5.3}$$

La fórmula 5.3 establece la relación lineal entre las valoraciones de los usuarios u y v para un conjunto de elementos comunes, representados por $CR_{u,v}$. Utiliza las valoraciones $R_{u,i}$ y $R_{v,i}$ de los usuarios para un elemento específico i, comparándolas con los promedios de las valoraciones de cada usuario ($\overline{R_u}$ y $\overline{R_v}$).

La correlación de Pearson oscila entre 1 (usuarios semejantes) y -1 (usuarios opuestos) y se utiliza en los sistemas de recomendación para identificar usuarios con gustos similares y generar recomendaciones basadas en estas similitudes. En general, se cree que las correlaciones negativas no son valiosas para aumentar la precisión de las predicciones [29] y se puede optar por no utilizarlas [28].

Luego, se crea un objeto de la clase KNNWithMeans a partir de la configuración establecida el diccionario *sim_ options*. Este objeto representa el modelo de recomendación basado en KNN con las opciones de similitud especificadas. A continuación, se entrena el modelo utilizando el conjunto de datos de entrenamiento que contiene las valoraciones de los usuarios y se calcula la matriz de similitudes utilizando el método *compute_ similarities()* propio del modelo KNNWithMeans de Surprise [22]. El resultado de todo este proceso es una matriz que contiene las similitudes calculadas entre los usuarios del conjunto de datos.

A continuación, se realiza la comparación de usuarios para identificar aquellos que tienen perfiles similares al del usuario al cual se desea recomendar. Para esto, se utiliza la función $getNeighbors$, que recibe como parámetros un usuario de referencia (*referenceUser*), un número k que indica la cantidad de vecinos deseados, un conjunto de entrenamiento (*trainSet*) y un umbral opcional (*threshold*). El propósito de esta función es obtener los vecinos más similares al usuario de referencia basándose en la matriz de similitudes previamente calculada.

En primer lugar, se convierte el usuario de referencia en el identificador interno correspondiente dentro del conjunto de entrenamiento. Luego, se verifica si se ha proporcionado un umbral. Si no se ha especificado, se obtienen los usuarios similares ordenados según su similitud con el usuario de referencia. Para ello, se recorre la fila de similitudes correspondiente al usuario de referencia en la matriz, y se guardan los usuarios similares junto con sus puntuaciones en una lista de usuarios similares. El propio usuario de referencia se excluye de esta lista. En caso de que se haya especificado un umbral, se seleccionan únicamente aquellos usuarios cuya similitud supere dicho umbral, y se almacenan en la lista de usuarios similares. Finalmente, se ordena la lista en forma descendente según sus puntuaciones, de tal forma que aquellos con mayor similitud se encuentran los primeros, y se seleccionan los k primeros vecinos utilizando la función nlargest de la biblioteca heapq. Como resultado de la función, se devuelve la lista de los vecinos más cercanos.

Una vez encontrados los usuarios más similares, se procede a buscar las recomendaciones para el usuario de referencia. Para cada vecino, se calcula una puntuación de candidatura para cada ítem, teniendo en cuenta la similitud del vecino y la valoración asignada al ítem, y se almacena dicha puntuación en un diccionario. Por último, se aplican filtros para eliminar las recomendaciones con puntuaciones bajas y las que el usuario ya haya visto, y se devuelve la lista de recomendaciones pensadas para el usuario.

El algoritmo de recomendación basado en usuarios desarrollado está disponible en el siguiente repositorio: https://github.com/inesgfortis/RecommenderSys/collaborativeFiltering

5.2.2. Basado en elementos

Los algoritmos de vecino más próximo basados en elementos son la transposición de los algoritmos basados en usuarios [28]. Mientras que los algoritmos basados en usuarios generan predicciones basadas en similitudes entre usuarios, los algoritmos basados en ítems generan predicciones basadas en similitudes entre ítems [30]. La predicción de un ítem debe basarse en las valoraciones de un usuario sobre ítems similares. Por ejemplo, si un recomendador de filtrado colaborativo basado en elementos detecta que a un usuario le gustaron las películas *"Tienes un e-mail"* y *"Sleepless in Seattle"*, podría recomendar otra película con los mismos géneros y/o reparto, como *"Joe contra el volcán"*, tal y como se muestra en la Figura 5.3.

Figura 5.3: Ejemplo de filtrado colaborativo basado en elementos

Los algoritmos basados en ítems son similares a los algoritmos basados en usuarios, pero presentan algunas diferencias clave. Para desarrollar este tipo de algoritmo, seguimos los pasos descritos en [16]. En lugar de crear una matriz de usuarios y sus valoraciones de películas, creamos una matriz que muestra las valoraciones por película realizadas por cada usuario existente. En esta matriz, las filas representan las películas, las columnas representan a los usuarios, y el valor de cada celda indica la valoración asignada a esa película por el respectivo usuario. Si un usuario no ha valorado una película en particular, la celda correspondiente se mantiene vacía (ver Cuadro 5.2).

Es importante destacar que esta matriz es la transpuesta de la matriz utilizada en el enfoque basado en usuarios. No obstante, existen pruebas de que los algoritmos de vecino más próximo basados en elementos son más precisos a la hora de predecir valoraciones que sus homólogos basados en usuarios [30].

Cuadro 5.2: Matriz película-usuario-rating

	usuario 1	usuario 2	usuario 3
Indiana Jones	4	1	
Star Wars	5		
Toy Story		5	
Casablanca			1

En este proceso, es necesario calcular las similitudes entre los artículos. Para lograrlo, se utiliza un diccionario adicional llamado *sim_options*, el cual tiene los mismos parámetros mencionados anteriormente pero con una configuración diferente específica para el cálculo de similitudes entre artículos. A continuación, se describe la configuración utilizada en este caso particular.

- *name*: indica el tipo de medida de similitud que se utilizará para calcular las similitudes entre palículas. En este caso, se utiliza similitud coseno ajustada, la cual se define en la ecuación 5.4. La razón por la cual en esta ocasión se ha seleccionado esta métrica se debe a que es la métrica de similitud más popular y considerada la más precisa para este tipo de algoritmos.
- *user_based*: determina si se calcularán las similitudes entre usuarios (user-based) o entre elementos (item-based). En este caso, se establece en False para calcular las similitudes entre elementos.
- *min_support*: especifica el número mínimo de elementos en común requeridos entre usuarios para considerar su similitud. Tal y como ocurría en el caso basado en usuarios, se establece en 5, lo que significa que solo se considerarán aquellos elementos (películas) que tengan al menos 5 usuarios en común que las hayan valorado. Este parámetro se utiliza para evitar que se consideren similitudes basadas en un número muy bajo de usuarios, lo cual podría llevar a recomendaciones poco confiables o poco representativas.

La métrica de similitud coseno ajustada puede calcularse de la siguiente manera:

$$\text{itemSim}(i,j) = \frac{\sum_{u \in \text{RB}i,j}(R_{u,i} - \overline{R_i})(R_{u,j} - \overline{R_j})}{\sqrt{\sum_{u \in \text{RB}i,j}(R_{u,i} - \overline{R_i})^2}\sqrt{\sum_{u \in \text{RB}i,j}(R_{u,j} - \overline{R_j})^2}} \tag{5.4}$$

La ecuación 5.4, calcula la similitud coseno entre los vectores de calificación de usuario de los elementos i y j, con ajustes basados en las calificaciones promedio de los elementos. $R_{u,i}$ representa la calificación del usuario u para el elemento i, y $\overline{R_i}$ denota la calificación promedio del elemento i. De manera similar, $R_{u,j}$ representa la calificación del usuario u para el elemento j, y $\overline{R_j}$ denota la calificación promedio del elemento j. RBi, j denota el conjunto de usuarios que han calificado tanto el elemento i como el elemento j.

La única diferencia con la correlación de Pearson (Ecuación 5.3) es que el ajuste del promedio se realiza con respecto al usuario, no al elemento [28]. Al igual que en la correlación de Pearson con el usuario, el valor de correlación oscila entre -1 y 1.

El siguiente paso es similar al del algoritmo basado en usuarios. Primero, se crea un objeto de la clase KNNWithMeans utilizando la configuración especificada en el diccionario *sim_ options*. Este objeto representa el modelo de recomendación basado en KNN con las opciones de similitud definidas.

Luego, se entrena el modelo utilizando el conjunto de datos de entrenamiento que contiene las valoraciones de los usuarios y se calcula la matriz de similitudes utilizando el método *compute_similarities()* proporcionado por el modelo KNNWithMeans de la biblioteca Surprise [22]. Este proceso resulta en una matriz que contiene las similitudes calculadas entre los elementos del conjunto de datos.

A continuación, se buscan las k películas que el usuario de referencia ha valorado mejor, donde se seleccionan las k primeras películas con la mayor puntuación (en este caso, se seleccionan las 10 primeras). Después, se realiza el cálculo de los ítems similares a aquellos que le gustaron al usuario de referencia, considerando la similitud ponderada por la valoración asignada. Se utiliza un diccionario llamado *candidates* para almacenar los ítems similares y sus puntuaciones. Mediante un bucle, se recorren los ítems vecinos junto con sus valoraciones. Para cada ítem vecino, se obtiene la fila de similitud correspondiente de la matriz de similitudes y se recorre esta fila para calcular la puntuación ponderada de cada ítem. La puntuación se calcula multiplicando la similitud del ítem por la valoración asignada por el usuario al ítem vecino, normalizada entre 0 y 1. Luego, los candidatos se ordenan de manera descendente según su puntuación para obtener los ítems más similares y relevantes. Finalmente, se filtran aquellas recomendaciones con una puntuación baja y que el usuario ya ha visto, y se devuelve la lista de recomendaciones para el usuario.

El algoritmo de recomendación basado en elementos desarrollado está disponible en el siguiente repositorio: https://github.com/inesgfortis/RecommenderSys/collaborativeFiltering

Los algoritmos de filtrado colaborativo ofrecen ventajas significativas en la generación de recomendaciones personalizadas. Son flexibles, aplicables en diversos dominios y permiten descubrir nuevos elementos de interés para los usuarios. Estos sistemas aprovechan la sabiduría colectiva de la comunidad de usuarios, se adaptan a cambios en las preferencias y proporcionan recomendaciones relevantes y adaptadas a cada individuo. No obstante, presentan una limitación importante: dependen de una comunidad de usuarios que estén familiarizados entre sí [28]. Los enfoques activos, tanto pull-active como push-active, requieren que los usuarios tengan conocimiento sobre las opiniones en las que confiar y el contenido que puede interesar a otros usuarios específicos. Además, los métodos basados en el usuario y en el elemento tienen sus propias limitaciones en cuanto a la consideración de la similitud entre filas y columnas en la matriz de valoraciones [1]. Sin embargo, existen modelos como el algoritmo SVD++, el cual se abordará en la siguiente sección, que combinan ambos métodos para mejorar la capacidad de recomendación. Por último, es importante destacar que, aunque los métodos basados en vecinos más cercanos fueron populares en el pasado, en la actualidad existen modelos más precisos disponibles en el campo de los sistemas recomendadores [1].

5.3. Técnicas de factorización de matrices

Las técnicas de factorización de matrices son un enfoque eficaz para abordar el problema de la recomendación en sistemas de filtrado colaborativo. Estas técnicas permiten descomponer una matriz de valoraciones en factores latentes, revelando así las relaciones ocultas entre los usuarios y los elementos. Al utilizar algoritmos de factorización, como la descomposición en valores singulares (SVD), se pueden extraer patrones y características subyacentes que ayudan a comprender los gustos y preferencias de los usuarios [31].

La idea fundamental detrás de la factorización de matrices es representar la matriz de valoraciones como el producto de dos matrices más pequeñas. Una matriz representa a los usuarios y la otra a los elementos, donde las dimensiones de ambas matrices son mucho menores que la matriz original (ver Figura 5.4). Estas matrices de factores latentes capturan las características esenciales de los usuarios y los elementos, lo que permite calcular las valoraciones desconocidas con base en la información disponible [31].

De acuerdo con [1], se conoce el modelo básico de factorización matricial como aquel en el que se observan todas las entradas de la matriz de valoraciones R. La idea clave es que cualquier matriz $m \times n$, R, de rango k $min\{m, n\}$ siempre puede expresarse en la siguiente forma de producto de factores de rango-k:

$$R \approx UV^T \tag{5.5}$$

En la ecuación 5.5, U es una matriz $m \times k$, y V es una matriz $n \times k$. Obsérvese que el rango tanto del espacio de filas como del espacio de columnas de R es k. Cada columna de U (o V) se denomina vector latente o componente latente, mientras que cada fila de U (o V) se denomina factor latente. La fila i de U (u_i) se denomina factor de usuario y contiene k entradas correspondientes a la afinidad del usuario i hacia los k conceptos de la matriz de valoraciones.

Por ejemplo, en el caso de la Figura 5.4, u_i es un vector bidimensional que contiene la afinidad del usuario i hacia los géneros histórico y romance en la matriz de valoraciones R. Del mismo modo, cada fila v_i de V se denomina factor de ítem y representa la afinidad del ítem i hacia estos k conceptos. En la figura 5.4, el factor de ítem contiene la afinidad del ítem hacia las dos categorías de películas.

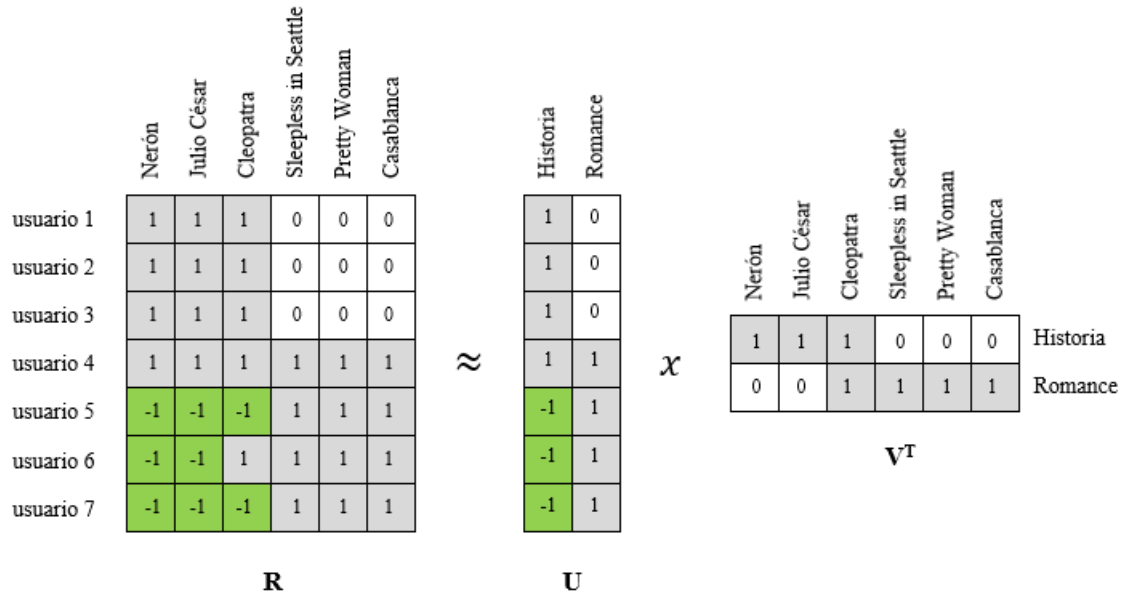


Figura 5.4: Ejemplo de factorización de matrices de rango 2

En la matriz de ratings resultante, los valores (-1, 0, 1) tienen un significado específico. Un valor de (-1) indica una valoración negativa o baja preferencia del usuario hacia el elemento, mientras que un valor de (0) indica una falta de valoración o una opinión neutral.

Por otro lado, un valor de (1) indica una valoración positiva o alta preferencia del usuario hacia el elemento. De acuerdo con esto, en la Figura 5.4 se observa que los tres primeros usuarios muestran una alta preferencia por las películas históricas, marcadas con un valor (1) en la matriz V^T , pero no se conoce su opinión acerca de los romances. El cuarto usuario muestra disfrute por ambos géneros, mientras que los tres últimos usuarios tienen una preferencia hacia los romances pero no hacia las películas históricas.

De la ecuación 5.5 se deduce que cada calificación r_{ij} en R puede expresarse aproximadamente como un producto punto del i-ésimo factor de usuario y el j-ésimo factor de artículo:

$$r_{ij} \approx \bar{u}_i \cdot \bar{v}_j \tag{5.6}$$

En el caso de la figura 5.4, los dos conceptos de la suma mencionada corresponden a los géneros romance e histórico. Por lo tanto, la suma puede expresarse de la siguiente manera:

$$\begin{aligned} r_{ij} \approx & (\text{Afinidad del usuario } i \text{ hacia la historia}) \times (\text{Afinidad del ítem } j \text{ hacia la historia}) + \\ & (\text{Afinidad del usuario } i \text{ hacia el romance}) \times (\text{Afinidad del ítem } j \text{ hacia el romance}) \end{aligned} \tag{5.7}$$

Las técnicas de factorización de matrices han demostrado ser altamente eficientes y precisas en la generación de recomendaciones personalizadas, ya que abordan las limitaciones de los métodos basados en vecinos más cercanos al modelar las interacciones entre usuarios y elementos en términos de factores latentes. Además, su capacidad para superar la dispersión de los datos y falta de información en los conjuntos, así como su manejo efectivo de grandes volúmenes de información, las convierte en una herramienta poderosa en los sistemas de recomendación [31, 32].

