

Desarrollo de un Sistema de recomendación de canciones usando Spotify

Víctor Arreaga, Oscar Bastidas, Solange Briones

Facultad de Ciencias de la Ingeniería, Universidad Estatal de Milagro, Milagro, Ecuador

Abstract—Esta investigación brindará una visión general de los sistemas de recomendación (SR), explicando qué son y su importancia actual. También se analizan los tres tipos de SR: filtrado basado en contenido, filtrado demográfico y filtrado colaborativo de los cuáles se hará una comparación y como se califican los SR.

Crearemos un SR del tipo basado en el contenido para el segmento práctico, que creará una lista de reproducción a partir de las canciones recién escuchadas de un usuario de Spotify. Utilizaremos el lenguaje de programación Python junto con el paquete Scikit Learn y sus algoritmos de clasificación, así como Numpy, Pandas y Spotipy. El conjunto de datos serán las canciones extraídas de Spotify usando su API.

Finalmente, creamos el SR y, como resultado, todas las canciones propuestas estuvieron disponibles en Spotify. Pudimos encontrar canciones que estaban conectadas a las canciones recientemente escuchadas del usuario usando la similitud del coseno.

Index Terms—SR, API, Spotify

I. INTRODUCCIÓN

En el presente documento constituye el estudio, análisis e indagación de los diferentes tipos de sistemas de recomendación, así como sus diferencias y diversos tipos de aplicaciones. Adicionalmente, en la sección práctica de este proyecto, demostraremos la implementación de un sistema de recomendación para la creación automática de una lista de reproducción es la plataforma musical de Spotify con un filtrado basado en contenido.

En la actualidad, los sistemas de recomendación son cada vez más importantes para la economía, una experiencia de usuario agradable, el marketing, etc. Cada vez se afinan más los algoritmos de recomendación para que acierten de forma más exacta las predicciones para los usuarios o clientes de acuerdo a sus gustos.

El uso más conocido de los algoritmos de recomendación es en el comercio electrónico, donde las preferencias de un cliente se utilizan para crear una lista de productos sugeridos. Numerosos sitios web y aplicaciones analizan el comportamiento de los usuarios en función de calificaciones, compras y vistas anteriores de productos antes de hacer recomendaciones a los usuarios en función de sus intereses. Para hacer que la tienda en línea sea única para cada consumidor y atraerlos mostrando a menudo cosas que se ajustan a sus preferencias, las empresas de comercio electrónico utilizan algoritmos de recomendación. SR se ha convertido en una herramienta útil que ofrece a los clientes sugerencias individ-

ualizadas para productos como ropa, libros, películas, música y zapatos.

II. METODOLOGÍA

En esta sección se dará una introducción, conceptualización y comparación de los diferentes tipos de SR que existen y como ejemplo se pondrá a prueba los SR.

A. Introducción a los sistemas de recomendación

Los sistemas de recomendación (SR) son herramientas que interactúan con grandes y complejos volúmenes de información que ofrecen una visión personalizada de los mismos priorizando los elementos que predice que pueden interesarle al usuario sobre el que se centra el enfoque. Se trata de un campo que ha gozado de un gran crecimiento por lo que se puede encontrar una gran variedad de técnicas basadas en recursos del área de la Inteligencia artificial.

Los SR están omnipresentes en Internet y son uno de los principales vehículos de satisfacción y retención de clientes para muchas empresas que ofrecen productos de comercio electrónico y/o entretenimiento como Amazon, YouTube, TikTok, Instagram o Netflix [14].

La base de todos los SR son los siguientes 3 pilares principales:

- **Usuarios:** el objetivo de las recomendaciones.
- **Ítems:** Los productos o servicios a recomendar. El sistema extraerá las características fundamentales de cada artículo para recomendarlo a los usuarios interesados.
- **Calificaciones:** la cuantificación del interés de un usuario en particular por un elemento específico. Esta votación puede ser implícita (el usuario expresa su interés en una escala preestablecida) o explícita (el sistema infiere la preferencia en función de la interacción usuario-elemento). Estos 3 pilares vienen a formar la matriz Usuario-Elemento, como se ve en la Figura 1.

Saber lo que le gustaría ver a un usuario es crucial para mantenerlo en un sitio, y también lo es predecir su próxima necesidad o deseo para que compre un producto o servicio específico, atendiendo a los intereses específicos de cada individuo. Para que esto funcione, se requiere una gran cantidad de datos para establecer ciertos perfiles y poder vincular a las personas con los elementos que podrían interesarles.

En el caso de Netflix, como en muchos otros, “los clientes han demostrado estar dispuestos a para indicar su nivel de



Fig. 1. Ilustración de la matriz de interacciones Usuario-Elemento. [14].

satisfacción [...], por lo que se dispone de un gran volumen de datos sobre qué películas atraen a qué clientes” [3].

B. La información en un sistema de recomendación

Hay 4 niveles principales de información sobre los que puede trabajar un SR, de menor a mayor cantidad y complejidad de información utilizada:

- **Nivel basado en la memoria:** Abarca las calificaciones (ya sean implícitas o explícitas).
- **Nivel basado en el contenido:** Agrega información adicional para mejorar la calidad de las recomendaciones, como la información demográfica de los usuarios (edad, ocupación, etnia...), la actividad relacionada con la aplicación o la información implícita y explícita de los elementos (por ejemplo, cuántos veces es un elemento mirado - implícito, qué actores protagonizan una película - explícito).
- **Nivel basado en información social:** Emplea datos pertenecientes a la web social de la que forma parte el usuario (qué elementos se recomiendan a los amigos, quiénes son los seguidores del usuario y a quién sigue el usuario...). Los elementos son etiquetados por los usuarios y se clasifican según las interacciones del usuario con ellos, como me gusta, no me gusta, compartir, etc.
- **Nivel basado en el contexto:** Hace uso del Internet de las Cosas (IoT) para incorporar información personal implícita del usuario a través de otros dispositivos sobre hábitos personales, geolocalización, biometría... Items-wise, integración ambiental, geolocalización y muchos otros son ejemplos de datos utilizados para la recomendación. Los resultados son recomendaciones similares a “Notamos que estás sudando y dando un paseo, John. Estas son las mejores heladerías a tu alrededor para disfrutar de un descanso fresco en este caluroso día de verano”.

