Sistema de Recomendaciones de Filtrado Colaborativo Basado en el Usuario

Katherine R. Uñapilco and Crhistian A. Turpo

I. INTRODUCCIÓN

Un sistema de recomendación proporciona sugerencias personalizadas de *ítems* (productos, servicios, libros, películas,etc) a los usuarios (individuos o empresas) al predecir el interés que el usuario podría tener en base a información relacionada a los usuarios, *ítems* e interacciones entre ítems y usuarios [1]. Hoy en día la cantidad de datos que se genera por segundo en el mundo es inmensa, el objetivo de los sistemas de recomendación es reducir la sobrecarga de esta información recuperando la información mas relevante para un usuario dado. Estos sistemas se han convertido en aplicaciones fundamentales en el mundo de hoy, podemos que empresas como Netflix, Youtube, Google, Facebook que nos sugieren de manera eficiente y acertada ítems relacionados a su plataforma, ya sean vídeos, series, películas entre otros, logrando así satisfacción y conformidad en el usuario, de esta manera se genera fidelidad en el usuario y por ende ganancias para la empresa.

En la actualidad existen muchos artículos relacionados con los sistemas de recomendaciones, cada uno con un enfoque diferente, ya sea basado en el contenido, filtro colaborativo, híbridos. Pero, no hay manera de elegir cual es el mejor y mas prometedor que nos garantice buenos resultados para situaciones especificas.

A. Problema

Dentro de los diferentes tipos de sistema de recomendación presentados en la sección II-A, tenemos a los de filtrado colaborativo basado en el usuario. Este tipo de sistema de recomendación toma en cuenta las calificaciones que los usuarios le dieron a los ítems, donde dos usuarios son considerados similares cuando califican a los ítems de manera parecida. Cuando dos usuarios parecidos son encontrados, los ítems que un usuario califico positivamente, son recomendados al otro y viceversa.

Para el desarrollo del sistema de recomendación de filtrado colaborativo basado en el usuario se presentan los siguiente problemas:

- No se pueden hacer recomendaciones a usuarios nuevos. Ya que al no haber calificado aún, no tendrán vecinos cercanos.
- Se tiene una dependencia entre el estado de la base de datos y las métricas de medida. Se emplean diferentes métricas si la base de datos es densa o si es esparza.
- Cuando se trata con una gran cantidad de ítems y usuarios, el tamaño de la matriz de elementos de usuario se vuelve grande y escaso, por lo que se convierte en un desafío hacer recomendaciones y mantener el rendimiento de las recomendaciones.
- La carga de grandes bases de datos genera un problema de escalabilidad. Se necesitará de más recursos computacionales para satisfacer las nuevas demandas, dado que las matrices con las que se trabaja son demasiado esparzas.
- Las distancias de Manhattan o Euclidean no contemplan todos los casos que se dan entre usuarios para calcular su similitud.
- Las personas cambian de gustos. Se pueden encontrar a un usuario similar a ti, pero basado en tus calificaciones del pasado.
- Se pueden crear usuarios *falsos* con la finalidad de promocionar algún producto en especifico.
- Usuarios *Black Sheep* son usuarios que van en contra del sistema de recomendación.

B. Objetivos

1) Objetivo General: Implementar un sistema de recomendaciones de filtro colaborativo basado en el usuario. Este sistema tiene que ser eficiente, tanto en tiempo de respuesta, como en precisión de resultados. El sistema también debe ser capaz de manejar grandes cantidades de datos.

2) Objetivos Específicos:

- Implementar una arquitectura para que el sistema de recomendaciones sea eficiente y escalable.
- Cargar diferentes bases de datos a la estructura definida para el sistema de recomendación a implementar.
- Implementar diferentes métricas de distancia.
- Implementar el sistema de recomendación con todas las funciones que este requiera.

II. CONCEPTOS GENERALES

A. Sistemas de Recomendación

Los sistemas de recomendación son aquellos que estudian las características de cada usuario y mediante un procesamiento de los datos, encuentra un subconjunto de ítems que pueden resultar de interés para el usuario. Toman preferencias del usuario y generan una recomendación apropiada que satisface al usuario. Las preferencias de los usuarios pueden ser adquiridas implícita y explícitamente.

Existen varios motivos para implementar un sistema de recomendaciones[6]:

- Aumentar el número de*ítems* vendidos.
- Vender un mayor diversidad de *ítems*.
- Aumentar la satisfacción del usuario.
- Aumentar la fidelidad de los usuarios.
- Mejor entendimiento de lo que los usuarios quieren.

B. Taxonomía de los Sistemas de Recomendación

Para implementar la función núcleo de los sistemas de recomendación se tiene que tomar en cuenta que los ítems que seleccionamos para los usuarios tiene que ser útiles. Para ellos existen varios enfoques de como realizar este proceso:

1) Filtrado Colaborativo: Se desarrolla en la sección II-C.

