Técnicas Híbridas en Sistemas de Recomendación

Christopher Guerra Herrero, Amanda Cordero Lezcano, and Alfredo Nuño Oquendo

Facultad de Matemática y Computación, Universidad de La Habana, Cuba

Resumen En este proyecto se desarrollar un sistema híbrido de recomendación de películas utilizando un enfoque monolítico que combina filtrado colaborativo y filtrado basado en contenido. Se emplea el diseño "feature combination", lo cual se aprovecha para proporcionar explicaciones en lenguaje natural sobre las recomendaciones. El sistema se alimenta de un repositorio público de películas de la Universidad Marta Abreu, y se complementa con un sistema de recuperación de películas. La implementación se ha realizado utilizando Vite y React para el *frontend* y Django para el *backend*.
Palabras clave: Sistemas de Recomendación, Filtrado Colaborativo, Filtrado Basado en Contenido, Feature Combination, Python, Django, Vite, React

1. Introducción

En la era actual, los sistemas de recomendación son esenciales para mejorar la experiencia del usuario en diversas plataformas, especialmente en el ámbito del entretenimiento. Este proyecto tiene como objetivo desarrollar un sistema híbrido de recomendación de películas que combine técnicas de filtrado colaborativo y basado en contenido, proporcionando así recomendaciones personalizadas a los usuarios. A lo largo de este informe, se detallarán los aspectos técnicos, la metodología utilizada y los resultados obtenidos en el desarrollo de este sistema.

2. Antecedentes y Estado del Arte

Los sistemas de recomendación tradicionales se dividen en tres categorías principales: filtrado colaborativo, filtrado basado en contenido y filtrado basado en conocimiento. Esta categorización es ampliamente aceptada en la literatura, siendo presentada de manera clara en la taxonomía de sistemas de recomendación de Robin Burke [2].

El *filtrado colaborativo* se basa en las preferencias de usuarios similares, utilizando técnicas como la similitud de vectores o la factorización de matrices para predecir las preferencias de un usuario basado en comportamientos de otros usuarios con intereses similares [1] Este enfoque tiene la ventaja de no requerir información explícita sobre los ítems, pero puede sufrir de problemas como la

escasez de datos (*cold start problem*) y la falta de diversidad en las recomendaciones.

Por otro lado, el *filtrado basado en contenido* utiliza las características de los ítems para hacer recomendaciones. Esta técnica analiza atributos como el género, el director o los actores de una película, y compara estas características con las preferencias históricas del usuario. El filtrado basado en contenido es especialmente útil en contextos donde el historial de comportamiento del usuario es limitado o cuando se busca recomendar ítems que son nuevos en el sistema [1].

Finalmente, el *filtrado basado en conocimiento* utiliza reglas explícitas sobre las necesidades y preferencias del usuario, junto con el conocimiento del dominio, para realizar recomendaciones. Este enfoque es menos común que los anteriores, pero es especialmente útil en sistemas donde es crucial entender el contexto o en dominios donde las preferencias son altamente especializadas y no pueden inferirse fácilmente a partir de datos históricos [2]

Existen diferentes diseños para la implementación de sistemas híbridos de recomendación, cada uno con sus propias ventajas y desventajas:

Diseño Monolítico. En el diseño monolítico, las diferentes técnicas de recomendación (como el filtrado colaborativo y el basado en contenido) se integran en un solo modelo o sistema. Este enfoque es más sencillo de implementar y permite una integración estrecha entre las técnicas, lo que facilita la creación de recomendaciones que combinan múltiples fuentes de datos. Sin embargo, su principal desventaja es que puede ser menos flexible y más difícil de escalar a medida que aumenta la complejidad del sistema o el volumen de datos [2]

Diseño Paralelo. El diseño paralelo emplea múltiples modelos de recomendación de manera independiente y luego combina sus resultados. Cada técnica opera de forma separada, y sus salidas se fusionan para generar la recomendación final. Este enfoque ofrece mayor flexibilidad y permite que cada técnica funcione de manera óptima en su propio espacio. Además, facilita la adición o eliminación de técnicas sin afectar a las demás. No obstante, la combinación de resultados puede ser compleja y requiere un sistema adicional para ponderar o seleccionar las recomendaciones finales [2]

Diseño por Pipelines. En el diseño por *pipelines*, las técnicas de recomendación se aplican en una secuencia, donde la salida de una técnica sirve como entrada para la siguiente. Este enfoque permite refinar las recomendaciones en cada paso, lo que puede resultar en una mayor precisión. Por ejemplo, un sistema puede primero aplicar filtrado colaborativo para identificar un conjunto de ítems recomendados y luego utilizar filtrado basado en contenido para ajustar esas recomendaciones según las características específicas de los ítems. La principal desventaja de este enfoque es que puede ser computacionalmente costoso y complicado de optimizar [1].