C. Tipos de sistemas de recomendación

Hay varios tipos de SR, y la principal diferencia radica en la forma en que filtran la información, lo que genera tres paradigmas principales: filtrado demográfico, filtrado basado

en contenido y filtrado colaborativo. La Figura 2 muestra un desglose en sus ramas individuales:

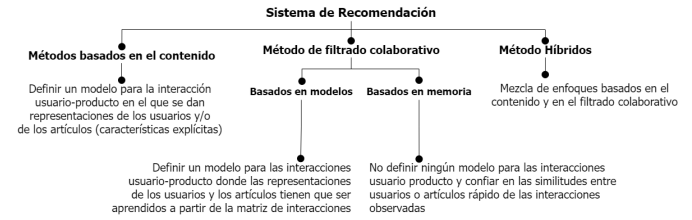


Fig. 2. Resumen de los diferentes tipos de algoritmos de los sistemas de recomendación. [14].

FILTRADO DEMOGRÁFICO

Las recomendaciones se dan en función de las características demográficas de los usuarios, basadas en el supuesto de que los usuarios con características similares tendrán intereses similares. Este tipo de recomendaciones suelen ser aburridas, imprecisas y rara vez innovadoras”.

Este tipo de filtrado da como resultado un SR extremadamente sesgados en la gran mayoría de los casos, una tendencia de la que el campo ML está tratando de alejarse, lo que lo convierte en un método de filtrado que ha caído en desgracia y rara vez se usa.

FILTRADO BASADO EN EL CONTENIDO

El filtrado basado en contenido se basa en la creación de un perfil en torno a las características/información/interés de un usuario o de un elemento que, en el caso de los usuarios, suele ser de carácter personal o sensible, como la edad, el sexo, el salario, la religión... [3].

La idea de los métodos basados en contenido es intentar construir un modelo, basado en las “características” disponibles, que explique las interacciones usuario-elemento observadas como se ve en la Figura 3. Esto convierte el problema en una dicotomía de clasificación o regresión (el usuario le gusta un artículo o no versus predecir la calificación aproximada que un usuario le dará a un artículo) [14].

Un ejemplo de filtrado basado en contenido sería pedirle al usuario que complete una encuesta sobre sus intereses antes de sugerir algo, como lo hacen Apple News o Medium para recomendar artículos y publicaciones para leer; o “el perfil de una película [...] (que incluye) atributos relacionados con su género, los actores participantes, su popularidad en la taquilla, etc.” [3].

Para este tipo de sistema de recomendación se usa la similitud del coseno (Ecuación 1), que es una función de dos vectores que indica la similitud entre ellos.

$$\text{Cos}(x, y) = \frac{x * y}{\|x\| * \|y\|} \quad (1)$$



Fig. 3. Descripción general del paradigma de métodos basados en contenido. [14].

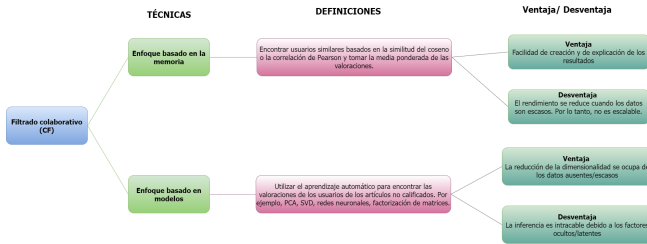


Fig. 4. Desglose de los dos enfoques del filtrado colaborativo. [15].

FILTRADO COLABORATIVO

A diferencia del Filtrado basado en contenido mencionado anteriormente, el Filtrado colaborativo emplea solo la actividad pasada de un usuario, por ejemplo, calificaciones pasadas, consultas de búsqueda, historial de navegación o compras.

“El Filtrado Colaborativo analiza las relaciones entre los usuarios y las interdependencias entre los productos para identificar nuevas asociaciones usuario-elemento” [3]. Este formato surge de la costumbre humana de pedir recomendaciones a amigos y familiares con gustos similares. Cuantos más usuarios interactúen con los elementos, más precisas serán las recomendaciones.

La información se representa comúnmente en matrices de calificación de elemento de usuario o de interacción de elemento de usuario, que casi siempre son escasas, ya que es muy común que haya una gran cantidad de elementos para que un usuario los evalúe.

Se pueden distinguir dos métodos principales: basado en memoria y basado en modelo como se ve en la Figura 4 y la Figura 5 a continuación.

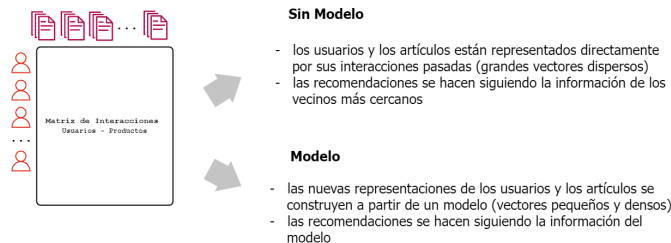


Fig. 5. Descripción general del paradigma de métodos de filtrado colaborativo [14].

