Simulación de Sistemas de Recomendación para una tienda virtual de música.

Niley Gonzalez Ferrales Arian Pazo Valido

Abstract:
Diseñamos un entorno de simulación de una tienda de música virtual, un sistema de agentes que interactúan con ella, y 2 sistemas de recomendación; un content-based y un knowledge-based. El entorno combina los agentes simulados con los sistemas de recomendación para poner a prueba estos últimos de forma inmediata y sin costos.

Introducción:
El sistema de agentes consiste en agentes utility-based, cuya función de utilidad es escuchar música que sea similar a su preferencia. Utilizando esa similitud, como función de fitness en una metaheurística hill-climbing; los agentes puntúan las canciones que le son recomendadas. El sistema de recomendación content-based utiliza NLP (Natural Language Processing) y heurísticas para determinar que grupo de canciones recomendar a un usuario, basado en el perfil de este.
Por otro lado el sistema knowledge-based utiliza un algoritmo CSP (Constraint Satisfaction Problem) para generar el grupo de canciones a recomendar.
El objetivo de este proyecto es ver como afectan los sistemas de recomendación a distintos tipos de usuarios que modelamos tomando en cuenta la información encontrada en ¨Meassuring Recommender System Effects with Simulated Users¨.
Como conclusión, comprobamos que los agentes más predispuestos a cambios (LooselyPreferenceAgent) son muy volubles ante el sistema y cambian sus preferencias dadas las recomendaciones o las opiniones de otros, por lo que en un sistema de recomendación en el que se tienda a recomendar los items más populares los gustos de estos agentes convergeran a lo que sea más popular. Sin embargo, aunque no tan pronunciado, en los agentes neutrales, se observa una tendencia similar.

1. Sistemas de Recomendación

El objetivo principal de un sistema de recomendación es proponer productos personalizados a usuarios; aprendiendo de su comportamiento y prediciendo sus preferencias para productos particulares.
Facilita el proceso de decisión para clientes, mejorando la experiencia de usuario, ya que ahorra el tiempo de busqueda, mostrando solo productos, que se asumen, son relevantes.
Sin embargo, existe una preocupación creciente de los sesgos que pueden causar estos sistemas; como hacer tender las recomendaciones hacia las canciones más populares o no recomendar muchas canciones nuevas o diversas, ya que es mejor para las empresas recomendar productos 'más seguros'.

En la práctica los sistemas de recommendación se dividen en 3 tipos principales: content-based, collaborative filtering-based y knowledge-based.
Los sistemas de recomendación content-based utilizan la descripción de los productos para predecir su utilidad basado en el perfil de un usuario. Los collaborative filtering-based asumen que usuarios con intereses similares, consumirán productos similares. En los knowledge-based, las recomendaciones se basan en conocimiento existente o reglas sobre las necesidades de los usuarios y las propiedades de los productos.

Los sistemas collaborative filtering-based utilizan, en su mayoría Machine Learning, mientras que nuestra intención era comprobrar sistemas de recomendación que utilicen IA clásica.

1.1 Content-based

Nuestra implementación de content-based está formado por 3 componentes principales: el analizador de contenido, el analizador de perfiles de usuarios y el componente de filtrado.

Analizador de contenido: Extrae la información relevante y estructurada de la música de los datasets. Su principal responsabilidad es representar el contenido de las canciones. Para ello vectoriza las canciones y crea una matriz de frecuencias. Esta matriz tiene como dimensiones: (tamaño del vocabulario de títulos + el número de artistas de la colección + el número de géneros de la colección , cantidad de canciones).

Luego por cada cancion, el vector de su columna en la matriz consiste en un vector de frecuencias, similar a tf*idf. Dada la importancia semántica de los artistas y géneros, cada uno es tratado como un término indexado, aunque no sean una sola palabra. Esta técnica hace que los vectores tengan más información, que resulta útil en NLP (Natural Language Processing).

Analizador de perfiles de usuarios: Analiza la información implicita y explicita de cada usuario para crear un perfil representativo. Dada la lista de canciones del sistema y las que el usuario ha escuchado en la tienda se crea un vector con la misma dimensión que los vectores de una canción, explicados anteriormente. Luego se utilizan las canciones que ha escuchado para calcular las frecuencias del vector del usuario.

Componente de filtrado: Dado un usuario y la matriz de frecuencias se determinan las canciones relevantes para ese usuario; calculando la similitud, utilizando el coseno entre los vectores, de cada canción con su perfil. Estas similitudes son ordenadas decrecientemente y luego son removidas las canciones con similitud 0 y las que el usuario ya ha escuchado.
Cada usuario tiene una cantidad de canciones a recomendarle cada vez, de la lista ordenada por la similitud se seleccionan las primeras dicha cantidad * 3. Y de ellas se seleccionan aleatoriamente cuales recomendar. Esto incluye diversidad en las recomendaciones, ya que incluso para el mismo usuario, en el mismo estado, podemos recomendar un grupo de canciones distintas.

1.2 Knowledge-based

Nuestro sistema de recomendación knowledge-based utiliza como conocimiento las restricciones del CSP.

La primera restricción es que en la lista de recomendaciones, tiene que haber una canción similar al perfil del usuario, o que este en el top de popularidad.

La segunda es que en la lista de recomendaciones, debe que haber una canción cuyo artista o genero esté en la preferencia del usuario.

La tercera es que no se pueden recomendar canciones que ya haya escuchado el usuario.

Y la última, que en la lista de recomendaciones no puede haber canciones repetidas.

Utilizamos como algoritmo de CSP un backtrack con heurísticas. Las variables son las canciones a recomendar, el dominio de cada variable es la lista de canciones.