En este proyecto, se ha optado por un enfoque híbrido, combinando las técnicas de filtrado colaborativo y filtrado basado en contenido en un sistema monolítico utilizando el diseño de combinación de características (*feature combination*). Este diseño permite aprovechar las ventajas de ambos enfoques y mitigar

sus desventajas. Además, la combinación de características permitió proporcionar explicaciones claras en lenguaje natural sobre las recomendaciones, lo que mejora la transparencia y la confianza del usuario en el sistema.

3. Metodología

3.1. Diseño del Sistema

El sistema desarrollado sigue una arquitectura monolítica en la que el *frontend* y *backend* están fuertemente integrados. El *frontend* se ha implementado utilizando Vite y React, mientras que el *backend* se ha desarrollado con Django, utilizando SQLite3 como base de datos. Además, se realizó un proceso de scraping para obtener la base de datos de películas desde un repositorio público de la Universidad Marta Abreu[3]

3.2. Técnicas Matemáticas Utilizadas

En la implementación del sistema de recomendación, se han empleado varias técnicas matemáticas fundamentales para evaluar la similitud entre ítems, medir la relevancia de términos y gestionar la proximidad entre cadenas de texto. Estas técnicas son cruciales para asegurar la precisión y la eficacia del sistema de recomendación.

Similitud Coseno La similitud coseno es una medida que cuantifica la semejanza entre dos vectores de características, calculando el coseno del ángulo entre ellos en un espacio multidimensional. Esta técnica es especialmente útil en sistemas de recomendación, donde se utiliza para comparar las preferencias de usuarios o las características de los ítems.

Dados dos vectores $\mathbf{A} = (A_1, A_2, \ldots, A_n)$ y $\mathbf{B} = (B_1, B_2, \ldots, B_n)$, que representan las características de dos ítems o usuarios, la similitud coseno $\mathbf{Sim}_{\cos}(\mathbf{A}, \mathbf{B})$ se define como:

$$\mathbf{Sim}_{\cos}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

El valor de la similitud coseno oscila entre -1 y 1, donde un valor de 1 indica que los vectores son idénticos (es decir, el ángulo entre ellos es 0 grados), un valor de 0 indica que son ortogonales (sin relación) y un valor de -1 indica que son diametralmente opuestos.

En el contexto del sistema de recomendación, la similitud coseno se emplea para identificar ítems similares a los que un usuario ha preferido en el pasado, o para encontrar usuarios con preferencias similares a las del usuario objetivo.

TF-IDF El término TF-IDF (Term Frequency - Inverse Document Frequency) es una medida ampliamente utilizada en recuperación de información y minería de texto para evaluar la relevancia de una palabra en un documento dentro de un conjunto de documentos (corpus). En sistemas de recomendación basados en contenido, TF-IDF se emplea para asignar pesos a las características textuales (como títulos o descripciones) de los ítems, lo que permite hacer recomendaciones más precisas.

La frecuencia del término (TF) en un documento se define como:

$$\mathbf{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

donde $f_{t,d}$ es el número de veces que el término t aparece en el documento d.

La frecuencia inversa de documentos (IDF) se define como:

$$\mathbf{IDF}(t, D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

donde $|D|$ es el número total de documentos en el corpus y $|\{d \in D : t \in d\}|$ es el número de documentos en los que aparece el término t.

El peso TF-IDF para un término en un documento se calcula como:

$$\mathbf{TF\text{-}IDF}(t, d, D) = \mathbf{TF}(t, d) \times \mathbf{IDF}(t, D)$$

Esta métrica permite identificar las palabras que son más relevantes para describir un ítem en particular, ayudando al sistema a realizar recomendaciones basadas en características textuales de los ítems.

Algoritmo de Levenshtein El algoritmo de Levenshtein, también conocido como distancia de edición, mide la distancia entre dos cadenas de caracteres. Específicamente, calcula el número mínimo de operaciones (inserciones, eliminaciones o sustituciones de un carácter) necesarias para transformar una cadena en otra. Este algoritmo es útil en sistemas de recomendación para gestionar errores tipográficos y encontrar coincidencias aproximadas entre los nombres de las películas.

Algoritmo en python:

```
def levenshtein(a, b):
  m = len(a)
  n = len(b)

 # Crear una matriz (m+1) x (n+1)
 d = [[0 for j in range(n+1)] for i in range(m+1)]

 for i in range(m+1):
   d[i][0] = i
```

```
for j in range(n+1):
  d[0][j] = j

for i in range(1, m+1):
  for j in range(1, n+1):
    if a[i-1] == b[j-1]:
      coste = 0
    else:
      coste = 1

    d[i][j] = min(d[i-1][j] + 1,    # Eliminación
                  d[i][j-1] + 1,    # Inserción
                  d[i-1][j-1] + coste)  # Sustitución

return d[m][n]
```

El uso del algoritmo de Levenshtein en el sistema de recomendación permite que los usuarios puedan buscar películas incluso si cometen errores tipográficos, asegurando que el sistema sea robusto y eficiente en la búsqueda y comparación de cadenas textuales.