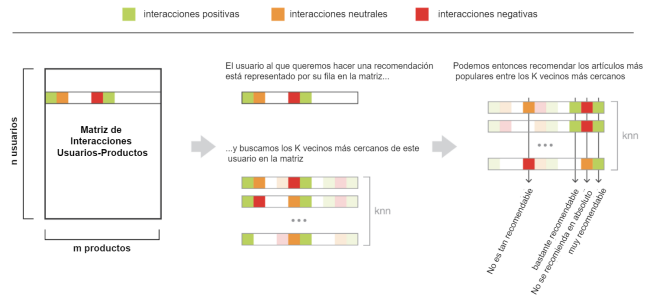


Fig. 6. Ejemplo del método KNN usuario-usuario. [14].

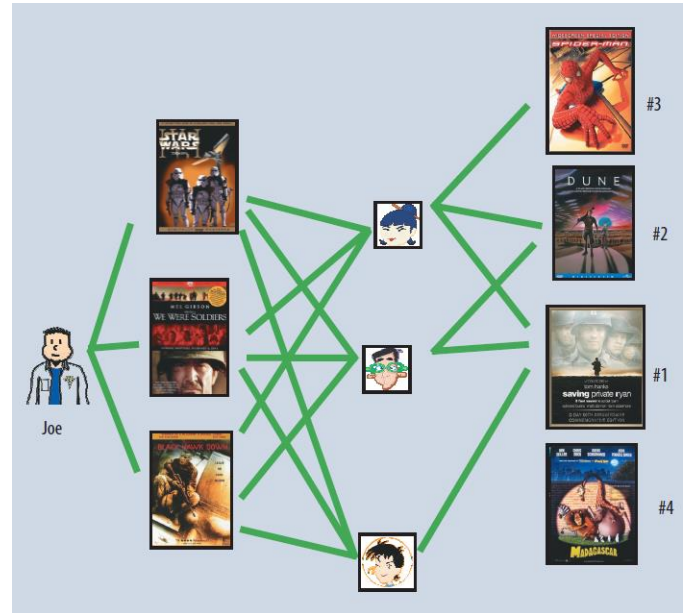


Fig. 7. Método de vecindad orientado a usuario. Se obtiene una predicción para Joe de los otros 3 usuarios considerados similares a él, a quienes les gustaron las películas. [3].

Métodos basados en memoria

También conocido como método de vecindad. Las recomendaciones se basan directamente en la matriz de calificaciones. Consisten en métodos basados en el usuario y en el artículo.

- Basado en el usuario:** También conocido como usuario-usuario. Las recomendaciones de un usuario X dependerán de las calificaciones de usuarios similares en elementos que X aún no ha calificado, como se ve en la Figura 6 y la Figura 7. La similitud entre los usuarios se puede calcular encontrando usuarios con calificaciones similares para una amplia selección de elementos.
- Basado en elementos:** También conocido como elemento-elemento. Las recomendaciones de un usuario X dependerán de sus propias calificaciones, y el sistema recomendará elementos similares a los ya calificados, como se ve en la Figura 8. La similitud entre los elementos se puede obtener delineando las características clave de los elementos o agrupándolos según cómo lo

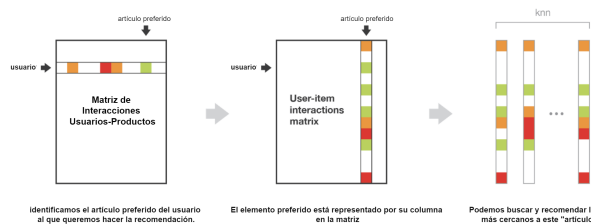


Fig. 8. Ejemplo del método KNN usuario-usuario. [14].

mayoría de los usuarios interactuaron con cada uno de ellos. “[...] los métodos basados en ítems tienen mejores resultados en general ya que la similitud entre ítems (es) más confiable que la similitud entre usuarios. Imagine que, después de un tiempo, Alice puede cambiar sus preferencias y calificaciones y esto provoca un rendimiento deficiente en los métodos basados en el usuario.

”[...] los métodos basados en artículos tienen mejores resultados en general, ya que la similitud entre artículos (es) más fiable que la similitud entre usuarios. Imaginemos que, al cabo de un tiempo, Alice puede cambiar sus preferencias y valoraciones, lo que provoca el bajo rendimiento de los métodos basados en el usuario.” [13] En la mayoría de los casos con los métodos usuario-usuario, cada usuario habrá interactuado con un número limitado de elementos, haciéndolos muy susceptibles a interacciones individuales registradas (varianza alta) pero más personalizadas (sesgo bajo). Lo contrario es cierto con los métodos de elemento a elemento (baja varianza, alto sesgo).

En general, KNN (K-Nearest Neighbours) es la implementación más popular, donde se buscan los k usuarios o elementos más similares a uno en particular para derivar recomendaciones basadas en ellos. También se emplea Aproximación de vecinos más cercanos (ANN), especialmente con conjuntos de datos más grandes. El principal problema de estos sistemas de recomendación es la complejidad y la falta de escalabilidad. Como parte de un sistema mayor con millones de usuarios y/o elementos, calcular constantemente los vecinos más cercanos puede ser una tarea hercúlea. Otro problema proviene del problema de “los ricos se hacen más ricos”, donde se recomiendan los mismos artículos populares para todos, o donde los usuarios quedan atrapados en “áreas de confinamiento de la información” donde se recomiendan artículos muy similares a los que ya les gustan, y otros nuevos o disruptivos. no tienen ninguna posibilidad (no se consideran “vecinos lo suficientemente cercanos”). [14]

Se puede ver una descripción general de cada tipo en Figura 9.