En resumen, las técnicas de factorización de matrices permiten descubrir relaciones ocultas y generar recomendaciones personalizadas de forma precisa, aunque continúan siendo objeto de investigación y desarrollo constante para mejorar su precisión y adaptabilidad a diferentes escenarios de recomendación [31, 32].

Es importante destacar que existen diferencias clave entre los métodos de factorización matricial en términos de las restricciones aplicadas a los factores latentes (por ejemplo, ortogonalidad o no negatividad) y la función objetivo utilizada (por ejemplo, minimización de la norma de Frobenius [33] o maximización de la estimación de la verosimilitud en un modelo generativo [34]). Estas diferencias son cruciales para evaluar la utilidad de los modelos de factorización matricial en diversos escenarios del mundo real [1].

A continuación se explorarán en detalle los algoritmos de recomendación basados en SVD. Estos algoritmos se basan en la premisa de que las valoraciones de los usuarios hacia los elementos pueden ser aproximadas mediante una combinación lineal de factores latentes. La idea es representar tanto a los usuarios como a los elementos en un espacio de menor dimensión, donde los factores latentes capturan características subyacentes relevantes para las preferencias de los usuarios.

La forma más fundamental de factorización de matrices es el caso sin restricciones, en el que no se imponen restricciones a las matrices factoriales U y V. Gran parte de la literatura de recomendaciones se refiere a la factorización de matrices sin restricciones como descomposición de valores singulares (SVD) [1].

La descomposición de valores singulares (SVD) es una forma de factorización de matrices en la que las columnas de U y V deben ser mutuamente ortogonales. La ortogonalidad mutua tiene la ventaja de que los conceptos pueden ser completamente independientes entre sí y pueden interpretarse geométricamente en diagramas de dispersión. Sin embargo, la interpretación semántica de una descomposición de este tipo suele ser más difícil, porque estos vectores latentes contienen cantidades tanto positivas como negativas, y están limitados por su ortogonalidad respecto a otros conceptos [1]. Para una matriz totalmente especificada, es relativamente fácil realizar la SVD con el uso de métodos de eigendecomposición, que son técnicas que buscan encontrar una representación más simple de la matriz original que permita analizar sus propiedades y realizar operaciones matriciales de manera más eficiente.

De acuerdo con la notación establecida anteriormente, se parte de una matriz R de $m \times n$ cuyo elemento r_{ij} corresponde a la calificación del usuario i al objeto j (si aún no se ha dado la calificación, el elemento correspondiente de R es cero). En el caso sin valoraciones numéricas, R se convierte en la matriz de adyacencia, ya que $r_{ij} = 0, 1$ para pares usuario-objeto conectados y no conectados, respectivamente. El proceso de recomendación tiene como objetivo determinar qué entradas de R que actualmente son cero tienen una alta probabilidad de ser distintas de cero en el futuro [2]. Obsérvese que R es una matriz dispersa para la mayoría de las aplicaciones, ya que sólo una pequeña fracción de todos sus elementos es distinta de cero. La reducción de la dimensionalidad se consigue introduciendo K variables ocultas que categorizan los gustos de los usuarios y los atributos de los objetos.

En este contexto, el algoritmo SVD busca encontrar la mejor aproximación de la matriz original de valoraciones mediante la descomposición en valores singulares. Este proceso implica identificar los vectores más significativos y utilizarlos para reconstruir las valoraciones originales. El resultado final es una matriz de predicciones que representa las valoraciones estimadas para los elementos no valorados por los usuarios.

Para lograr esto, se requiere una función objetivo que esté definida en términos de las entradas observadas para determinar las matrices U y V, de manera que la matriz R se ajuste lo mejor posible a UV^T. El conjunto S, que consiste en todos los pares usuario-artículo (i, j) observados en R, donde i toma valores de 1 a m como el índice de un usuario, y j toma valores de 1 a n como el índice de un artículo, se define de la siguiente manera:

$$S = \{(i, j) : r_{ij} \text{ observados}\} \tag{5.8}$$

Según [1], al factorizar la matriz incompleta R como la aproximación UV^T de matrices completamente especificadas $U = [u_{is}]_{m \times k}$ y $V = [v_{js}]_{n \times k}$, es posible predecir todas las entradas de R. Específicamente, la entrada (i, j) de la matriz R se puede predecir de la siguiente manera:

$$\hat{r}_{ij} = \sum_{s=1}^{k} u_{is} \cdot v_{js} \tag{5.9}$$

En la ecuación 5.9, el símbolo del cincunflejo indica que se trata de un valor predicho en lugar de un valor observado. De acuerdo con dicha ecuación, la diferencia entre el valor observado y el predicho de una entrada específica (i, j) viene dada por $e_{ij} = (r_{ij} - \hat{r}_{ij}) = (r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js})$.

No obstante, hay que tener considerar que en estos casos el conjunto observado S de valoraciones es pequeño, lo que puede provocar un sobreajuste. Un enfoque común para abordar este problema es utilizar la regularización, la cual reduce la tendencia del modelo a sobreajustarse a costa de introducir un sesgo en el modelo [1].

Teniendo en cuenta todo lo mencionado, la función objetivo definida en términos de las entradas observadas, se puede calcular únicamente sobre las entradas observadas en S de la siguiente manera:

$$\text{Minimizar J} = \frac{1}{2} \sum_{(i,j) \in S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right)^2 + \frac{\lambda_1}{2} \sum_{i=1}^{m} \sum_{s=1}^{k} u_{is}^2 + \frac{\lambda_2}{2} \sum_{j=1}^{n} \sum_{s=1}^{k} v_{js}^2 \qquad (5.10)$$

En la ecuación 5.10, los elementos u_{is} y v_{js} son las variables desconocidas que necesitan ser aprendidas para minimizar la función objetivo, algo que se puede lograr mediante métodos de descenso de gradiente [1]. Los valores λ_1 y λ_2 representan los parámetros de regularización, los cuales se pueden elegir utilizando los métodos de *hold-out* o de *cross-validation*. Además, la ecuación tiene la restricción de que las columnas de la matriz U deben ser mutuamente ortogonales, al igual que las columnas de la matriz V.

Para elaborar un algoritmo SVD (Descomposición en Valores Singulares) se utiliza la librería Surprise[22] en Python, lo que brinda la capacidad de generar recomendaciones personalizadas basadas en la descomposición en valores singulares. Para ello se deben cargar y preparar los datos de tal forma que estos contengan las columnas "userID"e "itemID"que representan los identificadores del usuario y el elemento respectivamente. Una vez que los datos están cargados, dividen en conjuntos de entrenamiento y prueba utilizando para poder evaluar la precisión del modelo posteriormente en datos no vistos. A continuación, se crea una instancia del modelo SVD utilizando la clase SVD existente. Hay que destacar que se pueden ajustar los parámetros del modelo según sea necesario mediante el ajuste de hiperparámetros, aunque en este proyecto se ha trabajado con los valores por defecto. Luego, se entrena el modelo utilizando el método fit y el conjunto de entrenamiento. Una vez que el modelo está entrenado, se realizan predicciones utilizando el conjunto de prueba con el método test. Esto proporcionará las valoraciones predichas para cada par usuario-elemento en el conjunto de prueba.

La Descomposición en Valores Singulares (SVD) es ampliamente utilizada en sistemas de recomendación debido a su capacidad para capturar la estructura latente de los datos y generar recomendaciones precisas. Sin embargo, los algoritmos basados en SVD presentan ciertas limitaciones. No consideran explícitamente la retroalimentación implícita de los usuarios, lo que afecta la precisión de las recomendaciones. Además, no tienen en cuenta la variabilidad en la calidad de las valoraciones y asumen que todas son igualmente confiables. Estas limitaciones pueden afectar su adaptabilidad a conjuntos de datos complejos y la precisión de las recomendaciones.

Para abordar estos problemas, se han propuesto varios marcos, como los modelos factoriales asimétricos y SVD++, que permiten incorporar la retroalimentación implícita. Estos algoritmos utilizan dos matrices factoriales de ítems diferentes: V para la retroalimentación explícita e Y para la retroalimentación implícita. Los factores latentes de usuario se derivan mediante una combinación lineal de las filas de la matriz Y correspondientes a los ítems valorados por el usuario. Esto permite que los factores del usuario reflejen sus preferencias influenciadas por los ítems que han elegido puntuar [1].

En la versión más sencilla de los modelos factoriales asimétricos, se utiliza una combinación lineal de los vectores factoriales de los ítems valorados para crear los factores de usuario. Esto da lugar a un enfoque asimétrico en el que ya no tenemos variables independientes para los factores de usuario. En cambio, tenemos dos conjuntos de factores de ítem independientes, explícitos e implícitos, y los factores de usuario se derivan como una combinación lineal de los factores de ítem implícitos.[1].

El modelo SVD++ combina este enfoque asimétrico con factores de usuario explícitos y un marco de factorización tradicional. Como consecuencia, el enfoque asimétrico puede considerarse como un precursor simplificado del SVD++ [1]. Este algortimo se introdujo como una extensión del SVD tradicional para mejorar la precisión y la adaptabilidad al considerar tanto las valoraciones explícitas de los usuarios como su comportamiento implícito. Esto se logra mediante la incorporación de información adicional, como desviaciones de usuario y desviaciones de elemento en el modelo, para capturar mejor la diversidad de las preferencias y ajustar las recomendaciones de manera precisa [32]. Además, al tener en cuenta la calidad variable de las valoraciones y las diferencias en la confiabilidad de los usuarios, el SVD++ genera recomendaciones más robustas y adaptadas a las preferencias individuales [32].

En resumen, SVD++ representa una mejora significativa sobre el algoritmo SVD tradicional al considerar tanto la retroalimentación explícita como implícita de los usuarios, lo que resulta en recomendaciones más precisas y personalizadas [32, 35].

A la hora de implementar este algoritmo en Python se realizan los mismos pasos que en el caso del algoritmos SVD, solo que en esta ocasión se crea una instancia del modelo utilizando la clase SVD++ existente.

Los algoritmos de factorización de matrices desarrollados está disponibles en el siguiente repositorio: https://github.com/inesgfortis/RecommenderSys/matrixFactorization

5.4. Deep learning aplicado a recomendadores

El Deep Learning es una disciplina avanzada dentro del campo de Machine Learning que ha transformado significativamente la capacidad de los algoritmos para aprender representaciones de alto nivel y descubrir características complejas de manera automática. En este contexto, se examinarán dos técnicas de vanguardia: las Restricted Boltzmann Machines (RBM) y los autoencoders (sección 5.4.1 y 5.4.2 respectivamente). Estas técnicas de vanguardia ofrecen un enfoque poderoso para modelar patrones y relaciones subyacentes en los datos, permitiendo la captura y representación efectiva de la información latente en los sistemas recomendadores.

A través de un análisis riguroso y detallado de las RBM y los autoencoders, se explorará en profundidad su estructura, principios de funcionamiento y su aplicación específica en la generación de recomendaciones precisas y personalizadas. Esta sección proporcionará una visión integral de la contribución del Deep Learning en la mejora de los sistemas recomendadores, así como una comprensión clara de su relevancia en el ámbito de la investigación en este campo.

5.4.1. Restricted Boltzmann Machines

De acuerdo con [16], la Restricted Boltzmann Machine (RBM), conocida como máquina de Boltzmann restringida en español, ha sido pionera en el campo de las redes neuronales aplicadas a los sistemas de recomendación desde 2007. A pesar de haber sido desarrollada mucho antes que algunas disciplinas médicas como la cirugía ocular, la RBM sigue siendo ampliamente utilizada en la actualidad, tanto en la investigación académica como en la implementación práctica de sistemas recomendadores.

Son modelos generativos basados en la teoría de las máquinas de Boltzmann [36]. Estas redes neuronales probabilísticas y no supervisadas se utilizan para aprender representaciones latentes de los datos de entrada, lo que las hace especialmente adecuadas para abordar el problema del filtrado colaborativo en sistemas de recomendación [37].

La estructura de una RBM consta de dos capas (ver Figura izquierda 5.5): una capa visible que representa los datos de entrada, como usuarios o elementos, y una capa oculta que captura características latentes [36].

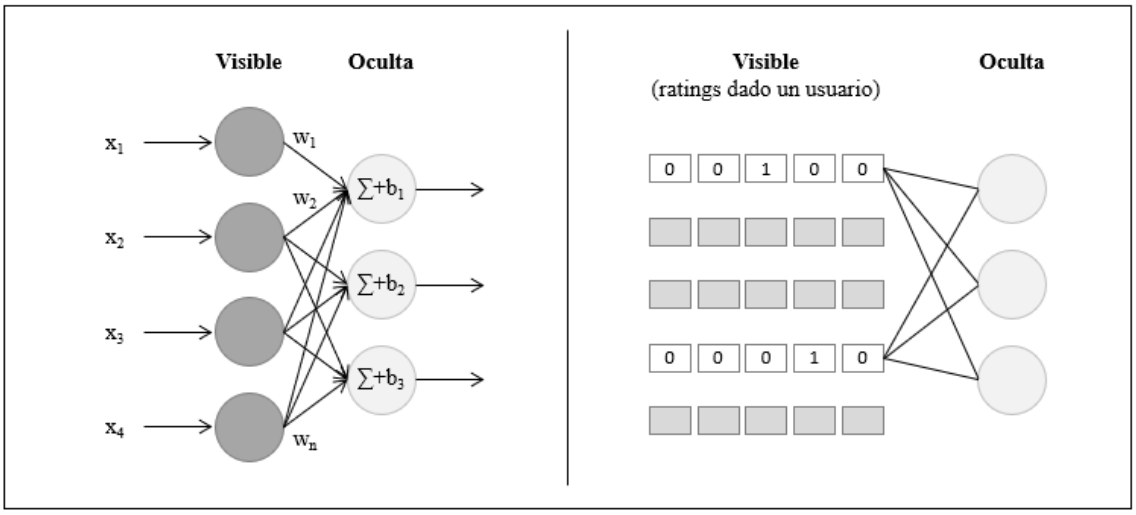


Figura 5.5: Izquierda) Estructura de una red RBM tradicional. Derecha) Estructura de una red RBM adaptada a un sistema recomendador.

Las RBM destacan por su capacidad para capturar relaciones no lineales y aprender representaciones de alta dimensionalidad. Además, utilizan representaciones factorizadas del amplio espacio de parámetros para reducir el sobreajuste [37].

Durante la fase de entrenamiento, se llevan a cabo dos pasos principales: el *forward pass* y el *backward pass*, los cuales permiten actualizar los pesos y las conexiones de la red para maximizar la probabilidad de los datos observados [36]. El *forward pass* implica la propagación de las activaciones desde las unidades visibles hasta las unidades ocultas, mientras que el *backward pass* supone la propagación desde las unidades ocultas hasta las unidades visibles para reconstruir los datos de entrada. Estos pasos se utilizan para calcular los gradientes y actualizar los pesos de la red durante el entrenamiento. Además, durante el entrenamiento, las unidades visibles se activan en función de las valoraciones de los ítems por parte de los usuarios, lo que permite aprender las relaciones entre ellos.

Estas características han demostrado ser muy efectivas en el campo de los sistemas de recomendación y han logrado un rendimiento destacado en el concurso *Netflix Prize*, una competición lanzada por Netflix en 2006 para mejorar su algoritmo de recomendación de películas [31].

En primer lugar, se ha llevado a cabo la preparación de los datos para que se ajusten a la entrada esperada por el algoritmo RBM. Esto implica la creación de una matriz en la que cada fila representa a un usuario y cada celda de la fila representa la valoración dada a una película (ver Figura derecha 5.5). Con el fin de lograr esta representación matricial adecuada para el modelo, se ha implementado una función encargada de realizar esta transformación en el conjunto de entrenamiento (Listing 5.1).

Listing 5.1: Tratamiento de datos RBM

```
def fitData(trainset):

    num_users = trainset.n_users
    num_movies = trainset.n_items

    # 3D matrix: users, movies and ratings
    # Ratings has size 10 given the possible rating values
    trainingMatrix = np.zeros([num_users, num_movies, 10],dtype=np.float32)

    for (uid, iid, rating) in trainset.all_ratings():
        adjustedRating = int(float(rating)*2.0) - 1
        trainingMatrix[int(uid), int(iid), adjustedRating] = 1

    # Flatten to a 2D array, with nodes for each possible rating type on
    each possible item, for every user.
    trainingMatrix = np.reshape(trainingMatrix,[trainingMatrix.shape[0],
    -1])

    return trainingMatrix
```

Una vez que los datos han sido preparados, se procede a desarrollar el algoritmo RBM. Dado que el paquete de Python *Surprise* [22] no proporciona un modelo RBM preentrenado que satisfaga nuestras necesidades, fue necesario crear el modelo desde cero. Para ello, se utilizó como punto de partida el código desarrollado por el autor del curso *Building Recommender Systems with Machine Learning and AI* [16], adaptándolo posteriormente a los requisitos específicos del proyecto. Esto resultó en la creación de la clase RBM, una subclase personalizada que implementa funcionalidades adicionales basadas en la clase *AlgoBase* de la biblioteca *Surprise*.