3.3. Sistema de Calificación y Usuarios

El sistema permite a los usuarios calificar películas, lo que se utiliza para generar recomendaciones basadas en usuarios similares. Además, las películas se clasifican por características como género, director y año de producción para las recomendaciones basadas en contenido.

4. Implementación

4.1. Tecnologías Utilizadas

El proyecto utiliza una combinación de tecnologías modernas para su implementación:

- *frontend*: Vite, React, Tailwind CSS.
- *backend*: Django, Django ORM.
- Base de datos: SQLite3.
- Herramientas adicionales: Bibliotecas de Python para scraping y funciones matemáticas.

4.2. Desarrollo del Sistema

La implementación incluyó el desarrollo de varias funcionalidades clave, incluyendo el sistema de búsqueda en la base de datos por nombre, la gestión de usuarios y la integración del sistema de calificación con el motor de recomendaciones.

5. Resultados

5.1. Evaluación del Sistema

El sistema desarrollado es funcional y capaz de proporcionar recomendaciones personalizadas de películas.

6. Conclusiones y Trabajo Futuro

El proyecto presentado ha logrado cumplir con los objetivos propuestos, desarrollando un sistema híbrido de recomendación de películas que integra de manera efectiva técnicas de filtrado colaborativo y filtrado basado en contenido mediante un enfoque monolítico. La implementación del sistema ha demostrado ser funcional y capaz de ofrecer recomendaciones personalizadas con un nivel satisfactorio de precisión. Este éxito se debe en gran medida a la combinación de características (*feature combination*) que permite al sistema proporcionar recomendaciones basadas en la integración de múltiples fuentes de datos, mejorando así la experiencia del usuario.

No obstante, el desarrollo de este proyecto ha permitido identificar varias áreas de mejora y posibles extensiones que podrían incrementar significativamente la eficacia y eficiencia del sistema:

- **Implementación de técnicas paralelas:** Actualmente, el sistema utiliza un enfoque monolítico que, aunque efectivo, puede beneficiarse de la implementación de técnicas paralelas. Estas técnicas permitirían la ejecución simultánea de diferentes métodos de recomendación, lo que no solo optimizaría el tiempo de procesamiento, sino que también podría mejorar la precisión de las recomendaciones al integrar resultados de múltiples algoritmos en tiempo real.
- **Mejora en la captura y limpieza de datos:** La calidad de las recomendaciones está directamente relacionada con la calidad de los datos utilizados. A lo largo del desarrollo, se observó que la captura y limpieza de los datos podrían mejorarse considerablemente. Específicamente, se podrían implementar procesos automatizados de limpieza de datos que eliminen inconsistencias, errores tipográficos y datos redundantes, lo que resultaría en un conjunto de datos más robusto y confiable.
- **Migración a bases de datos más robustas:** Si bien SQLite3 ha sido suficiente para la fase inicial del proyecto, una migración a bases de datos más robustas como MySQL o PostgreSQL sería beneficiosa para soportar un mayor volumen de datos y mejorar la escalabilidad del sistema. Estas bases de datos ofrecen mejores herramientas para la optimización de consultas, manejo de transacciones y soporte de operaciones concurrentes, aspectos críticos para un sistema de recomendación en producción.
- **Expansión del sistema de recomendaciones:** Se podría explorar la inclusión de técnicas más avanzadas de aprendizaje automático, como modelos

basados en redes neuronales o técnicas de *deep learning*, que podrían capturar patrones más complejos en los datos y proporcionar recomendaciones aún más precisas y personalizadas.

- **Integración de nuevas fuentes de datos:** Para mejorar la personalización y relevancia de las recomendaciones, se podría considerar la integración de nuevas fuentes de datos, como información demográfica adicional de los usuarios (edad, ubicación geográfica, etc.), o datos contextuales como la hora del día o la estacionalidad, que podrían influir en las preferencias de visualización.
- **Mejoras en la explicación de las recomendaciones:** Aunque el sistema actual proporciona explicaciones en lenguaje natural sobre las recomendaciones, futuras mejoras podrían enfocarse en hacer estas explicaciones más detalladas y transparentes.

La implementación de estas mejoras no solo fortalecería el sistema actual, sino que también lo posicionaría mejor para adaptarse a las demandas de un entorno de producción más complejo y dinámico.

Referencias

1. Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.
2. Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
3. Universidad Central "Marta Abreu"de Las Villas. Sitio visuales.uclv.cu. Disponible en: https://visuales.uclv.cu.