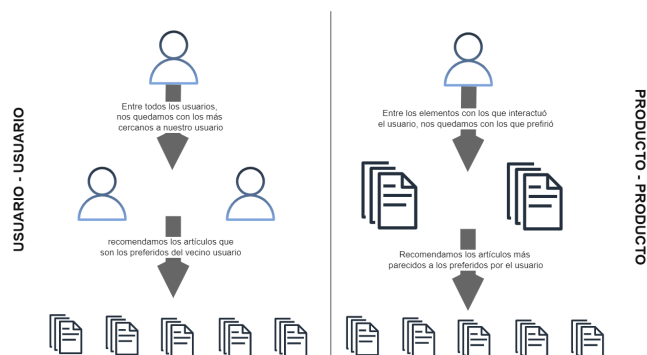


Fig. 9. Contraste entre los métodos de memoria basados en el usuario y los basados en elementos. [14].

Métodos basados en modelos

También conocidos como modelos de factores latentes. Las recomendaciones se obtienen de un modelo de la matriz de calificaciones. “[...] (estos enfoques) asumen un modelo “generativo” subyacente que explica las interacciones usuario-elemento y trata de descubrirlo para hacer nuevas predicciones” [3]. Un ejemplo muy simple y comprensible para los humanos de esto se puede ver en Figura 10.

Los modelos de mejor rendimiento generalmente se basan en la factorización matricial, que presenta la hipótesis de que las calificaciones de los usuarios están condicionadas a una serie de factores ocultos inherentes al tipo de elementos que se califican y, por lo tanto, la matriz de interacción usuario-elemento se divide en 2 densas matrices: la matriz usuario-factor y la matriz factor-elemento.

“Los métodos de factorización de matrices comprimen la matriz de elementos de usuario en una representación de baja dimensión en términos de factores latentes. Una ventaja de usar este enfoque es que en lugar de tener una matriz de alta dimensión que contenga una gran cantidad de valores faltantes, estaremos tratando con una matriz mucho más pequeña en un espacio de menor dimensión [...] Hay varias ventajas con este paradigma. Maneja la escasez de la matriz original mejor que las basadas en memoria. Además, comparar la similitud en la matriz resultante es mucho más escalable [...]”. [14]

A pesar de que, en general, brinda resultados superiores en comparación con el filtrado basado en contenido [3], el filtrado colaborativo tiene una desventaja importante: el problema del arranque en frío.

El problema de arranque en frío o simplemente, arranque en frío, se refiere a la incapacidad de recomendar artículos a nuevos usuarios o recomendar algo nuevo a un usuario existente debido a la falta de información suficiente.

Hay que tener en cuenta que este problema puede ocurrir en el Filtrado basado en contenido cuando se presenta un usuario con funciones nuevas nunca antes vistas, pero, en términos generales, tal evento rara vez ocurre y es mucho más predominante en el Filtrado colaborativo.

Hay una serie de posibles soluciones [14]:

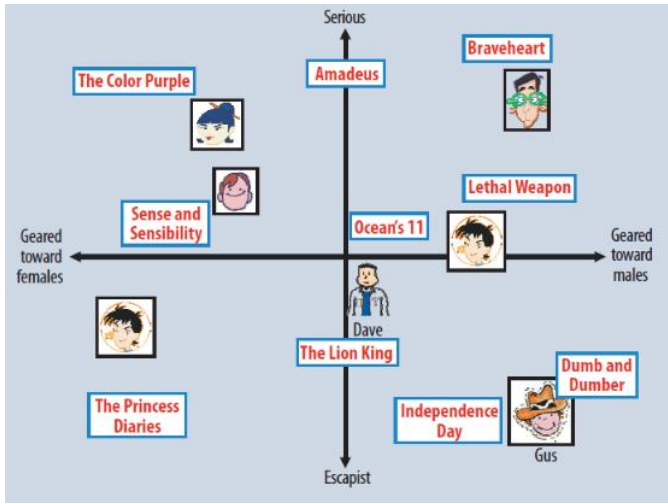


Fig. 10. Un ejemplo de un modelo de factor latente, que clasifica tanto a los usuarios como a los elementos en los mismos ejes. [3].

- **Estrategia aleatoria:** recomendar elementos aleatorios a usuarios nuevos o elementos nuevos a usuarios aleatorios.
- **Estrategia de máxima expectativa:** recomendar artículos populares a los nuevos usuarios o nuevos artículos a los usuarios más activos.
- **Estrategia exploratoria:** recomendar un conjunto de varios artículos a nuevos usuarios o un nuevo artículo a un conjunto de varios usuarios.
- Usar una encuesta u otra técnica no colaborativa para examinar los primeros años de un artículo o de un usuario.

EVALUACIÓN DE LOS SISTEMAS DE RECOMENDACIÓN

La evaluación se puede dividir en dos categorías: basada en métricas bien definidas y basada en la estimación de la satisfacción [14]. La evaluación de los sistemas de recomendación no es una tarea fácil a día de hoy y, en general, a menudo se recomienda utilizar una combinación de ambos tipos para garantizar la mejor SR posible [12].

Evaluación basada en métricas bien definidas

Al igual que otros modelos de Machine Learning, si el resultado es un valor numérico claro, se pueden usar las métricas de medición de errores existentes (como MAE, MSE o RMSE). Estos miden la calidad de las predicciones. Se tendría que realizar una división de prueba de tren, dividiendo el conjunto de datos en subconjuntos de entrenamiento y prueba, y realizar cálculos basados en esos resultados. Para ello, se podrían dividir tanto los ítems como los usuarios, por lo que habría que realizar dos splits tren-prueba diferentes, según el paradigma utilizado. En el caso del filtrado colaborativo basado en modelos, por ejemplo, dividir por usuarios no tendría sentido, ya que eso simplemente crearía problemas de arranque en frío para todos aquellos usuarios que el modelo no había visto previamente y, por lo tanto, solo sería sensato dividir los elementos.