La clase RBM desempeña un papel fundamental en el entrenamiento y la utilización de un modelo RBM para generar recomendaciones de calificaciones de películas para cada usuario. A continuación, se describe brevemente la funcionalidad de cada componente clave de la clase, permitiendo comprender su estructura y operación interna:

- El **método fit** se encarga de entrenar la RBM utilizando un conjunto de datos de entrenamiento y una matriz de entrenamiento. Se ajustan los parámetros de la RBM y se realizan las iteraciones necesarias para mejorar el modelo.
- El **método Train** se utiliza para entrenar la RBM. Aquí, se inicializan los pesos y los sesgos de la RBM, y se realizan las iteraciones necesarias para ajustar los pesos y los sesgos en función de los datos de entrenamiento.
- El **método GetRecommendations** se utiliza para obtener recomendaciones de calificaciones para un usuario dado. Se alimenta el usuario de entrada a la RBM y se obtienen las calificaciones recomendadas para las diferentes películas.
- El **método MakeGraph** se encarga de construir el gráfico computacional necesario para el entrenamiento de la RBM. Se realiza un muestreo de Gibbs para obtener muestras aproximadas de las distribuciones de las capas ocultas y visibles. Se iteran entre las capas condicionales, fijando el valor de una variable y muestreando la otra, hasta obtener suficientes muestras [38]. Esto permite estimar las probabilidades y realizar inferencias en la RBM.
- Los **métodos MakeHidden** y **MakeVisible** se utilizan para obtener las capas ocultas y visibles, respectivamente, utilizando las operaciones de la RBM
- El **método estimate** se utiliza para estimar la calificación que un usuario daría a una película en particular. Se verifica si el usuario y la película son conocidos y se devuelve la calificación estimada.

En resumen, esta clase implementa una RBM para generar recomendaciones de calificaciones de películas para usuarios. Utiliza técnicas de aprendizaje automático no supervisado y muestreo de Gibbs para entrenar el modelo y obtener predicciones de calificaciones para usuarios y películas específicas.

El algoritmo de recomendación RBM desarrollado está disponible en el siguiente repositorio: https://github.com/inesgfortis/RecommenderSys/deepLearning.

Las RBM ofrecen varias ventajas en el contexto de los sistemas de recomendación. En primer lugar, al ser modelos generativos no supervisados, pueden aprender automáticamente características latentes de los datos sin requerir etiquetas o información externa. Esto es especialmente útil para descubrir patrones y estructuras ocultas en conjuntos de datos complejos. Otra ventaja es su capacidad para capturar relaciones no lineales entre usuarios y elementos. A diferencia de los modelos lineales, las RBM pueden modelar interacciones complejas, lo que les permite capturar la naturaleza intrincada de las preferencias de los usuarios y las características de los elementos. Además, son capaces de aprender representaciones latentes de alta dimensionalidad, por lo que pueden capturar características complejas y sutiles de los usuarios y elementos que pueden pasar desapercibidas en otras técnicas de recomendación y ofrecer recomendaciones más precisas y personalizadas.

Sin embargo, también presentan algunas limitaciones. El tiempo de entrenamiento puede ser una desventaja, ya que el proceso iterativo requerido para ajustar los pesos y las conexiones entre las capas puede ser computacionalmente costosa y llevar tiempo, especialmente en conjuntos de datos grandes. Por otro lado, la interpretación de las representaciones latentes aprendidas por las RBM puede resultar desafiante. Aunque las RBM son capaces de capturar características relevantes, estas representaciones pueden ser difíciles de interpretar y comprender, lo que puede dificultar la explicación de los factores que influyen en las recomendaciones generadas.

Otro aspecto a considerar es que las RBM pueden requerir grandes cantidades de datos para obtener buenos resultados, dado que en conjuntos de datos pequeños o con información limitada, pueden no ser capaces de capturar adecuadamente las complejidades de las preferencias de los usuarios y los elementos. Por último, la configuración de los parámetros de las RBM puede ser un desafío. Ajustar el número de unidades ocultas y otros hiperparámetros de manera óptima puede requerir un proceso de prueba y error.

En resumen, las RBM ofrecen ventajas como el aprendizaje no supervisado, la captura de relaciones no lineales y la capacidad de aprender representaciones latentes de alta dimensionalidad. Sin embargo, también presentan desventajas relacionadas con el tiempo de entrenamiento, la interpretación de las representaciones latentes, los requisitos de datos y la configuración de parámetros. Cada caso de uso debe evaluar cuidadosamente estas características para determinar si las RBM son la opción adecuada para el sistema de recomendación.

Por último cabe resaltar la posibilidad de mejorar el modelo a través de técnicas de ajuste de hiperparámetros (*hyperparameter tuning*). Esta práctica consiste en explorar diversas configuraciones de los parámetros y la topología del modelo, con el objetivo de lograr una convergencia óptima del algoritmo. Se recomienda experimentar con diferentes valores de parámetros, como el número de nodos ocultos, la tasa de aprendizaje, el tamaño de *batch* y el número de *epochs*, con el fin de identificar aquellos valores que brinden los mejores resultados en términos de desempeño y precisión del modelo.

5.4.2. Autoencoders: AutoRec

Previamente se ha destacado que las Restricted Boltzmann Machines representan uno de los primeros modelos empleados en el ámbito de las redes neuronales aplicadas a los sistemas de recomendación. Sin embargo, a medida que el tiempo ha avanzado, se han desarrollado y adoptado técnicas más contemporáneas. De acuerdo con [16], a partir del año 2015 se empezaron a utilizar redes neuronales más profundas para los sistemas de recomendación, lo cual puede parecer reciente, pero en el contexto actual de la investigación en Inteligencia Artificial, ese período de tiempo constituye una etapa considerablemente amplia.

Un ejemplo de un modelo más contemporáneo en el campo de los sistemas de recomendación es el **AutoRec** [39], una arquitectura innovadora diseñada para el filtrado colaborativo (véase la sección 5.2), desarrollado por un grupo de investigadores de la Universidad Nacional Australiana.

El *AutoRec*, según lo expuesto en [39], consta de tres capas: una capa de entrada, una capa oculta y una capa de salida (ver Figura 5.6).

1. **Capa de entrada**. Se encuentra en la parte inferior de la red y se encarga de recibir y almacenar las valoraciones individuales de los usuarios para los elementos del sistema de recomendación. Cada valoración se representa como una entrada en esta capa.

2. **Capa oculta**. Corresponde con una una capa intermedia entre la capa de entrada y la capa de salida. Su función principal es aprender y extraer características latentes relevantes a partir de las valoraciones de los usuarios. Contiene un número reducido de nodos, lo que ayuda a comprimir y resumir la información relevante del sistema de recomendación

3. **Capa de salida**. Se encuentra en la parte superior de la red y se encarga de generar las predicciones de recomendación para los usuarios. Cada nodo de esta capa representa un elemento del sistema de recomendación, y su valor de salida corresponde a la predicción de recomendación para ese elemento. Utiliza la información aprendida en la capa oculta para generar las predicciones más precisas posible.

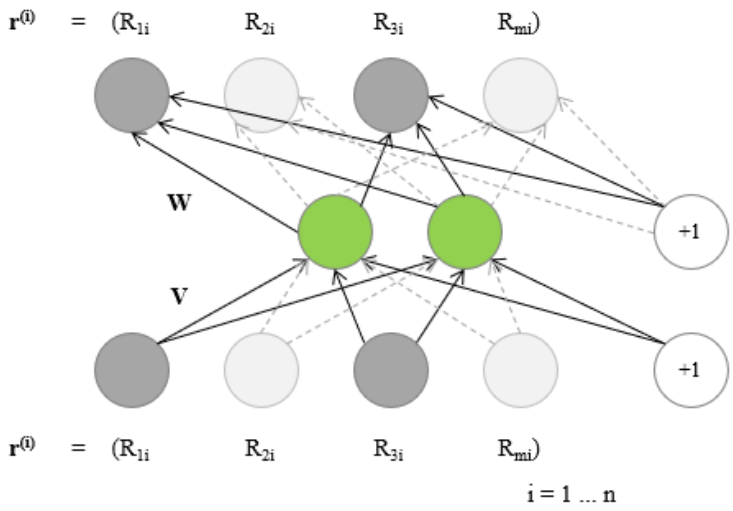


Figura 5.6: Arquitectura del modelo AutoRec basado en contenido

La Figura 5.6 representa la estructura del AutoRec, donde $r^{(i)} = (R_{1i}, \ldots, R_{mi}) \in \mathbb{R}^m$ corresponde con un vector parcialmente observado asociado a un elemento $i \in I = 1, \ldots, n$ y (W, V) representan las transformaciones que experimentan dichos vectores.

En cada instancia de la red, se utiliza una matriz de pesos que representa las conexiones entre las capas. Estos pesos determinan la fuerza y la dirección de la influencia de las neuronas de una capa sobre las neuronas de la capa siguiente. Los valores de los pesos se aprenden y ajustan durante el entrenamiento de la red, de modo que la red pueda capturar patrones y relaciones relevantes en los datos de entrada. Además, en las capas oculta y de salida que se utiliza un parámetro adicional, un sesgo *(bias)*, para introducir un término de compensación en el cálculo de la salida de cada neurona. Este sesgo permite que la red pueda aprender y representar relaciones no lineales en los datos y ajustar el nivel de activación de las neuronas [16, 39].

Esta arquitectura corresponde con la de un autoencoder. Los autoencoders son un tipo de red neuronal que se utilizan para aprender representaciones eficientes de los datos de entrada. Son modelos de aprendizaje no supervisado que constan de una capa de entrada, una o más capas ocultas y una capa de salida que intenta reconstruir los datos originales. La principal característica de los autoencoders es que se busca que la salida sea lo más similar posible a la entrada, lo que implica que la red debe aprender a comprimir la información relevante en las capas ocultas y luego descomprimirla en la capa de salida [40].

Para ello generalmente siguen una estructura simétrica, donde las capas ocultas están dispuestas en forma de un encoder y un decoder. El encoder es responsable de la codificación de los datos de entrada en una representación latente de menor dimensión, mientras que el decoder se encarga de reconstruir los datos originales (proceso de decodificación). Durante la etapa de codificación, los datos de entrada se comprimen a través de las capas ocultas, reduciendo la dimensionalidad de la representación. Luego, en la etapa de decodificación, la red intenta reconstruir los datos originales a partir de la representación latente en las capas ocultas [41].

En el caso del AutoRec, la codificación de la entrada se refiere a la construcción activa de los pesos y sesgos entre la capa de entrada y la capa oculta. Los patrones de entrada se codifican como un conjunto de pesos en la capa oculta. Luego, en la etapa de decodificación, la red intenta reconstruir los datos originales a partir de la representación latente utilizando los pesos entre las capas oculta y de salida [39].

Hay que destacar que esta estructura corresponde con la arquitectura concreta propuesta en el artículo [39]. No obstante, las fases de encoder y decoder pueden estar compuestas por más de una capa de neuronas en función del tamaño del conjunto de datos y la complejidad de estos.

El procedimiento para elaborar el modelo es similar al desarrollado con el algoritmo RBM. En primer lugar, es necesario adaptar el formato de los datos de entrada para que se ajusten a las expectativas del modelo. Con este propósito, se ha creado la función que se muestra en el Listing 5.2. Esta función se encarga de crear una matriz bidimensional llamada trainingMatrix, donde cada fila representa a un usuario y cada columna representa una película.

Listing 5.2: Tratamiento de datos AutoRec

```
def fitData(trainset):

    num_users = trainset.n_users
    num_movies = trainset.n_items

    # 2D matrix: users and movies
    trainingMatrix = np.zeros([num_users, num_movies], dtype=np.float32)

    for (uid, iid, rating) in trainset.all_ratings():
        # Normalize the input ratings (0,1)
        trainingMatrix[int(uid), int(iid)] = rating/5.0

    return trainingMatrix
```

De aucerdo con el Listing 5.2, se obtiene el número total de usuarios y películas presentes en el conjunto de entrenamiento y se inicializa la matriz con ceros. A continuación, se recorren todas las valoraciones en el conjunto de entrenamiento. Durante este recorrido, se normaliza el valor de la calificación para asegurar que esté en el rango adecuado y se asigna a la ubicación correspondiente en la matriz trainingMatrix, utilizando los identificadores del usuario (uid) y la película (iid) como índices. Al normalizar las valoraciones, se asegura que el modelo pueda trabajar de manera eficiente y capturar las relaciones y patrones relevantes en los datos de entrada.

A continuación, se procede a elaborar el algoritmo. Tal y como ocurría en el caso de RBM, *Surprise* [22] no proporciona un modelo AutoRec preentrenado que satisfaga los requerimientos del proyecto por lo que el modelo se elabora desde cero a partir del código desarrollado en el curso *Building Recommender Systems with Machine Learning and AI* [16], adaptándolo posteriormente a los requisitos específicos del proyecto.Este proceso implica la creación de la clase AutoRec, la cual es una subclase personalizada que implementa funcionalidades adicionales basadas en la clase *AlgoBase* de la biblioteca *Surprise*.

La clase AutoRec es responsable de construir y entrenar el modelo desde cero. Utiliza técnicas de redes neuronales y optimización para aprender representaciones latentes de los datos de entrada y generar recomendaciones personalizadas. Sus métodos permiten configurar los parámetros del modelo, realizar el entrenamiento, generar recomendaciones y realizar estimaciones de calificaciones para usuarios y películas específicas. A continuación, se describe brevemente la funcionalidad de cada componente clave de la clase, permitiendo comprender su estructura y operación interna:

- El **método fit** se encarga de entrenar el modelo AutoRec utilizando los datos de entrenamiento proporcionados. Primero, llama al método fit de la clase padre AlgoBase para configurar el conjunto de entrenamiento. Luego, llama al método Train para entrenar el modelo AutoRec y genera las predicciones de calificación para cada usuario y película en el conjunto de entrenamiento.
- El **método Train**, el cual realiza el entrenamiento del modelo AutoRec utilizando el algoritmo de optimización RMSprop. Itera sobre el número de épocas especificadas y realiza el entrenamiento en lotes de datos para mejorar la eficiencia. Durante cada iteración, se llama al método $run\_optimization$ para actualizar los pesos y los sesgos de la red neuronal.
- El **método GetRecommendations** se utiliza para generar recomendaciones para un usuario dado. Toma como entrada un usuario y pasa sus datos por la red neuronal para obtener las predicciones de calificación de las películas.
- El **método initializeWeightsBiases** se encarga de inicializar los pesos y sesgos de la red neuronal AutoRec. Crea variables para los pesos de las capas de codificación y decodificación, y los inicializa aleatoriamente.
- El **método neuralNet** define la estructura de la red neuronal AutoRec. Toma un usuario como entrada y realiza la propagación hacia adelante a través de la red neuronal, aplicando las transformaciones necesarias en las capas ocultas y de salida.

- El **método runOptimization** realiza una iteración de optimización utilizando el algoritmo RMSprop y la retropropagación del gradiente. Calcula la pérdida entre las predicciones del modelo y los datos de entrada, y actualiza los pesos y los sesgos de la red utilizando el optimizador RMSprop.
- El **método estimate** se utiliza para estimar la calificación de un usuario para una película específica. Verifica si el usuario y la película son conocidos por el modelo y devuelve la calificación predicha. Si la calificación predicha es muy baja, se lanza una excepción indicando que no se puede hacer una predicción válida.

Hay que destacar que el código fuente del curso [16] se basa en el artículo [39], pero con algunas modificaciones. En el artículo, se propone un enfoque de filtrado colaborativo basado en contenido, donde la red se entrena una vez por artículo y se alimenta cada usuario para ese artículo en la capa de entrada. Además, se utiliza una función de activación sigmoidea en la salida. En cambio, en el curso [16], se desarrolla un enfoque de filtrado colaborativo basado en usuarios. Otra diferencia es que en el algoritmo desarrollado en el curso, se incorpora el aprendizaje de sesgos para cada nodo, lo cual es una práctica actualmente utilizada para mejorar el rendimiento del modelo.

Es importante mencionar que en el artículo [39], se aborda el problema considerando únicamente las valoraciones observadas. Por lo tanto, se procesa cada paso de la red neuronal de forma individual, propagando únicamente la información de las valoraciones existentes en los datos de entrenamiento. Se omite la contribución de los nodos de entrada correspondientes a los datos faltantes de pares de elementos de usuario que no fueron valorados. No obstante, debido al volumen del conjunto de datos utilizado en el proyecto, no se implementa esta recomendación.

El algoritmo de recomendación AutoRec desarrollado está disponible en el siguiente repositorio: https://github.com/inesgfortis/RecommenderSys/deepLearning.

Una de las ventajas principales es su capacidad para aprender representaciones latentes de los datos sin necesidad de etiquetas o clasificaciones previas, lo que lo hace adecuado para conjuntos de datos sin anotaciones extensas. Por otro lado, tiene la capacidad de comprimir la información relevante en las capas ocultas y luego descomprimirla en la capa de salida, lo que permite capturar las características más importantes de los datos de entrada y generar representaciones más eficientes y compactas. Además, tiene la ventaja de ser mucho más fácil de implementar en un marco moderno como TensorFlow o Keras.

Aunque el modelo ofrece ventajas, también tiene limitaciones. Por un lado, depende de los datos observados para generar recomendaciones por lo que puede tener dificultades para recomendar elementos no valorados previamente por los usuarios en el conjunto de entrenamiento, hecho que puede limitar su capacidad para descubrir nuevos elementos. Además, el entrenamiento y la inferencia pueden ser computacionalmente costosos, lo que dificulta su uso en escenarios con muchos usuarios y elementos. Por otro lado, el modelo puede ser sensible a valores atípicos y ruido en los datos, lo que afecta la calidad de las recomendaciones. También puede tener dificultades para capturar relaciones complejas y sutiles entre usuarios y elementos debido a su estructura lineal.