La primera mejora al backtracking es el hecho de que solo se asignan valores que mantienen la restricción consistente (reduciendo el dominio de la variable a estos valores). Esto permite eliminar caminos en el arbol, que eran fallas seguras.

Además utilizamos degree-heuristic que escoge como próxima variable a asignar la que tiene más nodos adyacentes (vecinos en las restricciones) sin asignar. O sea, la variable involucrada en la mayor cantidad de restricciones con otras variables sin asignar.

El caso base de la recursividad es que todas las variables estén asignadas. Si la asignación no es consistente se revierte.

En otro caso, se obtienen todas las variables aún sin asignar, se seleciona una, a través de las heurísticas explicadas, y se obtiene su dominio consistente (subconjunto de su dominio, donde no se violan restricciones). Luego, se hace un llamado recursivo por cada valor de dicho dominio consistente.

2. Agentes.

Los agentes tienen varios comportamientos comunes. Todos utilizan el método stochasctic_hill_climbing() para evaluar las recomendaciones. Implementamos la metaheurística de forma que genera una asignación inicial entre 1 y 5, y luego en cada iteración, suma ese valor con una variable aleatoria uniforme entre -1 y 1, para variar el valor en su vecindad, si mejora la solucion actual, se actualiza.

La función de fitness a minimizar es $(\text{rate} - \text{utility\_function(song)}*5)^2$. O sea, mientras mayor sea la funcion de utilidad para una canción, mayor será la evaluación.

Además, a cada agente le corresponde uno de los comportamientos ante la música siguientes : Indiferentes (40% de los agentes), Casuales (32% de los agentes), Entusiastas (21 % de los agentes) y Sabios (7% de los agentes) información que tomamos de ¨A Survey of Music Recommendation Systems and Future Perspectives¨. De acuerdo a esta clasificación se definen la mayoría de comportamientos, a través de distribuciones de probabilidad. Por ejemplo, la probabilidad de interacción (publicar o leer en la tienda), de cambiar sus preferencias o de dar información explicita al registrarse en la tienda.

Aun cuando todos los agentes son utility-based, creamos 3 comportamientos distintos:

Uniformly Random Preference: su función de utilidad, dado una canción, es un variable aleatoria uniforme entre 0 y 1. Dado una lista de canciones y sus clasificaciones, si decide cambiar su preferencia (estado interno) se añaden las canciones con clasificación distinta de 0, y el valor de la preferencia se calcula con la función de utilidad.

Loosely Preference: su función de utilidad es una heurística que utiliza la similitud de jaccard en los artistas y géneros de la canción, y devuelve el máximo de ambas similitudes. Para añadir una canción a su preferencia la evaluación de la canción debe ser mayor o igual que 2.5.

Strongly Preference: su función de utilidad es una heurística que utiliza la similitud de jaccard en los artistas y géneros de la canción, y devuelve la media de ambas similitudes. Para añadir una canción a su preferencia la evaluación de la canción debe ser mayor o igual que 4.

3. Tienda simulada

La tienda es un entorno simulado con 3 tipos de eventos: arribo de usuarios, partida de usuarios e interacciones (publicaciones o lecturas de publicaciones). Cada vez que se genera uno de estos eventos se establece el tiempo del próximo, utilizando una variable aleatoria normal con media 5, 5 y 10 respectivamente para cada evento.
Al llegar un nuevo usuario se le recomiendan canciones y se contabiliza la cantidad de cambios que produjo esta acción en el agente.
Si la interacción correspondiente es una publicación (que consiste en una opinión positiva sobre una canción), se añade dicho tema a las publicaciones de la tienda.
Si la interacción correspondiente es una lectura, el usuario lee de 1 a 3 publicaciones, le da un valor aleatorio a cada tema e intenta cambiar su estado interno. Si se produce algún cambio, se contabiliza al igual que cuando se recomiendan nuevas canciones.

4. Modelo de simulación

El modelo recibe como parámetros el recomendador a utilizar, la cantidad de repeticiones y la duración (el horario de cierre de la tienda). En cada iteración se procesa la tienda hasta que cierra y luego se suma por cada tipo de agente la cantidad de cambios en sus preferencias. Al terminar todas las iteraciones se observan los resultados finales mediante métricas de tiempo, y graficando los cambios internos de los usuarios en cada iteración.

References:

1- Artificial intelligence in recommender systems Qian Zhang 1 · Jie Lu 1 · Yaochu Jin 2 2020

2- Measuring Recommender System Effects with Simulated Users Sirui Yao , Yoni Halpern, Nithum Thain , Xuezhi Wang, Kang Lee, Flavien Prost , Ed H. Chi, Jilin Chen, Alex Beutel 2021

3- Developing Constraint-based Recommenders Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach and Markus Zanker 2011

4- A Survey of User Profiling: State-of-the-Art, Challenges, and Solutions Christopher Ifeanyi Eke , Azah Anir Norman, Liyana Shuib, Henry Friday Nweke 2017

5- A Combinatorial Approach to Content-based Music Selection François Pachet, Pierre Roy, Daniel Cazaly

6- A Survey of Music Recommendation Systems and Future Perspectives Yading Song, Simon Dixon, and Marcus Pearce 2012

7- Music Recommender Systems Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas 2015

8- Content-based Recommender Systems: State of the Art and Trends Pasquale Lops, Marco de Gemmis and Giovanni Semeraro 2011

9- Fast Generation of Optimal Music Playlists using Local Search. Steffen Pauws , Wim Verhaegh 2006

10- Artificial Intelligence A Modern Approach, Global Edition Stuart J. Russel, Peter Norving

11- Temas de Simulación Luciano García Garrido , Luis Martí Orosa , Luis Pérez Sánchez