El **error absoluto medio** es una medida calculada a partir de la diferencia absoluta de cada predicción y el valor real, promediado. Cuanto más bajo, mejor. La fórmula es la siguiente:

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (2)$$

El **error cuadrático medio** es una medida que se obtiene del promedio de las diferencias al cuadrado de cada predicción y el valor real. Root Mean Squared Error toma la raíz cuadrada del resultado. Debido al cuadrado, estas métricas son mucho más susceptibles a los extremos, mientras que MAE trata todos los errores como iguales. Cuanto más bajo, mejor. Las fórmulas son las siguientes:

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (3)$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad (4)$$

Otro conjunto de métricas que se pueden utilizar son AUC-ROC, F1 Score, Accuracy, Precision o Recall, según la naturaleza de cada problema. Estos miden la calidad del conjunto de recomendaciones. Todas son métricas derivadas de la Matriz de confusión (Figura 11)

AUC-ROC se basa en la curva de la tasa de verdaderos positivos, o sensibilidad, frente a la tasa de falsos positivos, o 1-sensibilidad.

$$TPR = Sensitivity = \frac{TP}{TP + FN} \quad (5)$$

$$FPR = 1 - Sensitivity = \frac{FP}{FP + TN} \quad (6)$$

La línea donde el TPR y el FPR son iguales significa que todos los casos que debían clasificarse como X lo eran, pero todos los que tenían que clasificarse como Y no lo eran, lo que significa que todo se clasificaba como X. Cuanto mayor sea la distancia anterior esa línea, mejor es el modelo. Cualquier lugar debajo significa un modelo horrible. La precisión es la relación entre las predicciones correctas y las predicciones totales, la mayor, siendo 1 la mejor y 0 la peor:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (7)$$

La precisión es la relación entre las predicciones positivas verdaderas y el total de predicciones positivas (tanto verdaderas como falsas); en otras palabras, de las predicciones positivas, cuántas fueron realmente positivas, siendo 1 la mejor y 0 la peor:

$$Accuracy = \frac{TP}{TP + FP} \quad (8)$$

La recuperación es la relación entre las predicciones positivas verdaderas y las que deberían haber sido positivas, es decir, de las que deberían haber sido clasificadas como positivas, cuántas fueron realmente clasificadas como tales, siendo 1 la mejor y 0 la peor:

		Condiciones Previstas	
Población Total = P + N		Condiciones Previstas Positivo (PP)	Condiciones Previstas Negativo (PN)
Condición Actual	Condición Actual Positiva (P)	Verdadero Positivo (TP) hit	Falso Negativo (FN) Tipo II error, fallo, sobreestimación
	Condición Actual Negativa (N)	Falso Positivo (FP) Tipo I error, falsa alarma, subestimación	Verdadero Negativo (TN), Rechazo Correcto

Fig. 11. Matriz de confusión [14].

$$Precision = \frac{TP}{TP + FN} \quad (9)$$

F1 Score es el equilibrio entre precisión y recuperación, siendo 1 el mejor y 0 el peor. En términos generales, indica una puntuación global para un modelo de clasificación:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (10)$$

III. COMPARACIÓN DE LOS TIPOS DE SR

Después de haber explorado cada tipo de SR (FD, FC, FBC) desde un punto de vista teórico, en esta parte compararemos sus ventajas y desventajas.

TABLE I
VENTAJAS DE LOS TIPOS DE SR

Filtrado demográfico	Es irrelevante que el usuario puntué los ítems o a su vez, solicitar información a otros usuarios
Filtrado colaborativo	Solo necesitan almacenar el vector de valores del usuario en lugar de una gran base de datos de información
Filtrado basado en contenido	Se aconseja utilizar productos con cualidades similares a las que el usuario activo respondió favorablemente

TABLE II
DESVENTAJAS DE LOS TIPOS DE SR

Filtrado demográfico	Es difícil obtener la cantidad de datos demográficos ideal para dar una recomendación apropiada.
Filtrado colaborativo	Se hace patente el problema del nuevo usuario, que carece de un perfil definido por no haber valorado ninguna de sus cuentas.
Filtrado basado en contenido	No se pueden utilizar con sistemas en desarrollo. Y además los algoritmos son de un alto costo computacional

Asimismo, partiendo de la conceptualización de los tipos de SR se han tomado características claves, la especialidad en el campo de cada filtrado y un ejemplo de aplicación que se describen en la siguiente tabla:

TABLE III
COMPARATIVA DE LOS TIPOS DE SR

Tipo	Características	Especialidad	Ejemplos
FD	Imprecisas y rara vez son innovadoras	Desarrolla recomendaciones individuales	Netflix
FBC	Simple e interactivos	Determina similitudes entre los componentes visitados y hace recomendaciones basadas en la actividad del sistema.	Apple News Medium
FC	Son precisos y tiene un costo de Producción alto	Maneja la dispersión de los datos y se basan en la actividad del usuario y usuarios	Facebook: Páginas y publicaciones que pueden gustarte

IV. APLICACIÓN DEL SR BASADO EN EL CONTENIDO

Esta sección se encarga de explicar cómo se implementa el SR basado en el contenido. En donde usaremos una cierta cantidad de canciones que han sido recientemente escuchadas por un usuario de Spotify, a partir de estas canciones se generarán recomendaciones. Al final se deberá subir las recomendaciones en una nueva lista de reproducción para que sea escuchadas por el usuario.

DESARROLLO

MODELO DEL DOMINIO

Antes de empezar a programar empezamos definiendo las clases necesarias para la creación del programa.

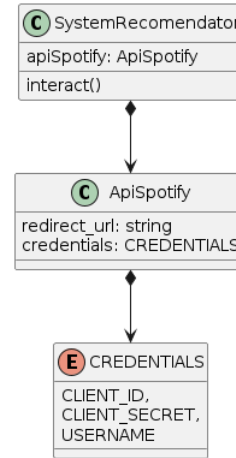


Fig. 12. Modelo del dominio

CONEXIÓN CON LA API DE SPOTIFY

Antes de comenzar con el SR, primero debemos conectarnos con la API de Spotify. Por lo cual necesitamos unas credenciales para poder acceder y extraer información de la plataforma. Las credenciales son las siguientes USERNAME, CLIENT_ID y CLIENT_SECRET que estas se obtienen desde una aplicación creada en el dashboard del sitio Developer Spotify.