Por último, cabe destacar que el algoritmo AutoRec desarrollado tiene potencial para mejorar mediante el aumento del conjunto de datos, dado que esto puede ayudar a aumentar la diversidad y la cantidad de datos disponibles. Esto a su vez puede mejorar el rendimiento y la capacidad de generalización del modelo, ya que se expone a una mayor variedad de escenarios y patrones. Además, se pueden aplicar técnicas de ajuste de hiperparámetros para identificar los valores óptimos que maximicen el rendimiento y la precisión del modelo. Estas técnicas exploran diferentes combinaciones de valores para los hiperparámetros y permiten encontrar la configuración más adecuada para el conjunto de datos específico y los requisitos del sistema de recomendación. Al encontrar los valores óptimos, se puede lograr un mejor ajuste y mejorar la calidad de las recomendaciones ofrecidas por el modelo.

5.5. Otras técnicas de recomendación

Hasta ahora, se han explorado una variedad de algoritmos y técnicas utilizados en los sistemas de recomendación para películas. Entre los enfoques que se han investigado se encuentran los algoritmos basados en contenido (sección 5.1), el filtrado colaborativo basado en usuarios (sección 5.2.1) y basado en elementos (sección 5.2.2), así como las técnicas de factorización de matrices como SVD y SVD++ (sección 5.3). También se han examinado el uso de técnicas de Deep Learning, como las redes neuronales restringidas como la RBM (sección 5.4.1), y el sistema de recomendación AutoRec (sección 5.4.2).

Sin embargo, el panorama de los sistemas de recomendación es aún más amplio y dinámico. En esta sección, se exploran una serie de técnicas adicionales que han ganado reconocimiento en el campo de los sistemas de recomendación. Estas técnicas representan enfoques novedosos y efectivos para abordar desafíos específicos en la generación de recomendaciones más precisas y personalizadas. A medida que se avance en esta sección, se analizarán en detalle las siguientes técnicas: 1. Sistemas de recomendación basados en grafos, 2. Modelos de aprendizaje profundo para sistemas de recomendación, 3. Técnicas de procesamiento del lenguaje natural (NLP), y 4. Sistemas de recomendación contextuales.

Cada una de estas técnicas presenta características únicas y ha demostrado su eficacia en diversos contextos de recomendación. Para comprender mejor su funcionamiento y aplicaciones, se explorarán los fundamentos teóricos de cada una de estas técnica con el objetivo de obtener una visión más completa del panorama actual de los sistemas de recomendación.

1. **Sistemas de recomendación basados en grafos**. Estos sistemas representan una técnica que utiliza las estructuras y propiedades de los grafos para modelar las relaciones entre los elementos y usuarios del sistema, generando recomendaciones personalizadas [42]. En este enfoque, los elementos, como usuarios, productos o contenido, se representan como nodos en un grafo, mientras que las relaciones entre ellos se modelan como aristas. Estas relaciones pueden abarcar diferentes aspectos, como la interacción entre usuarios, las calificaciones de los productos o la similitud entre el contenido. Al aprovechar la estructura del grafo, estos sistemas pueden identificar patrones y conexiones relevantes para generar recomendaciones basadas en las preferencias de los usuarios [43]. Un ejemplo destacado de algoritmo utilizado en sistemas basados en grafos es el algoritmo de recomendación Random Walk [44].

Un caso de uso común de los sistemas de recomendación basados en grafos es una plataforma de recomendación de películas, donde se emplea la información de interacción entre usuarios y películas para generar recomendaciones personalizadas. En este contexto, los usuarios y las películas se representarían como nodos en el grafo, y las interacciones, como las calificaciones o visualizaciones, se modelarían como aristas. Para generar recomendaciones relevantes, el sistema podría utilizar algoritmos de propagación de etiquetas o realizar caminatas aleatorias en el grafo para identificar películas similares o comunidades de usuarios con intereses afines. De esta manera, se obtienen recomendaciones que se ajustan a las preferencias individuales de los usuarios.

2. **Modelos de aprendizaje profundo para sistemas de recomendación**. Estas técnicas utilizan modelos de Deep Learning, como redes neuronales, para capturar patrones complejos en los datos de recomendación, permitiendo generar recomendaciones más precisas y contextualizadas, adaptadas a las preferencias y necesidades individuales [45]. Además de las técnicas mencionadas anteriormente, existen otros modelos más avanzados que han ganado popularidad en el campo de los sistemas de recomendación. Estos modelos son capaces de aprender representaciones latentes de usuarios y elementos, descubriendo patrones y relaciones ocultas en los datos de recomendación.

   Un ejemplo destacado para sistemas de recomendación es *Deep Neural Network (DNN)*, utilizado en plataformas como YouTube[1] [46]. Este modelo utiliza múltiples capas ocultas para aprender representaciones no lineales de usuarios y elementos. Mediante la alimentación de los datos de interacción usuario-elemento a la red neuronal, el modelo puede aprender patrones complejos y relaciones no lineales entre ellos, generando recomendaciones personalizadas y precisas.

   Otro enfoque popular es *Factorization Machines (FM)* [47]. Los FM combinan el aprendizaje lineal y el no lineal para modelar las interacciones entre usuarios y elementos. Utilizan técnicas de factorización para capturar las relaciones entre características y generar recomendaciones basadas en esas interacciones.

   Además, existen arquitecturas más avanzadas como los *Recurrent Neural Networks (RNN)* [48] y los *Transformer Networks* [49], que se han aplicado con éxito en sistemas de recomendación secuenciales, como la recomendación de secuencias de películas o música en función del historial de interacciones del usuario.

3. **Técnicas de procesamiento del lenguaje natural (NLP)**. Las técnicas de Procesamiento del Lenguaje Natural (NLP) han ganado popularidad en los sistemas de recomendación debido a su capacidad para comprender y utilizar la información textual de los usuarios y los elementos recomendados [50]. Estas técnicas permiten analizar el contenido textual, como reseñas, descripciones o comentarios, y extraer características relevantes para mejorar la precisión y la personalización de las recomendaciones [51]. En el contexto de los sistemas de recomendación de películas, las técnicas de NLP se utilizan para analizar y comprender el contenido textual asociado a las películas, como sinopsis, reseñas y descripciones. Estos enfoques permiten una comprensión más profunda de las preferencias y necesidades de los usuarios, y se pueden combinar con otros algoritmos para generar recomendaciones más contextualizadas y relevantes [51].

[1] YouTube: `https://www.youtube.com/`

Un ejemplo de caso de uso de las técnicas de NLP en un sistema de recomendación es un servicio de recomendación de películas basado en reseñas de usuarios. El sistema puede utilizar técnicas de procesamiento del lenguaje natural para analizar las reseñas de las películas escritas por los usuarios y extraer características importantes, como los personajes, el estilo de película o las emociones transmitidas. Luego, el sistema puede utilizar estas características para recomendar películas similares a aquellos que recibieron reseñas positivas de los usuarios con gustos similares.

4. **Sistemas de recomendación contextuales**. Estos algoritmos tienen en cuenta el contexto en el que se realiza una recomendación para generar recomendaciones más precisas y relevantes, adaptadas a las circunstancias específicas del usuario en un momento dado. [52]. El contexto se refiere a información adicional sobre el usuario, el elemento a recomendar y el entorno en el que se produce la interacción. Esta información contextual puede incluir características como la ubicación geográfica, el tiempo, el dispositivo utilizado o el estado de ánimo, entre otros.

   La aplicación de los sistemas de recomendación contextuales abarca diversos dominios, como el comercio electrónico, la música, las redes sociales, la salud, entre otros [18]. Estos sistemas han sido objeto de investigación en la comunidad académica y se han propuesto diversos enfoques y algoritmos para su implementación.

   Un ejemplo de un caso de uso de los sistemas de recomendación contextuales es el sistema de recomendación de restaurantes basado en la ubicación y las preferencias del usuario [52]. Este tipo de sistema utiliza el contexto geográfico del usuario para recomendar restaurantes cercanos que se ajusten a sus preferencias culinarias. Además, también puede considerar otros factores contextuales como el horario, el día de la semana y las reseñas de otros usuarios [52].

Por último, es importante destacar que estas técnicas mencionadas son solo algunas de las muchas posibilidades dentro del campo de los sistemas de recomendación para películas. La investigación en este campo sigue evolucionando rápidamente, y nuevos enfoques y técnicas continúan emergiendo para abordar desafíos específicos y mejorar la experiencia de recomendación.

Capítulo 6

Evaluación de resultados

En el ámbito de los sistemas de recomendación de películas, la comparación de resultados desempeña un papel crucial para evaluar y seleccionar el algoritmo más efectivo. A medida que se desarrollan diversos enfoques de recomendación, surge la necesidad de examinar y contrastar sus resultados para determinar cuál ofrece un rendimiento óptimo en términos de precisión, relevancia, cobertura y diversidad.

6.1. Análisis de las métricas de rendimiento

En esta sección, se llevará a cabo una comparación exhaustiva de los resultados obtenidos mediante diferentes algoritmos de recomendación de películas. Se analizarán los diversos enfoques desarrollados, entre ellos, el sistema basado en contenido, los filtros colaborativos basados en usuarios y elementos, las técnicas de factorización de matrices y de Deep Learning. También se incluirán los resultados obtenidos mediante un modelo de recomendación aleatorio (Random).

El objetivo principal de esta comparación es evaluar el desempeño relativo de cada algoritmo utilizando métricas previamente seleccionadas (consultar Capítulo 4), con el fin de obtener una visión integral de sus fortalezas y limitaciones. Se busca identificar las técnicas que destacan en términos de precisión, relevancia, diversidad y novedad en la recomendación de películas. Además, se pretende obtener una visión completa del rendimiento de cada algoritmo y realizar comparaciones significativas entre ellos. Este análisis proporcionará información valiosa para tomar decisiones informadas sobre el algoritmo de recomendación de películas más adecuado en un contexto específico.

A continuación se presentan los resultados obtenidos en cada modelo:

Cuadro 6.1: Métricas de rendimiento de los modelos

Modelo	RMSE	MAE	MAP	MAR	Avg.NDCG	Coverage	User Cov.	Novelty
user-based	0.9030	0.6864	0.7299	0.4667	0.9510	0.0543	0.9967	4231
item-based	0.9172	0.6978	0.7430	0.4480	0.9518	0.0207	1	5854
SVD	0.8802	0.6762	0.7370	0.4480	0.9539	0.0384	0.9180	421
SVD++	0.8688	0.6642	0.7438	0.4495	0.9544	0.0345	0.9082	694
RBM	1.1620	0.9641	0.1630	0.0079	0.9360	0.0028	0	667
AutoRec	2.6319	2.3002	0.5348	0.1272	0.9334	0.0188	1	2701
Random	1.4320	1.1429	0.6299	0.2937	0.9312	0.0297	1	1846

RMSE (Root Mean Squared Error)

Esta métrica evalúa la diferencia entre las calificaciones reales y las predicciones realizadas por el modelo, siendo un indicador de precisión de forma que un valor de RMSE más bajo indica una mayor precisión en las predicciones. Para facilitar la comparación de los diferentes valores de RMSE de los modelos, se ha elaborado la siguiente gráfica.

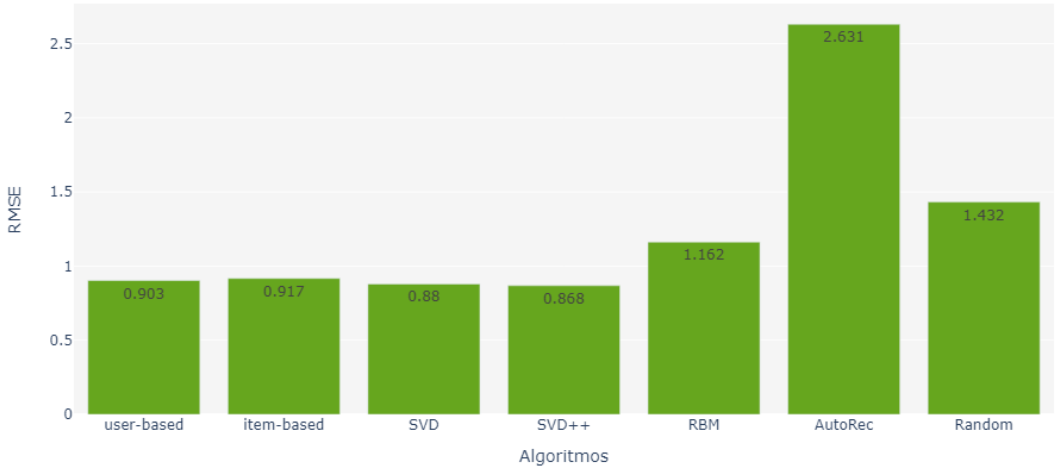


Figura 6.1: Valores de RMSE asociados a cada modelo desarrollado

En la Figura 6.1, se observa que el modelo SVD++ muestra el valor más bajo de RMSE (0.8688), seguido de cerca por el modelo SVD (0.8802). Estos modelos presentan un rendimiento superior en términos de precisión en comparación con los demás algoritmos. Por otro lado, el modelo AutoRec exhibe el valor de RMSE más alto (2.6319), incluso superando al modelo de predicción aleatoria. Esto indica una baja precisión en las predicciones realizadas por el modelo AutoRec. Este valor de RMSE de 2.63 significa que, en promedio, las predicciones del modelo AutoRec difieren del valor real de la valoración de una película en aproximadamente 2.63 puntos en el rango de valores de las valoraciones de las películas.

Existen varios posibles motivos que podrían explicar los malos resultados del modelo AutoRec. En primer lugar, la falta de superposición entre los elementos evaluados por los usuarios en el conjunto de datos utilizado para entrenar y evaluar el modelo puede afectar su rendimiento. Además, es probable que los parámetros del modelo no estén ajustados de manera óptima debido a la falta de ajuste de hiperparámetros, lo que dificulta la captura de las complejas relaciones entre los usuarios y las películas. Además, la complejidad del modelo AutoRec también introduce más variables y posibilidades de error, lo que resulta en predicciones menos precisas.

Por otro lado, es importante destacar que las predicciones del modelo RBM son notablemente inferiores a los valores reales, o determinados como reales. Los valores determinados como reales corresponden a aquellas combinaciones de usuario-elemento que no tienen una valoración establecida, por lo que se calcula la media ponderada de las puntuaciones de valoración utilizando los valores normalizados como pesos y se establece como valor real. De esta forma la diferencia entre los valores reales y predichos en muchos casos no es tan significativa, y por tanto, el RMSE no es tan elevado como en el caso del AutoRec.

MAE (Mean Absolute Error)

Al igual que el RMSE, el MAE mide la diferencia entre las calificaciones reales y las predicciones, pero sin considerar la magnitud de los errores. Nuevamente, el modelo SVD++ muestra el menor MAE (0.6642), seguido por el modelo SVD (0.6762). Estos dos modelos demuestran una mayor precisión en las predicciones en comparación con los otros algoritmos. En contraste, los modelos RBM y AutoRec presentan valores más altos de MAE (0.9641 y 2.3002 respectivamente), lo que indica una mayor discrepancia entre las calificaciones reales y las predicciones realizadas. Por otro lado, los modelos RBM y AutoRec presentan valores más altos de MAE (0.9641 y 2.3002 respectivamente), lo que indica una mayor discrepancia entre las calificaciones reales y las predicciones realizadas.

En particular, el modelo AutoRec muestra nuevamente el valor más alto de MAE (2.3002), y al igual que con el RMSE, este resultado es peor en comparación con el modelo aleatorio, lo que indica una mayor discrepancia entre las calificaciones reales y las predicciones realizadas por el modelo AutoRec. Al calcular el MAE, se toma la diferencia absoluta entre las predicciones y los valores reales, y luego se promedian esas diferencias. Por lo tanto, con un MAE de 2.3002, podemos interpretar que las predicciones del modelo AutoRec tienen un error absoluto promedio de 2.3 puntos en relación con los valores reales.

MAP (Mean Average Precision)

Esta métrica se utiliza para evaluar la calidad de las recomendaciones en términos de precisión y relevancia. En el caso de MAP, el modelo SVD++ muestra el valor más alto (0.7438), seguido por el modelo item-based (0.7430). Estos modelos destacan por su mayor precisión y relevancia en las recomendaciones realizadas. Por otro lado, los modelos RBM y AutoRec presentan los valores más bajos de MAP (0.1630 y 0.5348, respectivamente), lo que indica una menor calidad en las recomendaciones generadas por estos modelos.