TABLE IV
DESCRIPCIÓN DE LAS CLASES DE LA FIGURA 12

SystemRecommendation	clase principal
ApiSpotify	clase responsable de acceder y obtener datos de los servicios de Spotify.
Credentials	un enumerado que contiene la información de inicio de sesión para los servicios de Spotify
Console	clase responsable de solicitar y mostrar información en la línea de comando

La clase encargada para realizar la conexión con la API de Spotify es la `ApiSpotify`. En su interior contiene la librería `Spotipy` para realizar las peticiones a la API. A continuación, se muestra el método que permite la conexión con la API de Spotify.

```

1 def __get_connection(self, scope: str):
2     return spotipy.Spotify(
3         auth=spotipy.util.prompt_for_user_token(
4             username=Credentials.USERNAME.value,
5             scope=scope,
6             client_id=Credentials.CLIENT_ID.value,
7             client_secret=Credentials.CLIENT_SECRET.
8             value,
9             redirect_uri="http://localhost:8080",
10        )

```

Listing 1. Método que realiza la conexión con la API de Spotify

Los *scopes* son los permisos que se requieren para acceder a la API. En este caso, solo se requiere el permiso *user-read-recently-played* para obtener las canciones que recientemente han sido escuchadas. Y para crear la lista de reproducción se necesitará del permiso *playlist-modify-public*.

```

1 self.__spotipy = self.__get_connection("user-read-
2     recently-played")

```

Listing 2. Ejemplo de conexión con la API de Spotify

DESCRIPCIÓN DE LOS DATASETS

Conjunto de datos de las canciones recientemente escuchadas y candidatas

Una vez que tenemos lista la conexión con la API de Spotify entonces conseguimos la información de las canciones que recientemente han sido escuchadas. La Tabla 5 muestra las características utilizadas en este conjunto de datos para identificar cada canción según su aplicabilidad. Las variables de entrada serán las siguientes: *id*, *acousticness*, *danceability*, *duration_ms*, *energy*, *instrumentalness*, *key*, *liveness*, *loudness*, *mode*, *speechiness*, *tempo*, *valence*. La variable de salida será el "id" de cada canción que se recomiende. Las características más destacadas del conjunto de datos se describen en detalle a continuación:

- Acústica sirve para identificar el componente acústico que se usa en la canción.
- Bailabilidad sirve para identificar cuanailable es la canción.
- Energía sirve para identificar si la canción es calmada o inquietante.

TABLE V
DATASET DE LAS CANCIONES RECIÉN ESCUCHADAS Y CANDIDATAS

name	id	acousticness
Red Roses	3acSD0pHBGuVJqdzEwSzt	0.441
danceability	duration_ms	energy
0.75	0.253077	0.415
key	liveness	loudness
8	0.116	-13.931
mode	speechiness	tempo
1	0.0366	129.965

Aclaremos que las canciones candidatas será una lista de todos los artistas que se escucharon recientemente unidas con artistas que recién han estrenado canciones en la plataforma sin importar el género musical. Una vez obtenidos la lista de artistas se conseguirá de cada artista una cantidad de álbumes y luego de cada álbum una cantidad de canciones. También se podría haber hecho de otra manera la generación de canciones candidatas, pero solo mostramos la que nos pareció mejor ya que hacemos una lista de artistas con diferentes géneros musicales.

A continuación, en la siguiente figura mostramos el concepto principal de la implementación del SR.

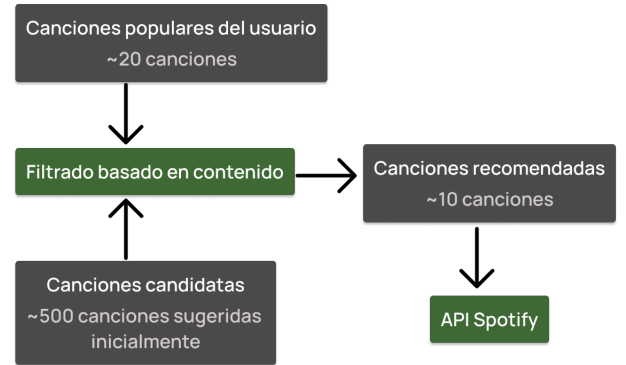


Fig. 13. Sistema de recomendación.

CREACIÓN DEL PROGRAMA

En este apartado se mostrará los métodos más destacados para la creación del programa. Por lo cual se tomará en consideración la clase `SystemRecommendation` y con sus métodos `interact()`, `__set_cosine_similarity()`, `__get_recommendations_tracks()`, que hacen posible la recomendación de canciones.

```

1 def interact(self) -> None:
2     self.__user_tracks = self.__api_spotify.
3         get_top_tracks()
4     self.__candidates_tracks = self.__api_spotify.
5         get_candidates_tracks()
6     self.__api_spotify.create_playlist(self.
7         __get_recommendations_tracks())

```

Listing 3. Método interactuar

Este método (Listing 3) inicia la ejecución del programa. En donde, comienza obteniendo las canciones recientemente escuchadas por el usuario. Asimismo, se obtiene una lista de canciones candidatas. Y por último se crea la lista de reproducción con las canciones recomendadas (Figura 13).