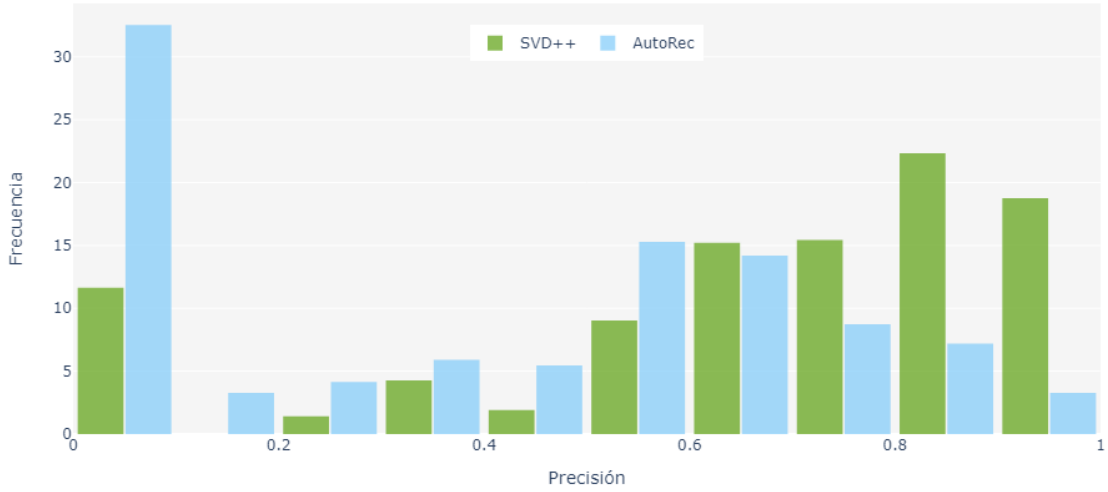


Figura 6.2: Distribución de la precisión de los modelos SVD++ y AutoRec

La Figura 6.2 presenta los histogramas que representan la distribución de la precisión para los modelos SVD++ y AutoRec, considerando un valor de diez recomendaciones por usuario. En el caso del modelo AutoRec, se observa que la mayoría de los usuarios tienen una precisión inferior a 0.2, mientras que el modelo SVD++ muestra frecuencias más altas en valores de precisión más elevados. Estos resultados son consistentes con los valores mostrados en el Cuadro 6.1, los cuales reflejan las medias de precisión obtenidas considerando a todos los usuarios. Este gráfico destaca la superioridad del modelo SVD++ y el desempeño menos satisfactorio del modelo AutoRec en relación con la precisión y relevancia de las recomendaciones generadas.

Puede resultar sorprendente que los modelos más complejos, como RBM y AutoRec, obtengan los peores resultados de precisión. No obstante, estos modelos utilizan técnicas avanzadas de Deep Learning y presentan una arquitectura y funcionamiento más complejos, y a medida que aumenta la complejidad del modelo, también aumenta la posibilidad de errores y desajustes en las predicciones. Es fundamental ajustar correctamente una gran cantidad de parámetros para obtener un rendimiento óptimo, y si no se logra, puede afectar negativamente los resultados. Además, estos modelos requieren una gran cantidad de datos para entrenarse de manera efectiva. Como se ha mencionado anteriormente, el conjunto de datos utilizado presenta una gran cantidad de valores ausentes, lo que puede dificultar que los modelos capturen adecuadamente las características y patrones de los usuarios y las películas. Por ello, las predicciones realizadas por estos modelos pueden ser menos precisas.

Por otro lado, es importante destacar que se utiliza un umbral para determinar las mejores recomendaciones. Este umbral indica la calificación mínima estimada para que una película se considere una de las mejores recomendaciones. En la mayoría de los modelos, este umbral se establece en 3.5. Sin embargo, en el caso del modelo RBM, se ha ajustado a 3.0, ya que ninguna de las predicciones realizadas por este modelo alcanza el valor de 3.5.

En resumen, aunque los modelos RBM y AutoRec son más complejos y utilizan técnicas avanzadas, su desempeño en términos de precisión es inferior en comparación con otros modelos debido a la dificultad de ajustar correctamente sus parámetros y a la limitación en la capacidad de capturar las características y patrones subyacentes en los datos.

MAR (Mean Average Recall)

El MAR mide la capacidad del modelo para recuperar elementos relevantes en las recomendaciones. En la tabla, se observa que el modelo user-based obtiene el valor más alto de MAR (0.4667), seguido por los modelos item-based, SVD y SVD++. Estos modelos demuestran una mejor capacidad para recuperar elementos relevantes en comparación con los otros algoritmos. Por otro lado, los modelos RBM y AutoRec presentan los valores más bajos de MAR (0.0079 y 0.1272, respectivamente), lo que indica una menor capacidad de recuperación de elementos relevantes en sus recomendaciones.

Un aspecto relevante a destacar del Cuadro 6.1 en relación al MAR es que ninguno de los valores de recall alcanza el 50 %. Esto puede deberse al hecho de que el número de recomendaciones seleccionadas para la evaluación de los modelos, que es de 10, es muy bajo en comparación con el número de películas disfrutadas por los usuarios. Para analizar este detalle en profundidad, se ha elaborado la siguiente gráfica.

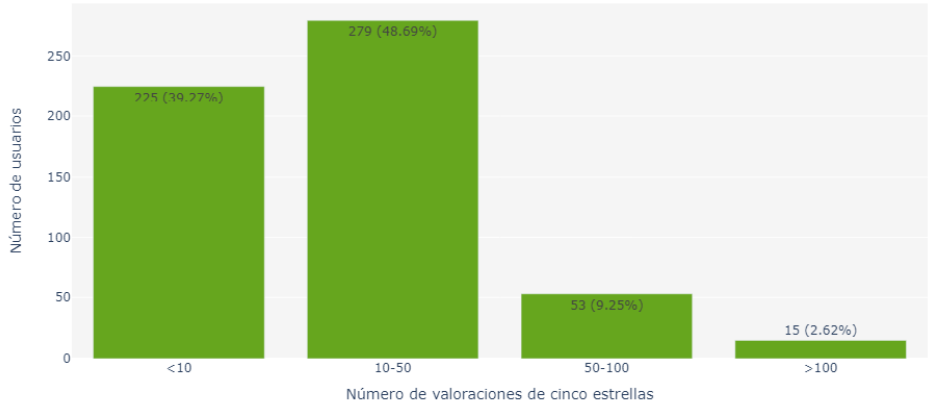


Figura 6.3: Distribución de las valoraciones con cinco estrellas por usuario

La Figura 6.3 muestra la distribución de usuarios en función del número de películas que han valorado con cinco estrellas. Según el gráfico de barras, aproximadamente el 60 % de los usuarios han otorgado una valoración muy positiva a más de diez películas, lo que puede explicar por qué los valores de recall son en general tan bajos. Además, si calculamos la media de películas valoradas con cinco estrellas por usuario, esta es de 23, lo cual es más del doble del número de recomendaciones seleccionadas para los usuarios.

Esta información sugiere que, al utilizar un número limitado de recomendaciones para evaluar los modelos, es posible que se esté subestimando su capacidad de recuperar elementos relevantes. Sería recomendable considerar un mayor número de recomendaciones para una evaluación más precisa y completa de los modelos.

Otro aspecto relevante a tener en cuenta es el umbral utilizado para determinar si una recomendación es considerada o no. Como se mencionó anteriormente, este umbral representa la calificación mínima estimada para que una película sea considerada una de las mejores recomendaciones. Normalmente, se establece en 3.5. Sin embargo, es importante considerar que en el dataset utilizado, la calificación máxima posible para una película es de 5. Por lo tanto, al establecer un umbral inferior al valor máximo, existe la posibilidad de que se estén recomendando películas que no se corresponden con las mejores opciones.

Es fundamental tener en cuenta esta discrepancia entre el umbral establecido y la calificación máxima posible al evaluar la calidad de las recomendaciones, dado que esto puede tener un impacto en la precisión y relevancia de las sugerencias proporcionadas por los modelos. Una recomendación más rigurosa y ajustada podría ser establecer el umbral en el valor máximo, 5, para garantizar que solo se sugieran películas con las calificaciones más altas.

Mean NDCG (Normalized Discounted Cumulative Gain)

El NDCG es una métrica que permite evaluar la calidad de las recomendaciones generadas por un modelo de recomendación considerando tanto la relevancia de los elementos recomendados como la posición en la que se encuentran en la lista de recomendaciones.

En la Cuadro 6.1, se puede observar que el modelo SVD++ obtiene el mayor valor de Mean NDCG, con un resultado de 0.9544. Esto indica que las recomendaciones generadas por este modelo presentan una alta relevancia y se posicionan en lugares favorables en comparación con los demás algoritmos evaluados. De cerca le sigue el modelo SVD, con valor de 0.9539, lo que demuestra también su capacidad para ofrecer recomendaciones de calidad. Por otro lado, los modelos RBM y AutoRec presentan los valores más bajos de Mean NDCG en la tabla, con 0.9360 y 0.9334 respectivamente. Estos resultados indican que las recomendaciones generadas por estos modelos tienen una menor calidad en términos de relevancia y posición en la lista de recomendaciones.

Es importante tener en cuenta que cuanto más cercano a 1 sea el valor del NDCG, mayor será la calidad de las recomendaciones. En general, los valores de NDCG obtenidos en este conjunto de modelos son bastante altos y cercanos a 1, lo que indica que todas las recomendaciones proporcionadas por los modelos tienen un alto grado de relevancia y utilidad para los usuarios. Sin embargo, es importante destacar que el hecho de considerar 10 recomendaciones y un rango de calificación de 0 a 5 puede influir en que estas cifras sean muy cercanas a 1, aunque no necesariamente ideales. Una forma de abordar esto sería hacer el denominador del DCG y el iDCG más exigente, por ejemplo, dividiendo por la posición en lugar de usar el logaritmo de la posición. Esto permitiría una evaluación más rigurosa de la calidad de las recomendaciones y tener una perspectiva más precisa sobre su desempeño.

Coverage

Esta métrica evalúa la proporción de elementos únicos que son recomendados por el sistema de recomendación, es decir, mide la capacidad del modelo para abarcar un amplio espectro de elementos y evitar la sobreexposición a un conjunto reducido de opciones.

De acuerdo con el Cuadro 6.1, el modelo RBM presenta el valor más bajo de Coverage, con un resultado de 0.0028. Esto significa que el modelo recomienda una menor cantidad de elementos únicos en comparación con los otros algoritmos evaluados. Esto puede indicar que el modelo está limitando sus recomendaciones a un conjunto reducido de elementos, lo que puede resultar en una experiencia de usuario menos diversa y con menos descubrimientos de nuevas opciones. Por otro lado, los modelos user-based y SVD muestran una cobertura ligeramente mejor en comparación con los demás algoritmos. Esto indica que estos modelos logran recomendar una mayor variedad de elementos únicos, lo que puede brindar a los usuarios una experiencia más enriquecedora al presentarles diferentes opciones para explorar.

Hay que resaltar que una cobertura baja no necesariamente implica que el modelo sea deficiente, ya que puede estar enfocado en ofrecer recomendaciones altamente personalizadas y ajustadas a los gustos individuales de los usuarios. Sin embargo, es fundamental equilibrar la cobertura con la precisión y la relevancia de las recomendaciones, para garantizar una experiencia de usuario satisfactoria y adecuada a sus necesidades y preferencias.Por lo que la elección del modelo óptimo dependerá de los objetivos y las características específicas del sistema de recomendación, considerando la importancia de equilibrar la cobertura con la precisión y la relevancia de las recomendaciones.

User Coverage

Esta métrica evalúa la proporción de usuarios para los cuales el sistema de recomendación puede generar recomendaciones, es decir, refleja la capacidad del modelo para ofrecer sugerencias personalizadas a un amplio espectro de usuarios.

El Cuadro 6.1 muestra que los modelos item-based, AutoRec y Random, tienen un User Coverage de 1, lo que indica que son capaces de realizar recomendaciones para todos los usuarios presentes en el conjunto de datos, lo que garantiza una cobertura completa en términos de usuarios. Los modelos user-based, SVD y SVD++ también muestran un valor alto, lo que indica que, aunque puede haber una pequeña proporción de usuarios para los cuales no se pueden generar recomendaciones, en general, estos modelos tienen una capacidad de recomendación amplia y cubren la mayoría de los perfiles de usuarios.

Por otro lado, el modelo RBM presenta un User Coverage igual a cero, lo que indica que no puede hacer recomendaciones para todos los usuarios del conjunto de datos. Esto puede deberse a las características particulares del modelo y a las restricciones en su capacidad para generar sugerencias personalizadas. En consecuencia, hay una proporción de usuarios para los cuales no se pueden ofrecer recomendaciones, lo que limita su alcance y utilidad en términos de cobertura de usuarios.

Novelty

Esta métrica se utiliza para medir la novedad de las recomendaciones generadas por el modelo. En el Cuadro 6.1, se observa que el modelo item-based tiene el valor más alto de Novelty (5854), seguido por el modelo user-based (4231). **Un valor alto de novelty indica que las recomendaciones son más inusuales y diferentes a las opciones habituales del usuario**. Esto implica que se presentan recomendaciones menos predecibles, introduciendo al usuario a nuevas experiencias o contenidos que de otra manera podrían haber pasado desapercibidos. Un alto novelty puede ser beneficioso cuando se busca diversificar y ampliar los intereses del usuario, ofreciendo opciones fuera de su zona de confort. Por ejemplo, se entiende que un modelo con valor de novelty igual a 5854 incluye recomendaciones de películas que han sido pocas veces valoradas por los usuarios dado que su ranking promedio se encuentra en torno a la película 5854 más popular de 8000 posibles.

Por otro lado, los modelos SVD, RBM y SVD++ muestran valores más bajos de Novelty, lo que indica que sus recomendaciones se ajustan más a los gustos y preferencias previas del usuario. En estos casos, **las recomendaciones son más predecibles y similares a las opciones que el usuario ya ha mostrado interés**. Un bajo novelty puede ser útil cuando se busca ofrecer recomendaciones más precisas y personalizadas, basadas en las preferencias establecidas del usuario.

Es importante considerar el contexto y los objetivos del sistema de recomendación al evaluar si un modelo es mejor que otro en términos de novedad. Si el objetivo es sorprender y ampliar los horizontes del usuario, un alto novelty puede ser preferible. Sin embargo, si se busca ofrecer recomendaciones altamente relevantes y ajustadas a los gustos individuales, un bajo novelty puede ser más adecuado. La elección óptima dependerá de las necesidades y preferencias específicas de los usuarios y del propósito del sistema de recomendación.

En base a los resultados obtenidos en las métricas offline seleccionadas, se puede concluir que el **modelo SVD++ muestra el mejor rendimiento general en comparación con los otros algoritmos**. Obtiene buenos resultados en términos de precisión (RMSE, MAE), calidad de las recomendaciones (MAP, Mean NDCG), capacidad de recuperación de elementos relevantes (MAR) y User Coverage. Sin embargo, es importante tener en cuenta que estos resultados se basan en métricas offline y pueden no reflejar completamente el rendimiento en un entorno en vivo. Por lo tanto, es recomendable realizar pruebas adicionales y considerar otros factores antes de seleccionar el modelo más adecuado para un sistema de recomendación de películas.

Por otro lado, es importante recordar que también se ha desarrollado un algoritmo basado en contenido que se centra únicamente en los atributos de las películas, en este caso los géneros, sin considerar las calificaciones de los usuarios. Debido a su naturaleza y características, la evaluación de este algoritmo difiere de la de los demás modelos y, por lo tanto, no se ha incluido en la tabla de métricas del estudio (Cuadro 6.1). La diferencia con el resto de algoritmos es que este no tiene como objetivo predecir las calificaciones de un usuario ni realizar recomendaciones basadas en los gustos de un usuario específico, sino que se enfoca en proporcionar recomendaciones en función de una película determinada. Por ejemplo, en el cuadro 6.2 se muestran las mejores 15 sugerencias para la película *"Toy Story"* obtenidas mediante el modelo desarrollado en la sección 5.1.

Cuadro 6.2: Recomendaciones basadas en la película Toy Story

Recomendaciones
1. Antz
2. Toy Story 2
3. The Adventures of Rocky and Bullwinkle
4. The Emperor's New Groove
5. Monsters, Inc.
6. The Wild, The
7. Shrek the Third
8. The Tale of Despereaux
9. Asterix and the Vikings
10. Turbo
11. The Good Dinosaur
12. Moana
13. Space Jam
14. Shrek
15. The Twelve Tasks of Asterix

Para evaluar la calidad de estas recomendaciones, se ha creado una métrica de popularidad adaptada a este caso particular. Se consideran películas populares aquellas que han obtenido altas valoraciones por parte de los usuarios que también disfrutaron de *"Toy Story"*, es decir, aquellos que le otorgaron una calificación igual o superior a 4.

Para seleccionar las películas populares se ha definido la función *computePopularity*, encargada de evaluar la popularidad de las películas en función de las calificaciones de los usuarios y compararlas con un conjunto de recomendaciones. Esta función busca identificar las películas que han sido bien valoradas por los usuarios que también han dado una alta calificación a la película de referencia.

Esto permite determinar qué películas tienen una mayor probabilidad de ser atractivas para los usuarios que disfrutaron de la película de referencia. Además, la función proporciona una puntuación que indica el porcentaje de películas populares presentes en las recomendaciones, lo que permite evaluar la capacidad del sistema de recomendación para sugerir películas relevantes y populares. En resumen, la función busca identificar películas populares y las compara con las recomendaciones para evaluar la efectividad del sistema en proporcionar sugerencias atractivas y alineadas con los gustos de los usuarios. Además, permite diferenciar entre películas populares sin tener en cuenta los géneros de las películas o considerar únicamente aquellas que tienen al menos un género en común con la película de referencia *"Toy Story"*.

En la Figura 6.4, se presentan las películas populares según los dos posibles criterios: considerando todas las películas independientemente del género y considerando solo aquellas que comparten al menos un género con Toy Story.