Después que se tienen listas las canciones recientemente escuchadas y candidatas, se procede a realizar la recomendación de canciones por medio del siguiente método.

```
1 def __get_recommendations_tracks(self):
2     self.__set_cosine_similarity()
3     ids_top_tracks = []
4     ids_playlist = []
5     for i in range(self.__user_tracks.shape[0]):
6         ids_top_tracks.append(self.__user_tracks["id"]
7                                "[i])
8         for i in self.__get_candidates_tracks(i, 5):
9             ids_playlist.append(self.
10                                __candidates_tracks["id"]
11                                [i])
12     return list(set([x for x in ids_playlist if x
13                     not in ids_top_tracks]))
```

Listing 4. Método para obtener las recomendaciones

Y como se puede apreciar en el método (Listing 4), se debe tener listo la similitud del coseno (1) así como se explicó en la parte teórica del SR basado en contenido.

```
1 def __set_cosine_similarity(self):
2     scaler = StandardScaler()
3     top_tracks_scaled = scaler.fit_transform(self.
4     __user_tracks.iloc[:, 1:].values)
5     candidate_tracks_scaled = scaler.fit_transform(
6     self.__candidates_tracks.iloc[:, 1:].values
7     )
8     top_tracks_normalized = np.sqrt(
9     (top_tracks_scaled * top_tracks_scaled).sum(
10     axis=1)
11     )
12     candidates_tracks_normalized = np.sqrt(
13     (candidate_tracks_scaled *
14     candidate_tracks_scaled).sum(axis=1)
15     )
16     self.__cosine_similarity = cosine_similarity(
17     top_tracks_scaled
18     / top_tracks_normalized.reshape(
19     top_tracks_scaled.shape[0], 1),
20     candidate_tracks_scaled
21     / candidates_tracks_normalized.reshape(
22     candidate_tracks_scaled.shape[0], 1),
23     )
```

Listing 5. Método para establecer la similitud del coseno.

Para calcular la similitud del coseno primero hay que convertir cada canción con sus características en un vector (Figura 14). Luego, se calcula la similitud del coseno entre los vectores de las canciones recientemente escuchadas por el usuario y las canciones candidatas.

En la Figura 15 se puede observar que contiene una recta roja y verde, en donde la verde corresponde a la canción que recién ha sido escuchada y la roja a la canción candidata. Esta es una manera representativa de cómo funciona la similitud del coseno, ya que como se puede observar que la canción 12 está más cerca de la canción 6 de las canciones que recién se han escuchado. Esto quiere decir, que las canciones candidatas con un ángulo menor son las que

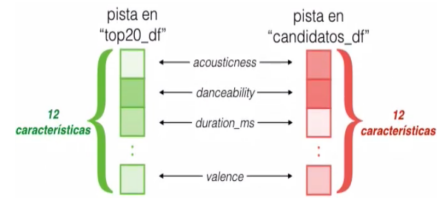


Fig. 14. Similitud de una canción candidata con una canción que recién ha sido escuchada.

tendrán un valor cercano a 1, y mientras que el ángulo sea mayor, las canciones candidatas tendrán un valor negativo, entonces éstas no se las tomará en cuenta. Para la implementación de la similitud del coseno hicimos uso de la librería Sklearn con los métodos `cosine_similarity()` y `StandardScaler()`. Y asimismo para la normalización de los vectores usamos Numpy.



Fig. 15. Cálculo de la similitud del coseno

La ecuación 11 representa los cálculos requeridos para la similitud del coseno.

$$\text{cosineSimilarity} = \text{linearKernel}\left(\frac{\text{topScaled}}{\text{topNormalized}}, \frac{\text{candidateScaled}}{\text{candidateNormalized}}\right) \quad (11)$$

Una vez que se tiene la similitud del coseno, se procede a obtener las canciones recomendadas. En donde todas las canciones que tengan un valor mayor a 0.8 serán recomendadas. El siguiente método es el encargado de obtener las canciones recomendadas.

```
1 def __get_candidates_tracks(self, index,
2     number_candidates, umbral=0.8):
3     result = np.where(self.__cosine_similarity[index
4     , :] >= umbral)[0]
5     result = result[np.argsort(self.
6     __cosine_similarity[index, result])[:, -1]]
7     if len(result) >= number_candidates:
8         return result[0:number_candidates]
9     return result
```

Listing 6. Método para filtrar las canciones candidatas.

Al hacer esto, lograríamos el objetivo de esta práctica. Todo lo que queda por hacer es crear una nueva lista de reproducción y agregarle cada una de las canciones recomendadas. Para hacerlo, utilizamos el siguiente método de la clase `ApiSpotify`.

```
1 def create_playlist(self, recommendations):
2     Console.clean_screen()
3     self.__spotify = self.__get_connection("playlist
4     -modify-public")
5     playlist = self.__spotify.user_playlist_create(
6         user=Credentials.USERNAME.value,
7         name=name,
8         description=description,
9     )
10    self.__spotify.playlist_add_items(playlist["id"]
11    ], recommendations)
```

Listing 7. Método para crear y asignar canciones en una playlist.

V. CONCLUSIONES

Este estudio de SR nos ayudó a comprender cómo las plataformas modernas como Spotify, Netflix, Amazon y otras sugieren productos a los consumidores. Dado que las economías de las plataformas crecen como resultado de los SR, es vital que las plataformas propongan productos.

Por medio de la similitud de coseno se logró encontrar las canciones relacionadas con las canciones recientemente escuchadas por el usuario. Por otro lado, se logró crear una nueva lista de reproducción con las canciones candidatas.

El uso de la librería `Sklearn` nos facilitó la implementación de la similitud de coseno. Asimismo `numpy`, para la normalización de los vectores, `pandas` para la creación de los dataframes y `spotify` para la conexión con la API de Spotify.

La implementación de este sistema de recomendación fue una práctica sencilla, pero que nos permitió aprender a usar las librerías mencionadas y tener una idea de cómo funciona un sistema de recomendación.