Independientemente del género	**Al menos un género compartido**
1. The Shawshank Redemption	1. Star Wars: Episode IV - A New Hope
2. Forrest Gump	2. Star Wars: Episode V - The Empire Strikes Back
3. Star Wars: Episode IV - A New Hope	3. Jurassic Park
4. Pulp Fiction	4. Indiana Jones and the Raiders of the Lost Ark
5. The Silence of the Lambs	5. Star Wars: Episode VI - Return of the Jedi
6. Star Wars: Episode V - The Empire Strikes Back	6. The Lion King
7. The Matrix	7. Aladdin
8. Jurassic Park	8. Back to the Future
9. Indiana Jones and the Raiders of the Lost Ark	9. Shrek
10. Star Wars: Episode VI - Return of the Jedi	10. Apollo 13
11. The Sixth Sense	11. The Lord of the Rings: The Return of the King
12. Schindler's List	12. The Lord of the Rings: The Fellowship of the Ring
13. Saving Private Ryan	13. The Princess Bride
14. The Usual Suspects	14. Monty Python and the Holy Grail
15. The Lion King	15. Finding Nemo

Figura 6.4: Películas populares entre los usuarios que han disfrutado *Toy Story*

La principal diferencia entre las películas presentes en las listas de la Figura 6.4 radica en el criterio de selección. En la primera lista, las películas se seleccionaron independientemente del género, lo que significa que no se tuvo en cuenta el tipo de película al elegirlas. Por otro lado, en la segunda lista, las películas se seleccionaron considerando el criterio de género, lo que implica que todas las películas incluidas comparten un género similar al de la película de referencia. Por lo tanto, las películas de la segunda lista son más específicas en términos de género y podrían tener una temática o estilo similar a "Toy Story", mientras que las películas de la primera lista son más diversas en cuanto a género.

Al comparar la lista de recomendaciones con las películas populares, podemos observar que en el caso de la lista de la izquierda no hay coincidencias, mientras que si consideramos únicamente películas que comparten el género, encontramos una coincidencia: *"Shrek"*. Estos resultados no son sorprendentes, ya que el hecho de considerar únicamente los géneros de las películas para hacer recomendaciones las hace muy básicas y predecibles. Además, es importante tener en cuenta que, de todas las películas disponibles, solo se han seleccionado 15 recomendaciones, lo cual representa una parte muy pequeña del conjunto de opciones posibles, por lo que no es de extrañar que haya muy pocas coincidencias. Por último, es relevante destacar que para obtener las películas populares se ha tenido en cuenta la opinión de los usuarios, la cual está influenciada por muchos más parámetros aparte del género de la película.

En definitiva, el algoritmo de recomendación basado en contenido, que permite relacionar películas únicamente en función de sus géneros, supone un modelo muy sencillo y predecible. Este tipo de algoritmo analiza las características de una película en particular, como su género, y luego busca otras películas con géneros similares para recomendar. Si dos películas tienen géneros en común, es probable que el algoritmo las considere relevantes y las recomiende.

Esto puede ser útil en algunos casos, ya que los usuarios que disfrutan de un género específico pueden encontrar películas similares que les gusten. Sin embargo, este enfoque limitado no tiene en cuenta otros aspectos importantes de las películas, como la trama, los actores o la dirección, que pueden ser determinantes en las preferencias de los usuarios. Además, este algoritmo no puede capturar la complejidad de las preferencias individuales, ya que se basa únicamente en criterios de género. Por lo tanto, aunque este enfoque pueda proporcionar recomendaciones iniciales, es probable que sea menos efectivo en comparación con otros algoritmos más sofisticados que tienen en cuenta una variedad de factores y características de las películas, así como las preferencias individuales de los usuarios.

6.2. Análisis de las recomendaciones

El enfoque anterior se ha centrado exclusivamente en las métricas obtenidas de los algoritmos de recomendación. Sin embargo, es importante tener en cuenta que los resultados de estas métricas offline no reflejan necesariamente el rendimiento en un entorno en vivo. Por lo tanto, es recomendable realizar pruebas adicionales y considerar otros factores al seleccionar el modelo más adecuado para un sistema de recomendación de películas.

Un enfoque interesante para evaluar la efectividad de los diferentes algoritmos de recomendación es considerar la diversidad en los gustos y experiencias de los usuarios. Con este propósito, se ha llevado a cabo un experimento utilizando dos usuarios como casos de estudio. El objetivo principal es observar cómo las recomendaciones varían según los gustos y experiencias de cada usuario, y analizar la capacidad de los algoritmos para adaptarse y proporcionar recomendaciones relevantes y atractivas para cada perfil. Esta evaluación nos ayudará a comprender cómo diferentes enfoques de recomendación se desempeñan en escenarios con usuarios diversos, lo que contribuirá a mejorar nuestro conocimiento sobre la efectividad de los sistemas de recomendación en la generación de recomendaciones personalizadas y contextualizadas.

Para lograr esto, se han seleccionado dos usuarios con perfiles y preferencias diferentes, lo que implica que tienen gustos y experiencias distintas en el contexto del sistema de recomendación. Es importante destacar que ninguno de las 232 películas valoradas por el primer usuario coincide con alguna de las 216 valoradas por el segundo usuario, lo que significa que sus preferencias también difieren. En la figura 6.5 se muestran las películas mejor y peor valoradas por el primer usuario, mientras que en la figura 6.6 se muestran las películas asociadas al segundo usuario.

Películas mejor valoradas	
Película	Rating
MASH	5
Excalibur	5
Indiana Jones and the Last Crusade	5
Pink Floyd: The Wall	5
From Russia with Love	5
Goldfinger	5
The Dirty Dozen	5
Gulliver's Travels	5
American Beauty	5
South Park: Bigger, Longer and Uncut	5

Películas peor valoradas	
Película	Rating
The Talented Mr. Ripley	1
Psycho	2
I Still Know What You Did Last Summer	2
Toys	2
The Mummy	2
Encino Man	3
Independence Day	3
Escape to Witch Mountain	3
Pete's Dragon	3
Batman Returns	3

Figura 6.5: Preferencias del primer usuario

Películas mejor valoradas	
Película	Rating
No Man's Land	5
The Philadelphia Story	5
12 Angry Men	5
The Outlaw Josey Wales	5
The Princess Bride	5
Star Wars: Episode V	5
Strictly Ballroom	5
Election	5
Rebel Without a Cause	5
Dial M for Murder	5

Películas peor valoradas	
Película	Rating
Galaxy Quest	1
Mars Attacks!	1
The Piano	1
Wings of Desire	1
Hook	1
The Matrix	1
The English Patient	1
Chocolat	1
Sleepless in Seattle	1
Mr. Mom	1

Figura 6.6: Preferencias del segundo usuario

Analizando las figuras 6.5 y 6.6, podemos observar que el primer usuario muestra una preferencia por películas de acción y aventuras, como *"MASH", "Excalibur"e "Indiana Jones and the Last Crusade"*. También muestra interés en películas clásicas como *"From Russia with Love"y "Goldfinger"*. En general, otorga una calificación máxima de 5 a todas las películas, lo que indica una tendencia a valorar positivamente la mayoría de las películas que ha visto. Por otro lado, el segundo usuario tiene gustos más variados y muestra preferencia por películas clásicas y aclamadas, como *"The Philadelphia Story", "12 Angry Menz Rebel Without a Cause"*. También se inclina por películas de géneros diversos, como *"Star Wars: Episode Vz "Strictly Ballroom"*. En cuanto a las películas peor valoradas, el segundo usuario ha otorgado una calificación mínima de 1 a todas ellas, lo que indica una tendencia a ser más crítico o exigente en sus evaluaciones.

Para obtener las películas mejor y peor valoradas por cada usuario, se utiliza una función que filtra las clasificaciones de películas del usuario, combina los datos con información adicional de las películas y organiza los datos en el formato deseado. Luego, se ordenan las películas según su clasificación y se seleccionan las k películas con las calificaciones más altas y más bajas, creando dos DataFrames separados. Esta función proporciona una visión rápida de las preferencias del usuario en términos de sus películas mejor y peor valoradas.

A continuación, se llevará a cabo un análisis de las recomendaciones generadas por dos de los algoritmos desarrollados: SVD++, que ha demostrado un rendimiento destacado en general, y RBM, a pesar de que sus resultados no han cumplido las expectativas debido a su arquitectura compleja.

SVD++
1. The Shawshank Redemption
2. The Lord of the Rings: The Return of the King
3. Snatch
4. The Lord of the Rings: The Fellowship of the Ring
5. The Departed
6. Fight Club
7. Amelie
8. Rear Window
9. Ran
10.The Seventh Seal

RBM
1. The Boat
2. Whiplash
3. The Maltese Falcon
4. Ghost in the Shell
5. In Bruges
6. Manhattan
7. Howl's Moving Castle
8. Cool Hand Luke
9. On the Waterfront
10.The Martian

Figura 6.7: Recomendaciones de los modelos para el primer usuario

Si revisamos las películas recomendadas por el modelo SVD++, encontramos títulos como *"The Shawshank Redemption", "The Lord of the Rings: The Return of the King", "Fight Club"y "Ran"*. Además, estas películas son ampliamente reconocidas por su calidad cinematográfica y su popularidad entre los amantes del cine. Por otro lado, el modelo RBM recomienda películas como *"The Boat", "Whiplash"y "The Maltese Falcon"*. Estas películas no se ajustan tanto a las preferencias del usuario en términos de género (acción y aventuras) ni a su interés en películas clásicas. Sin embargo, algunas de estas películas podrían ser valoradas positivamente por el usuario debido a su calidad cinematográfica. Esto era de esperar dado que anteriormente se ha comprobado mediante las métricas de evaluación que el rendimiento del modelo RBM era de los más bajos por lo que se espera que las recomendaciones generadas pueden ser menos precisas y adecuadas para el usuario en comparación con el modelo SVD++.

Repetimos este análisis para el segundo usuario seleccionado.

SVD++
1. The Godfather
2. Schindler's List
3. Kill Bill: Vol. 1
4. Pulp Fiction
5. Apocalypse Now
6. Chinatown
7. The Godfather: Part II
8. Wallace & Gromit: The Wrong Trousers
9. A Clockwork Orange
10.Dr. Strangelove

RBM
1. The Boat
2. The Maltese Falcon
3. Whiplash
4. Ghost in the Shell
5. Manhattan
6. In Bruges
7. Howl's Moving Castle
8. Cool Hand Luke
9. On the Waterfront
10.Moon

Figura 6.8: Recomendaciones de los modelos para el segundo usuario

Al examinar las películas recomendadas por el modelo SVD++, para el segundo usuario se observa que se sugieren títulos ampliamente reconocidos, como *"The Godfather", "Schindler's List", "Pulp Fiction"y "A Clockwork Orange"*. Estas películas destacan tanto por su calidad cinematográfica como por su relevancia histórica en la industria del cine. Además, algunas de ellas se ajustan a las preferencias del usuario en cuanto a películas clásicas y populares. También se incluyen películas como *"Kill Bill: Vol. 1"y "Wallace & Gromit: The Wrong Trousers"*, que ofrecen variedad en términos de género y estilo.

En cambio, entre las recomendaciones del modelo RBM aparecen películas como *"The Boat", "Whiplash"y "In Bruges"*. Estas películas, aunque no se ajustan exactamente a las preferencias del usuario en cuanto a películas clásicas, aún podrían ser consideradas interesantes y de calidad cinematográfica. Sin embargo, hay menos coincidencia en términos de género, ya que se sugieren películas como *"Ghost in the Shell"y "Howl's Moving Castle"*, que pueden tener un enfoque más orientado hacia la ciencia ficción y la animación.

En resumen, al analizar las recomendaciones para el segundo usuario, se observa que el modelo SVD++ ofrece recomendaciones más acertadas y alineadas con sus preferencias. Estas recomendaciones abarcan películas clásicas, aclamadas y géneros diversos, demostrando una comprensión precisa de sus gustos. En contraste, el modelo RBM, aunque ofrece algunas películas interesantes, no se adapta tan adecuadamente a las preferencias del usuario en términos de género y películas clásicas. Esto respalda la efectividad del modelo SVD++ al generar recomendaciones más ajustadas, respaldando su rendimiento superior en las métricas de evaluación.

El análisis de las recomendaciones realizadas por el modelo RBM para ambos usuarios revela ciertas limitaciones y conclusiones importantes. **Es interesante observar que, a excepción de una película, el modelo ha sugerido la misma lista de películas para ambos usuarios**. Este hallazgo plantea cuestiones sobre el rendimiento y la capacidad del modelo para adaptarse a las preferencias individuales de los usuarios.

Una de las limitaciones destacadas del modelo RBM es su falta de personalización. Parece tener dificultades para capturar las preferencias únicas de cada usuario, lo que resulta en recomendaciones similares para perfiles diferentes. Esto indica que el modelo puede tener dificultades para capturar la diversidad de gustos y preferencias de los usuarios, lo que limita su capacidad de proporcionar recomendaciones más precisas y adaptadas.

Además, la similitud en las recomendaciones también plantea interrogantes sobre la capacidad del modelo para capturar sutilezas en los gustos de los usuarios. Al ser una arquitectura más compleja, el modelo RBM puede estar más enfocado en patrones generales y amplios en los datos, lo que puede dificultar la detección de preferencias más específicas o matices en los gustos de los usuarios.

Otra conclusión importante es que el modelo RBM puede enfrentar dificultades al adaptarse a usuarios con gustos variados. En el caso del segundo usuario, cuyas preferencias abarcan diferentes géneros y películas clásicas, el modelo RBM parece tener dificultades para satisfacer todas esas preferencias de manera adecuada. Como resultado, las recomendaciones generadas carecen de diversidad y personalización.

En contraste, el modelo SVD++ ha demostrado un mejor rendimiento en las métricas y ha generado recomendaciones más acertadas para ambos usuarios. Esto indica que el modelo es más efectivo en la captura de las preferencias individuales y en la generación de recomendaciones más personalizadas y adaptadas.

En definitiva, al analizar las recomendaciones y compararlas con las métricas de evaluación, **se observa una correspondencia entre el rendimiento de los modelos en las métricas y la adecuación de las recomendaciones a las preferencias del usuario**. El modelo que muestra un mejor rendimiento en las métricas (SVD++) también genera recomendaciones más acertadas y alineadas con las preferencias del usuario, mientras que el modelo con un desempeño inferior en las métricas (RBM) ofrece recomendaciones menos adecuadas en este caso. Esto respalda la conexión entre las métricas de evaluación y la calidad de las recomendaciones proporcionadas por los modelos.

Capítulo 7

Sistemas recomendadores en entornos productivos

A medida que el campo de los sistemas recomendadores ha evolucionado, se ha vuelto cada vez más importante no solo desarrollar y evaluar modelos de recomendación eficaces, sino también implementarlos en entornos productivos para obtener resultados prácticos y beneficios reales. Un entorno de producción se refiere al contexto en el cual un sistema recomendador está en funcionamiento y se utiliza para ofrecer recomendaciones a los usuarios en tiempo real. A diferencia de las evaluaciones offline, donde se analizan conjuntos de datos históricos y se calculan métricas para medir el rendimiento del modelo, los entornos de producción son entornos en vivo donde los usuarios interactúan con el sistema en tiempo real. Esto plantea una serie de desafíos únicos que no se abordan completamente mediante la evaluación con métricas offline.

Entonces, ¿por qué no es suficiente con evaluar los modelos con métricas offline? La razón principal radica en la brecha que existe entre la evaluación en entornos controlados y la aplicación en situaciones reales. En los entornos offline, se asume que los datos históricos disponibles son un reflejo preciso de las preferencias de los usuarios y las interacciones futuras. Sin embargo, esto no siempre es cierto, ya que los patrones y las preferencias de los usuarios pueden cambiar con el tiempo, y los modelos de recomendación necesitan adaptarse a estas dinámicas. Al evaluar los modelos de recomendación únicamente con métricas offline, podemos obtener una medida de su rendimiento en retrospectiva, pero no necesariamente reflejan cómo se comportarán en un entorno de producción [53]. Los sistemas recomendadores en entornos productivos deben enfrentar desafíos en tiempo real, como la escalabilidad para manejar grandes volúmenes de datos, la necesidad de actualizaciones continuas para adaptarse a las preferencias cambiantes de los usuarios, y la capacidad de proporcionar recomendaciones precisas y relevantes en tiempo real.

En este capítulo, nos centraremos en el siguiente paso después de la fase de desarrollo y selección del modelo SVD++. Dado que este algoritmo ha demostrado un mejor rendimiento general, exploraremos cómo abordar los desafíos asociados con su implementación en entornos productivos. Una vez que el modelo ha sido entrenado y está listo para realizar inferencias y proporcionar recomendaciones, analizaremos estrategias clave para implementarlo y desplegarlo de manera efectiva en producción. Consideraremos aspectos críticos como la escalabilidad del sistema, la gestión de datos en tiempo real y las técnicas de inferencia eficientes, que permiten ofrecer recomendaciones rápidas y precisas.

El principal desafío al implementar sistemas recomendadores en entornos productivos radica en la constante entrada de nuevos usuarios y nuevos artículos en el sistema. Este fenómeno se conoce como arranque en frío, el cual se ha descrito previamente (Sección 2.2).