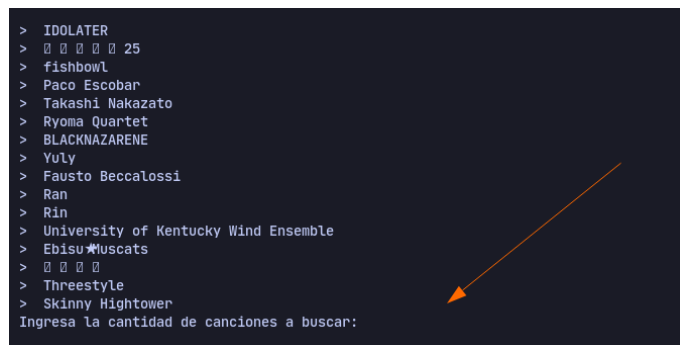
Pudimos solidificar una serie de principios del sistema de recomendación a lo largo de la parte práctica. Por ejemplo, la normalización de vectores, la similitud de coseno, la conexión a la API de Spotify, etc. Nos complace que hayamos tenido éxito en el logro de los objetivos tanto en el aspecto teórico como en el práctico.

REFERENCES

- [1] K. Falk, Practical recommender systems. Shelter Island, NY: Manning Publications Company, 2019.
- [2] H. Weihong and C. Yi, "An e-commerce recommender system based on content-based filtering." Wuhan University Journal of Natural Sciences, 2006.
- [3] Y. Koren, R. Bell and C. Volinsky, Técnicas de factorización de matrices para sistemas de recomendación. 2019.
- [4] S. K. Gorakala and M. Usulli, Building a recommendation system with R: Learn the art of building robust and powerful recommendation engines using R. Birmingham: Packt Publishing, 2015.
- [5] N. Manouselis, Recommender Systems for learning. New York: Springer, 2013.
- [6] G. Uchiyigit and M. Y. Ma, Personalization techniques and recommender systems. Singapore etc.: World Scientific, 2008.
- [7] F. Gedikli, Recommender Systems and the social web leveraging tagging data for recommender systems. Wiesbaden: Springer Fachmedien Wiesbaden, 2013.

- [8] R. Eric. Building Recommendation Systems with Python. Birmingham, England: PACKT Publishing, 2019.
- [9] T. A. Budd, Exploring python. Boston Mass.: McGraw Hill, 2010.
- [10] A. Grigorev and L. Massaron, Machine learning bookcamp build a portfolio of real-life projects. Shelter Island: Manning, 2021.
- [11] B. Paskhaver, Pandas in action. Shelter Island, NY: Manning Publications Co., 2021.
- [12] J. Lu, G.-quan Zhang, and Q. Zhang, Recommender systems: Advanced Developments. Singapore: World Scientific Publishing Co. Pte. Ltd., 2021.
- [13] K. Yasar, Sistemas de recomendación: Métodos de filtrado colaborativo basados en la memoria. Medium, 2018. [Online]. Disponible en: <https://is.gd/tWpY3z>.
- [14] B. Rocca, Introducción a los sistemas de recomendación. Towards Data Science, 2019. [Online]. Disponible en: <https://is.gd/JHXGlf>.
- [15] P. Grover, "Varias implementaciones del filtrado colaborativo". Towards Data Science, 2017. [Online]. Disponible en: <https://is.gd/j4f015>
- [16] C. Bits. Recomendador musical end-to-end — Parte 1: la API de Spotify - YouTube, 2021. [Online]. Disponible en: <https://is.gd/NWv5FZ>.

ANEXOS: EJECUCIÓN DEL PROGRAMA



```
> IDOLATER
> 25
> fishbowl
> Paco Escobar
> Takashi Nakazato
> Ryoma Quartet
> BLACKNAZARENE
> Yuly
> Fausto Beccalossi
> Ran
> Rin
> University of Kentucky Wind Ensemble
> EbisuMuscats
> 25
> Threestyle
> Skinny Hightower
Ingresar la cantidad de canciones a buscar:
```

Fig. 16. Canciones que recién se han estrenado por los artistas



```
> fishbowl
> Paco Escobar
> Takashi Nakazato
> Ryoma Quartet
> BLACKNAZARENE
> Yuly
> Fausto Beccalossi
> Ran
> Rin
> University of Kentucky Wind Ensemble
> EbisuMuscats
> 25
> Threestyle
> Skinny Hightower
Analizando artista: 18 de 88
```

Fig. 17. Buscando canciones estrenadas por los artistas

```

> Paco Escobar
> Takashi Nakazato
> Ryoma Quartet
> BLACKNAZARENE
> Yuly
> Fausto Beccalossi
> Ran
> Rin
> University of Kentucky Wind Ensemble
> Ebisu★Muscats
> ☐ ☐ ☐ ☐
> Threestyle
> Skinny Hightower
Analizando álbum: 46 de 246

```

Fig. 18. Buscando albums de los artistas

```

> Fausto Beccalossi
> Ran
> Rin
> University of Kentucky Wind Ensemble
> Ebisu★Muscats
> ☐ ☐ ☐ ☐
> Threestyle
> Skinny Hightower
Analizando canción: 23 de 818

```

Fig. 19. Buscando canciones de los albums

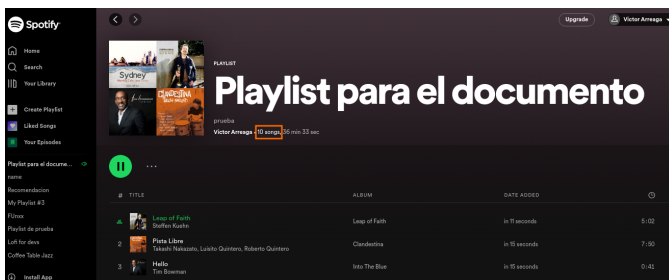


Fig. 20. Lista de reproducción creada con 10 canciones