Cuando un nuevo usuario ingresa al sistema, no existen valoraciones asociadas a ninguno de los artículos cubiertos por el sistema de recomendación, lo que significa que no hay vínculos usuario-artículo disponibles para ese usuario en particular [1]. Para abordar las inferencias sobre usuarios en estas situaciones de arranque en frío, existen dos soluciones comunes:

1. **Solicitar información al usuario**. Una estrategia consiste en solicitar al nuevo usuario información acerca de sus gustos en películas. Esto puede incluir preguntar sobre su género favorito, así como pedirle que indique una serie de películas que haya disfrutado mucho y otras que no haya disfrutado tanto. Al recolectar esta información desde el momento en que el usuario se registra en el sistema, se puede comenzar a construir un perfil inicial que permita realizar recomendaciones más personalizadas.

2. **Recomendar películas populares**. Otra opción es recomendar películas populares que hayan sido bien recibidas por la mayoría de los usuarios. En ausencia de información específica del nuevo usuario, esta estrategia se basa en la idea de que las películas populares suelen tener una amplia aceptación y es más probable que generen una experiencia satisfactoria para un amplio espectro de usuarios. Sin embargo, es importante tener en cuenta que esta aproximación puede generar recomendaciones menos personalizadas.

Ambas estrategias presentan ventajas y desventajas. La primera opción permite recopilar información personalizada y brindar recomendaciones más adaptadas a los gustos individuales del usuario. Sin embargo, puede requerir una mayor interacción inicial por parte del usuario y puede no ser factible si el usuario es reticente a proporcionar información personal. La segunda opción es más sencilla de implementar, pero puede no ser tan precisa en términos de adaptación a los gustos individuales.

Además del escenario de nuevos usuarios, también puede presentarse el caso en el que un usuario registrado en la plataforma desee recibir recomendaciones actualizadas. En este caso, basta con generar nuevas recomendaciones para ese usuario en particular. Por ejemplo, si las diez primeras recomendaciones no son suficientes, se pueden mostrar las diez siguientes y así sucesivamente, hasta satisfacer las necesidades del usuario.

Por otro lado, cuando se introduce un elemento nuevo en el sistema, no existen valoraciones de usuarios ni información histórica sobre este artículo por lo que se crea nuevamente un vacío de datos que dificulta la capacidad del sistema para realizar recomendaciones precisas y personalizadas para ese elemento en particular. Cuando el sistema no tiene información suficiente para comprender las preferencias y los gustos de los usuarios en relación con el nuevo artículo las recomendaciones pueden perder calidad ya que sin datos de retroalimentación de los usuarios, el sistema no puede basar sus recomendaciones en patrones de comportamiento pasados, lo que limita su capacidad para comprender las preferencias individuales y generar recomendaciones relevantes.

Para abordar este desafío, los sistemas recomendadores emplean diversas estrategias. Una de ellas consiste en desarrollar un algoritmo content-based que permita utilizar las características del nuevo artículo, para inferir su posible relevancia y asociaciones con otros elementos para detectar posibles potenciales usuarios a los que les pueda gustar este nuevo ítem [1]. Para el caso concreto del conjunto de datos movieLens se podrían caracterizar a los usuarios por sus tres géneros favoritos en función de las valoraciones, y recomendar la nueva película a aquellos que tengan entre sus favoritos alguno de los géneros asociados al nuevo elemento.

Es crucial que se almacene toda la información relevante, incluyendo el feedback de los usuarios, tanto para usuarios y elementos existentes como para nuevos, con el fin de mantener un sistema de recomendación actualizado y adaptado a las preferencias cambiantes de los usuarios. Esta información debe ser guardada en una base de datos junto con los datos de entrenamiento del modelo. Posteriormente, estos datos se utilizan para construir iteraciones del modelo, lo que permite la incorporación de nuevas interacciones y actualizaciones en la recomendación.

Una vez que se ha almacenado toda la información relevante, podemos realizar varias evaluaciones para medir el rendimiento del sistema recomendador en producción. Por un lado, se puede comparar el rating de las recomendaciones generadas con el rating real proporcionado por los usuarios (utilizando, por ejemplo, la métrica de error cuadrático medio, RMSE Online). Al mismo tiempo, podemos utilizar estos datos para calcular métricas de precisión, como precisión@k y recall@k, que nos ayudan a evaluar qué tan bien el sistema recomienda los elementos relevantes en los primeros k elementos.

Estas métricas online nos proporcionan una visión del rendimiento actual del modelo en producción y nos permiten analizar si ha habido una degradación en su desempeño a lo largo del tiempo. Aquí es donde entra en juego el ciclo de vida de un sistema recomendador, que se refiere al período de tiempo en el que el modelo puede mantenerse en producción sin experimentar una degradación significativa en las métricas de rendimiento.

Cuando un modelo comienza a mostrar signos de degradación en su rendimiento, es necesario considerar su reentrenamiento. Hay dos motivos principales que pueden desencadenar la necesidad de reentrenar el modelo:

1. El modelo está teniendo un rendimiento deficiente en producción en comparación con las métricas online establecidas. Esto significa que las predicciones del modelo se están alejando significativamente de los ratings reales proporcionados por los usuarios en el feedback.

2. Existe un gran volumen de datos nuevos, como nuevos usuarios o películas, que indican la necesidad de una versión actualizada del modelo. En este caso, se puede implementar un proceso periódico y automatizado de reentrenamiento del modelo, siempre que se conozca aproximadamente la cantidad de nuevos datos generados.

Ante esta situación se puede realizar una actualización del modelo, por ejemplo, entrenándolo con datos adicionales. Además de la actualización del modelo, es fundamental mantener la trazabilidad de las versiones anteriores. Esto implica realizar una versión del modelo actualizado y conservar las versiones anteriores en registros o archivos específicos.

La versión anterior del modelo es útil para comparaciones y análisis posteriores, y permite realizar un seguimiento de los cambios y mejoras realizados en el sistema recomendador a lo largo del tiempo.

También se puede probar una técnica diferente, por ejemplo, cambiar de un sistema en producción como el SVD++ a un AutoEncoder se debe realizar un test A/B. Estos métodos denominados pruebas A/B miden el impacto directo del sistema de recomendación en el usuario final [1].

La idea básica de estos métodos es comparar dos algoritmos de la siguiente manera. En primer lugar se divide a los usuarios en dos grupos diferentes, A y B. A continuación se asigna un algoritmo para el grupo A (SVD++) y otro algoritmo para el grupo B (AutoEncoder) durante un periodo de tiempo, manteniendo el resto de condiciones, como el proceso de selección de usuarios, en los dos grupos lo más similares posible. Al final del proceso, se compara el rendimiento y la satisfacción global de los algoritmos de los dos grupos.

Para poner en práctica y comprobar todo lo que se ha comentado acerca de los entornos de producción y la evaluación online de algoritmos de recomendación, **se ha desarrollado un Dash interactivo**. Este Dash tiene como objetivo simular un entorno vivo en el cual los usuarios pueden experimentar la funcionalidad del sistema recomendador y recibir recomendaciones personalizadas en tiempo real.

El desarrollo de este Dash proporciona herramienta de visualización de las recomendaciones versátil y amigable que permite a los usuarios identificarse y acceder al sistema de recomendación. A través de esta interfaz, los usuarios pueden interactuar con el algoritmo SVD++ y recibir recomendaciones basadas en sus preferencias y comportamientos. También recopila datos en tiempo real sobre las interacciones de los usuarios. Estos datos son esenciales para evaluar y ajustar continuamente el desempeño del algoritmo, así como para mejorar la calidad de las recomendaciones ofrecidas. Además, con él se pueden realizar comprobaciones y pruebas exhaustivas del algoritmo en un entorno controlado ya que los usuarios pueden proporcionar retroalimentación instantánea sobre la relevancia y utilidad de las recomendaciones recibidas, lo que permite realizar ajustes y mejoras en tiempo real.

Este dashboard simula una aplicación de recomendación de películas altamente personalizada. Está diseñada para brindar a los usuarios una experiencia única al proporcionar recomendaciones precisas y relevantes basadas en sus gustos cinematográficos. La aplicación utiliza el conjunto de datos pequeño de MovieLens, que contiene información detallada sobre las películas y las valoraciones de los usuarios.

El funcionamiento de la aplicación se centra en el inicio de sesión de los usuarios. Aquellos que ya están registrados pueden acceder directamente a un menú personalizado que muestra las 10 mejores recomendaciones de películas específicas para ellos. Estas recomendaciones se generan utilizando un algoritmo de filtrado colaborativo avanzado conocido como SVD++ desarrollado en la sección 5.3, dado que este algoritmo ha demostrado el mejor rendimiento al predecir las preferencias de los usuarios y sugerir películas que se adaptan a sus gustos individuales. Es importante destacar que se ha fijado el número de recomendaciones en 10 para optimizar los tiempos de ejecución de la aplicación y proporcionar resultados rápidos a los usuarios.

Por otro lado, para los nuevos usuarios que aún no están registrados en la aplicación, se ha implementado un proceso adicional. En lugar de acceder directamente al menú de recomendaciones, se les redirige a una encuesta interactiva que les permite identificar y registrar sus preferencias cinematográficas. Durante esta encuesta, se presentan una serie de películas y se les pide a los usuarios que las valoren según su agrado. Este proceso tiene como objetivo comprender los gustos y preferencias únicos de cada nuevo usuario.

Una vez que el nuevo usuario ha completado la encuesta y ha identificado sus gustos cinematográficos, el sistema de recomendación utiliza estos datos para generar recomendaciones personalizadas. Basándose en los gustos similares de otros usuarios registrados en la aplicación, se busca a aquellos que comparten intereses y preferencias cinematográficas más cercanas. Para determinar la similitud entre usuarios, se calcula una distancia basada en las valoraciones de las películas. Cuanto menor sea la distancia, más similares serán los gustos entre usuarios. A partir de las películas valoradas positivamente por estos usuarios similares, se seleccionan las recomendaciones más relevantes y se muestran al nuevo usuario. Si no se encuentra ningún usuario lo suficientemente similar al nuevo usuario, se utilizan las películas más populares entre todos los usuarios registrados para generar recomendaciones iniciales. Esto asegura que el nuevo usuario reciba al menos algunas recomendaciones relevantes y pueda comenzar a explorar el contenido de la aplicación.

La aplicación también ofrece la opción de explorar y modificar las preferencias en tiempo real. Además de las recomendaciones, el menú muestra una sección donde se presentan las películas que el usuario ha valorado positiva y negativamente. También permite valorar nuevas películas o modificar valoraciones realizadas previamente. De esta forma, se ofrece al usuario una visión clara de sus gustos y preferencias, brindándole la oportunidad de ajustar su perfil y recibir recomendaciones aún más precisas y adaptadas a sus intereses cambiantes. Cabe destacar que, con el objetivo de que la aplicación sea sencilla e intuitiva, todas las películas se muestran a través de la carátula asociada, tal y como se muestra en la Figura 7.1.

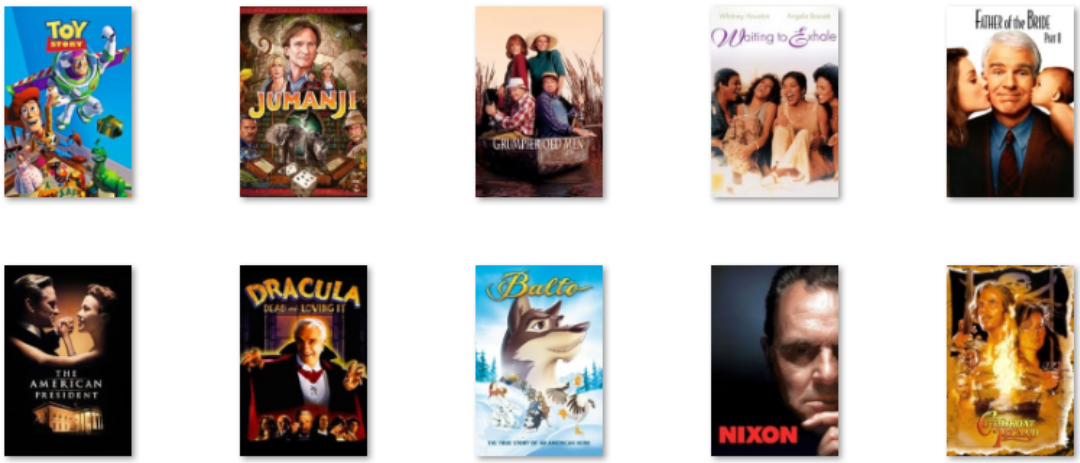


Figura 7.1: Ejemplo de presentación de las recomendaciones

En resumen, esta aplicación de recomendación de películas ofrece una experiencia altamente personalizada a los usuarios registrados. A través de un proceso de registro y valoración, se capturan los gustos cinematográficos de cada usuario y se generan recomendaciones personalizadas utilizando el algoritmo SVD++ desarrollado previamente. Con la posibilidad de explorar y modificar las preferencias en tiempo real, los usuarios pueden disfrutar de una experiencia cinematográfica a medida y descubrir nuevas películas que se ajusten a sus gustos individuales.

El entorno de producción simulado desarrollado está disponible en el siguiente repositorio: https://github.com/inesgfortis/RecommenderSys/dashboard.

Capítulo 8

Conclusiones y trabajo futuro

En conclusión, este trabajo ha abordado la **exploración, desarrollo, evaluación y comparación de diversos algoritmos de recomendación, con el objetivo de comprender su rendimiento y aplicabilidad en diversos contextos**. Se ha constatado la existencia de múltiples enfoques para la construcción de sistemas de recomendación efectivos, cada uno con sus ventajas e inconvenientes inherentes.

A lo largo del estudio, se han analizado algoritmos como los sistemas basados en usuarios, los sistemas basados en elementos, la descomposición de valores singulares (SVD y SVD++), así como técnicas de Deep Learning como RBM y AutoRec. Cada uno de estos enfoques ha sido sometido a una evaluación utilizando métricas de rendimiento offline, tales como RMSE, MAE, MAP y otras. Estas métricas han proporcionado una visión cuantitativa del desempeño de los algoritmos en términos de precisión y calidad de las recomendaciones. No obstante, se ha resaltado que, si bien **las métricas offline son útiles para la comparación en un entorno controlado, no logran capturar plenamente la experiencia del usuario final**. Por consiguiente, es esencial considerar la retroalimentación y la satisfacción del usuario como la medida definitiva del éxito de un sistema de recomendación. Además, se ha enfatizado la **importancia de evaluar los algoritmos en entornos en tiempo real**, mediante la implementación de un dashboard que simule un entorno de producción. Esta evaluación en tiempo real permite recopilar comentarios y opiniones directas de los usuarios, lo que contribuye a la adaptación y mejora continua de los algoritmos, con el fin de ofrecer recomendaciones más relevantes y personalizadas.

En resumen, los sistemas de recomendación se configuran como herramientas poderosas en la era digital, al permitir a las empresas mejorar la experiencia del usuario, aumentar la satisfacción del cliente y promover el crecimiento empresarial. **La elección del algoritmo de recomendación más adecuado depende del contexto específico, el problema que se pretenda abordar, así como los datos y tecnologías disponibles**. La evaluación precisa y la retroalimentación de los usuarios finales resultan fundamentales para garantizar el éxito y la efectividad de un sistema de recomendación.

En última instancia, el objetivo principal de los sistemas de recomendación radica en ofrecer una experiencia de usuario excepcional, en la cual los usuarios se sientan satisfechos y encuentren valor en las recomendaciones proporcionadas. Mediante la continuación de la investigación y la mejora constante de los algoritmos de recomendación, resulta posible avanzar en la personalización de la experiencia del usuario y maximizar los beneficios tanto para las empresas como para los usuarios.

Apéndice A

Alineación del proyecto con los ODS

Los Objetivos de Desarrollo Sostenible (ODS) son una serie de metas y objetivos establecidos por las Naciones Unidas (ONU) en 2015 para abordar los desafíos globales y promover el desarrollo sostenible a nivel mundial. Consisten en 17 objetivos interrelacionados que abarcan áreas sociales, económicas y ambientales, y tienen como finalidad principal erradicar la pobreza, promover la igualdad de género, garantizar la educación de calidad, fomentar la acción climática y fortalecer las alianzas. En otras palabras, los ODS representan un llamado a la acción para alcanzar las metas expuestas para el año 2030.

En relación a este trabajo, se destaca su contribución a algunos de los Objetivos de Desarrollo Sostenible, especialmente en las dimensiones social y económica. Los objetivos a los que contribuye de manera significativa son los siguientes:

- **ODS 5 Igualdad de género.** Los algoritmos utilizados en los sistemas de recomendación se basan en datos que permiten clasificar a los usuarios en diferentes grupos sin discriminación de género. Estos algoritmos no tienen prejuicios de género y promueven la igualdad de oportunidades para todos los usuarios.

- **ODS 9 Industria, Innovación e infraestructura.** La introducción de los sistemas de recomendación ha generado una auténtica revolución en la economía del comercio electrónico. Estas técnicas fomentan la creatividad y la innovación mediante el uso de la tecnología, impulsando así el progreso en diferentes sectores.

- **ODS 10 Reducción de las desigualdades.** Como se ha expuesto anteriormente, los modelos de recomendación pueden aplicarse en una amplia gama de contextos e industrias. Esta tecnología es accesible para personas de diferentes orígenes y contribuye a reducir las brechas existentes, promoviendo un acceso equitativo a las recomendaciones personalizadas.

- **ODS 11 Ciudades y comunidades sostenibles.** El crecimiento del comercio electrónico, en el cual se basan los sistemas de recomendación, impulsa un desarrollo sostenible a través de Internet. Estos sistemas permiten optimizar la oferta y la demanda, reduciendo la necesidad de recursos innecesarios y promoviendo la eficiencia en el consumo.

A continuación se presentan los objetivos previamente mencionados (Figura A.1)

Figura A.1: Contribución de los sistemas de recomendación a los ODS

En conclusión, los sistemas de recomendación analizados en este estudio se presentan como una valiosa herramienta tecnológica con un impacto significativo en diversas industrias y contextos, brindando una oportunidad para abordar los desafíos globales desde una perspectiva tecnológica.

Apéndice B

Métricas de Evaluación

A continuación se presenta el código correspondiente a las métricas de evaluación de los algortimos descritas en el Capítulo 4

Listing B.1: metrics.py

```
from collections import defaultdict
import pandas as pd
import numpy as np

class evaluationMetrics:

    def __init__(self):
        pass

    def getTopN(self, predictions, n=10, minimumRating=4.0):
        """Returns the top N recommended movies for each user based on
    estimated ratings.

        Args:
            predictions (list of tuples): A list of predictions where each
    tuple contains a user ID, a movie ID, an actual rating, an estimated
    rating, and some additional information.
            n (int, optional): The number of top recommended movies to
    return for each user. Defaults to 10.
            minimumRating (float, optional): The minimum estimated rating
    for a movie to be considered a top recommendation. Defaults to 4.0.

        Returns:
            defaultdict: A defaultdict where the keys are user IDs and the
    values are lists of tuples,
                where each tuple contains a movie ID and its estimated
    rating. The length of each list
                is equal to the specified value of `n` or less if there are
     not enough movies that meet
                the minimum rating threshold.
        """
        topN = defaultdict(list)

        for userID, movieID, actualRating, estimatedRating, _ in
    predictions:
            if (estimatedRating >= minimumRating):
                topN[int(userID)].append((int(movieID), estimatedRating))

        for userID, ratings in topN.items():
```

```
            ratings.sort(key=lambda x: x[1], reverse=True)
            topN[int(userID)] = ratings[:n]

        return topN


    def getPrecision(self,predictions, k=10, threshold=3.5):
        """Returns precision at k metric for each user

        Args:
        predictions: list of tuples (uid, iid, true_r, est, _)
        k: int, the maximum number of recommended items to consider for
    each user
        threshold: float, the minimum estimated rating for an item to be
    considered relevant

        Returns:
        precisions: dict, a dictionary where each key is a user ID and the
    value is the precision@k score for that user
        """
        # First map the predictions to each user.
        user_est_true = defaultdict(list)
        for uid, _, true_r, est, _ in predictions:
            user_est_true[uid].append((est, true_r))

        precisions = dict()
        for uid, user_ratings in user_est_true.items():

            # Sort user ratings by estimated value
            user_ratings.sort(key=lambda x: x[0], reverse=True)

            # Number of recommended items in top k
            n_rec_k = sum((est >= threshold) for (est, _) in user_ratings[:
    k])

            # Number of relevant and recommended items in top k
            n_rel_and_rec_k = sum(((true_r >= threshold) and (est >=
    threshold))
                                  for (est, true_r) in user_ratings[:k])

            # Precision@K: Proportion of recommended items that are
    relevant
            # When n_rec_k is 0, Precision is undefined. We here set it to
    0.

            precisions[uid] = n_rel_and_rec_k / n_rec_k if n_rec_k != 0
    else 0

        return precisions


    def getRecall(self,predictions, k=10, threshold=3.5):
        """Return recall at k metric for each user

        Args:
        predictions: list of tuples (uid, iid, true_r, est, _)
        k: int, the maximum number of recommended items to consider for
    each user
        threshold: float, the minimum estimated rating for an item to be
    considered relevant
```

```
        Returns:
        recalls: dict, a dictionary where each key is a user ID and the
    value is the recall@k score for that user
        """

        # First map the predictions to each user.
        user_est_true = defaultdict(list)
        for uid, _, true_r, est, _ in predictions:
            user_est_true[uid].append((est, true_r))

        recalls = dict()
        for uid, user_ratings in user_est_true.items():

            # Number of relevant items
            n_rel = sum((true_r >= threshold) for (_, true_r) in
    user_ratings)

            # Number of recommended items in top k
            n_rec_k = sum((est >= threshold) for (est, _) in user_ratings[:
    k])

            # Number of relevant and recommended items in top k
            n_rel_and_rec_k = sum(((true_r >= threshold) and (est >=
    threshold))
                                  for (est, true_r) in user_ratings[:k])

            # Recall@K: Proportion of relevant items that are recommended
            # When n_rel is 0, Recall is undefined. We here set it to 0.

            recalls[uid] = n_rel_and_rec_k / n_rel if n_rel != 0 else 0

        return recalls


    def predictionsToDataframe(self,predictions):
        """
        Convert a list of predictions into a pandas dataframe.

        Args:
        - predictions: a list of tuples (user_id, item_id, actual_rating,
    predicted_rating, _)

        Returns:
        - a pandas dataframe with columns "user_id", "item_id", "rating", "
    prediction", sorted by user_id and prediction
        """

        data = []
        for uid, iid, true_r, est_r, _ in predictions:
            data.append([uid,iid,true_r,est_r])

        df = pd.DataFrame(data, columns = ["user_id", "item_id","rating","
    prediction"])
        df.sort_values(["user_id","prediction"],inplace = True, ascending =
     [True,False])

        return df


    def getNDCG(self,predictions, k):
        """
```

```
    Compute the Normalized Discounted Cumulative Gain (NDCG) at k for
each user in a pandas dataframe.

    Args:
    - df: a pandas dataframe with columns "user_id", "item_id", "rating
", "prediction", sorted by user_id and prediction
    - k: the number of items to consider for computing the NDCG

    Returns:
    - a tuple (ndcgs, mean_ndcg), where:
      - ndcgs is a list of NDCGs (one for each user)
      - mean_ndcg is the average NDCG over all users
    """

    # Adapt the format of the model prediction output to df
    df_test = self.predictionsToDataframe(predictions)

    grouped = df_test.groupby('user_id')
    ndcgs = []
    for _, group in grouped:
        group = group.head(k)
        group.reset_index(inplace = True, drop = True)
        dcg = (group['rating'] / np.log2(group.index + 2)).sum()
        ideal = group.sort_values(by='rating', ascending=False)
        ideal.reset_index(inplace = True, drop = True)
        idcg = (ideal['rating'] / np.log2(ideal.index + 2)).sum()
        ndcgs.append(dcg / idcg)

    return ndcgs, np.mean(ndcgs)


def getCoverage(self,topNrec, total_items, users):
    """
    Compute the item coverage of the recommendations.

    Args:
        topNrec (dict): A dictionary containing top-N recommendations
for each user.
        total_items (int): Total number of unique items in the dataset.
        users (list): List of user IDs for whom the recommendations are
 generated.

    Returns:
        The item coverage of the recommendations as a ratio of unique
recommended items to total items.
    """
    rec_items = []

    for user in users:
        rec_items += [pred[0] for pred in topNrec[user]]

    num_rec_items = len(set(rec_items))
    coverage = num_rec_items/total_items

    return coverage



def getUserCoverage(self,topNPredicted, numUsers, ratingThreshold=0):
    """
    Calculate user coverage of recommended items.
```

```
    Args:
        topNPredicted: A dictionary where keys are user IDs and values
are lists of tuples (item ID, predicted rating).
        numUsers: An integer indicating the total number of users in
the dataset.
        ratingThreshold: A float indicating the minimum predicted
rating required for an item to be considered as relevant.

    Returns:
        float indicating the proportion of users for whom at least one
recommended item is relevant according to the rating threshold.
    """
    hits = 0
    for userID in topNPredicted.keys():
        hit = False
        for movieID, predictedRating in topNPredicted[userID]:
            if (predictedRating >= ratingThreshold):
                hit = True
                break
        if (hit):
            hits += 1

    user_coverage = hits / numUsers

    return user_coverage


def getPopularity(self,trainset):
    """
    Calculate the popularity rank of movies in the training set based
on the number of ratings received.

    Args:
    trainset: The training set in Surprise format.

    Returns:
    dict_popularity: A dictionary containing the popularity rank of
each movie in the training set,
                     where the key is the movie id and the value is its
 popularity rank.
    """
    # Convert the training set to a pandas DataFrame
    train_df = pd.DataFrame(trainset.all_ratings(), columns=['userid',
'movieid', 'rating'])
    train_df['userid'] = train_df['userid'].astype(int)
    train_df['movieid'] = train_df['movieid'].astype(int)

    # Count the number of ratings received by each movie
    interacciones = train_df['movieid'].value_counts()

    # Assign a popularity rank to each movie based on the number of
ratings received
    dict_popularity = dict(zip(interacciones.index, range(1, len(
interacciones) + 1)))

    return dict_popularity


def getNovelty(self,topNPredicted, trainset):
    """
```

```
        Computes the novelty of the recommendations based on the popularity
    of the recommended items.

        Args:
        - topNPredicted: dictionary containing the top N predicted items
    for each user
        - trainset: the training set of the dataset

        Returns:
        - novelty: a float value representing the average popularity rank
    of the recommended items
        """
        # Define the popularity of films on the basis of the number of
    ratings received
        rankings = self.getPopularity(trainset)

        # Compute novelty
        n = 0
        total = 0
        for userID in topNPredicted.keys():
            for rating in topNPredicted[userID]:
                movieID = rating[0]
                rank = rankings[trainset.to_inner_iid(movieID)]
                total += rank
                n += 1

        novelty = total/n
        return novelty



    def addToMetricsDataframe(self,dataframe):
        """
        Appends rows from the input DataFrame to an existing metrics Excel
    file.
        If the file doesn't exist, creates a new metrics DataFrame with the
     required columns.

        Args:
            dataframe (pd.DataFrame): DataFrame containing the new metrics
    to be appended.

        Returns:
            None
        """
        # File path
        FILE_PATH = "metrics.xlsx"

        # Metrics file
        try:
            existing_metrics = pd.read_excel(FILE_PATH)
        except FileNotFoundError:
            existing_metrics = pd.DataFrame(columns=["Model", "RMSE", "MAE"
    , "MAP", "MAR", "Mean_NDCG", "Coverage", "User_Coverage", "Novelty"])

        # Append new rows to the existing metrics DataFrame
        updated_metrics = pd.concat([existing_metrics, dataframe],
    ignore_index=True)

        # Save the updated metrics DataFrame to the Excel file
        updated_metrics.to_excel(FILE_PATH, index=False)
```

Bibliografía

[1] Aggarwal CC, et al. Recommender systems. vol. 1. Springer; 2016.

[2] Lü L, Medo M, Yeung CH, Zhang YC, Zhang ZK, Zhou T. Recommender systems. Physics reports. 2012;519(1):1-49.

[3] Chen PY, Wu SY, Yoon J. The impact of online recommendations and consumer feedback on sales. In: Proceedings of the 25th International Conference on Information Systems; 2004. p. 711-24.

[4] Schafer JB, Konstan JA, Riedl J. E-commerce recommendation applications. Data mining and knowledge discovery. 2001;5:115-53.

[5] Newman ME. Power laws, Pareto distributions and Zipf's law. Contemporary physics. 2005;46(5):323-51.

[6] Weibull W. A statistical distribution function of wide applicability. Journal of Applied Mechanics-Transactions of the ASME. 1951;18(3):293-7.

[7] Huang Z, Chen H, Zeng D. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems (TOIS). 2004;22(1):116-42.

[8] Sarwar B, Karypis G, Konstan J, Riedl J. Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Fifth international conference on computer and information science. Citeseer; 2002. p. 27-8.

[9] Zhang ZK, Liu C, Zhang YC, Zhou T. Solving the cold-start problem in recommender systems with social tags. Europhysics Letters. 2010;92(2):28002.

[10] McNee SM, Riedl J, Konstan JA. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI'06 extended abstracts on Human factors in computing systems; 2006. p. 1097-101.

[11] Zhou T, Kuscsik Z, Liu JG, Medo M, Wakeling JR, Zhang YC. Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the National Academy of Sciences. 2010;107(10):4511-5.

[12] Mobasher B, Burke R, Bhaumik R, Williams C. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. ACM Transactions on Internet Technology (TOIT). 2007;7(4):23-es.

[13] Lam SK, Frankowski D, Riedl J. Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems. In: Lecture Notes in Computer Science. vol. 3995. Heidelberg, Germany: Springer; 2006. p. 14-29.

[14] Xiang L, Yuan Q, Zhao S, Chen L, Zhang X, Yang Q, et al. Temporal recommendation on graphs via long-and short-term preference fusion. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining; 2010. p. 723-32.

[15] Sinha R, Swearingen K. The role of transparency in recommender systems. In: CHI'02 extended abstracts on Human factors in computing systems; 2002. p. 830-1.

[16] Sundog Education by Frank Kane. Building Recommender Systems with Machine Learning and AI; 2022. Online. Available from: `https://www.udemy.com/course/building-recommender-systems-with-machine-learning-and-ai/`.

[17] Banik R. Hands-on recommendation systems with Python: start building powerful and personalized, recommendation engines with Python. Packt Publishing Ltd; 2018.

[18] Burke R. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction. 2002;12:331-70.

[19] Burke R. Hybrid web recommender systems. The adaptive web: methods and strategies of web personalization. 2007:377-408.

[20] Schein AI, Popescul A, Ungar LH, Pennock DM. Methods and metrics for cold-start recommendations. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; 2002. p. 253-60.

[21] Gomez-Uribe CA, Hunt N. The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS). 2015;6(4):1-19.

[22] Hug N. Surprise: A Python library for recommender systems; 2021. Retrieved from `http://surpriselib.com/`.

[23] Järvelin K, Kekäläinen J. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems (TOIS). 2002;20(4):422-46.

[24] Castells P, Vargas S, Wang J. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. In: Proceedings of International Workshop on Diversity in Document Retrieval, DDR. New York: ACM Press; 2011. p. 29-37.

[25] Lü L, Liu W. Information filtering via preferential diffusion. Physical Review E. 2011;83(6):066119.

[26] Adomavicius G, Tuzhilin A. 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. 2005;17(6):734-49.

[27] Salton G. Introduction to modern information retrieval. McGraw-Hill. 1983.

[28] Schafer JB, Frankowski D, Herlocker J, Sen S. Collaborative filtering recommender systems. The adaptive web: methods and strategies of web personalization. 2007:291-324.

[29] Herlocker JL, Konstan JA, Borchers A, Riedl J. An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval; 1999. p. 230-7.

[30] Sarwar B, Karypis G, Konstan J, Riedl J. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web; 2001. p. 285-95.

[31] Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009;42(8):30-7.

[32] Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining; 2008. p. 426-34.

[33] Custódio AL, Rocha H, Vicente LN. Incorporating minimum Frobenius norm models in direct search. Computational Optimization and Applications. 2010;46(2):265-78.

[34] Myung IJ. Tutorial on maximum likelihood estimation. Journal of mathematical Psychology. 2003;47(1):90-100.

[35] Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:12052618. 2012.

[36] Hinton GE. A practical guide to training restricted Boltzmann machines. Neural Networks: Tricks of the Trade: Second Edition. 2012:599-619.

[37] Salakhutdinov R, Mnih A, Hinton G. Restricted Boltzmann machines for collaborative filtering. In: Proceedings of the 24th international conference on Machine learning; 2007. p. 791-8.

[38] Salakhutdinov R, Mnih A. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In: Proceedings of the 25th international conference on Machine learning; 2008. p. 880-7.

[39] Sedhain S, Menon AK, Sanner S, Xie L. Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th international conference on World Wide Web; 2015. p. 111-2.

[40] Vincent P, Larochelle H, Bengio Y, Manzagol PA. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning; 2008. p. 1096-103.

[41] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. science. 2006;313(5786):504-7.

[42] Zhou T, Ren J, Medo M, Zhang YC. Bipartite network projection and personal recommendation. Physical review E. 2007;76(4):046115.

[43] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: A review of methods and applications. AI open. 2020;1:57-81.

[44] Gori M, Pucci A. Research paper recommender systems: A random-walk based approach. In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06). IEEE; 2006. p. 778-81.

[45] He X, Liao L, Zhang H, Nie L, Hu X, Chua TS. Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web; 2017. p. 173-82.

[46] Covington P, Adams J, Sargin E. Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM conference on recommender systems; 2016. p. 191-8.

[47] Rendle S. Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST). 2012;3(3):1-22.

[48] Hidasi B, Karatzoglou A, Baltrunas L, Tikk D. Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:151106939. 2015.

[49] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems. 2017;30.

[50] Zhang L, Wang S, Liu B. Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2018;8(4):e1253.

[51] Kapoor N, Vishal S, Krishnaveni K. Movie recommendation system using nlp tools. In: 2020 5th International Conference on Communication and Electronics Systems (ICCES). IEEE; 2020. p. 883-8.

[52] Adomavicius G, Tuzhilin A. Context-aware recommender systems. In: Recommender systems handbook. Springer; 2010. p. 217-53.

[53] Krauth K, Dean S, Zhao A, Guo W, Curmei M, Recht B, et al. Do offline metrics predict online performance in recommender systems? arXiv preprint arXiv:201107931. 2020.