2) Basado en el Contenido: Las técnicas de recomendación basadas en el contenido (CB) son aquellas que recomiendan artículos o productos que sean similares a los artículos previamente preferidos por un usuario específico. Los principios básicos de los sistemas de recomendación de CB son:

- Analizar la descripción de los elementos preferidos por un usuario para determinar los atributos comunes (preferencias) que pueden ser usados para distinguir estos elementos. Estas preferencias se almacenan en un perfil de usuario.
- Comparar los atributos de cada elemento con el perfil de usuario, de modo que solo se recomendarán los elementos que tengan un alto grado de similitud[1].

3) Filtrado Demográfico: Este tipo de sistemas de recomendaciones esta basado en el perfil demográfico del usuario. Los usuarios son considerados similares de acuerdo a parámetros demográficos como nacionalidad, edad, genero, etc [7].

4) Basados en el Conocimiento: Este tipo de sistemas de recomendación ofrece sugerencias basados en el conocimiento que guardan de los usuarios, de los ítems y de como se relacionan entre ellos. Para hacerlo se conserva una base de conocimiento funcional de como un ítem se relaciona con un usuario [1].

5) Basado en Inteligencia Artificial: Este tipo de sistema de recomendación incluye técnicas Bayesianas, redes neuronales, técnicas de *clustering*, algoritmos genéticos y conjuntos de tecnicas *Fuzzy* [1].

6) Híbridos: Los sistemas de recomendación híbridos son la combinación de más de un enfoque. Estos sistemas fueron implementados con la finalidad de sobreponerse a los principales problemas de otros enfoques, como el *cold start*, lo esparza que son las matrices, etc. Otro punto

por el que se implementaron estos sistemas fueron para mejorar la exactitud y eficiencia del proceso de recomendación.

C. Filtrado Colaborativo

Este tipo de sistema trabaja con preferencias del usuario en forma de calificaciones de ítems y explora la similitud con todos los otros usuarios. Este sistema de recomienda un ítem en base a las opiniones de otros. La tarea principal es predecir la calificación que un usuario le daría a un ítem basado en lo que los otros usuarios calificaron a dicho ítem [4]. Los datos se representan en un matriz R, donde $R_{m,n}$ representa la calificación del usuario m al ítem n.

USER-ITEM RATINGS TABLE

Item / User	Item1	Item2	… …	Itemn
User1	R11	R12	… …	R1n
User2	R21	R22	… …	R2n
… …	… …	… …	… …	… …
Userm	Rm1	Rm2	… …	Rmn

Representación de los datos para un sistema de recomendaciones de filtrado colaborativo [5].

Dentro de los sistemas de recomendaciones de filtrado colaborativo tenemos las basadas en el usuario y las basadas en los ítems.

1) Filtrado Colaborativo Basado en el Usuario: El filtrado colaborativo basado en el usuario asume que las personas que tienen gustos similares son parecidas. Al usar esta lógica recomienda ítems encontrando usuarios con puntuaciones similares al usuario activo (a quien estamos tratando de recomendar una película). Es decir, si dos personas valoraron de manera similar un ítem, se asume que gustarán de ítems que el otro no haya valorado. Una aplicación específica de esto es el Algoritmo del vecino más cercano basado en el usuario.

D. Métricas de Distancia

El paso más crítico en las técnicas de filtrado colaborativo basado en los usuarios es el cálculo de la similitud entre ellos. Para esto primero calculamos la similitud entre un usuario i y un usuario j de acuerdo a las valoraciones que le dieron a determinados ítems. Existen varios métodos diferentes para calcular la similitud entre usuarios o elementos.

1) Manhattan y Euclidean: Manhattan es la distancia entre dos puntos medidos a lo largo de los ejes en ángulos rectos.

$$d(x,y) = \sum_{i}^{n} |x_i - y_i| \tag{1}$$

Euclidean es la raíz cuadrada de la sumatoria de diferencias al cuadrado entre las coordenadas y dado por el teorema de Pitágoras.

$$d(x,y) = \sqrt{\sum_{i}^{n} (x_i - y_i)^2} \tag{2}$$

En 1 y en 2, x_i representa el puntaje que el usuario "A" le dio a cierto ítem y y_i representa el puntaje que un usuario "B" le dio al mismo ítem.

- Utilidad: Cuando nuestros datos sean densos, es decir casi todos los atributos tengan asignado un valor, usamos estas distancias métricas.

2) Coeficiente de correlación de Pearson: Indica qué tan lejos están todos los puntos de datos a una línea de mejor ajuste. Es definida por:

$$Pearson(x,y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(x_i - \overline{y})^2}}$$

Un problema al implementar esta fórmula es que requiere de múltiples pases a través de los datos. Por ello, se puede usar en su lugar una fórmula alternativa, que es una aproximación de Pearson:

$$Ap(x,y) = \frac{\sum_{i=1}^{n} x_i y_i - \frac{\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n}}{\sqrt{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}}\sqrt{\sum_{i=1}^{n} y_i^2 - \frac{(\sum_{i=1}^{n} y_i)^2}{n}}}$$

- Valores atípicos: Esta correlación tiene la desventaja de ser sensible a estos valores. Ejemplo:

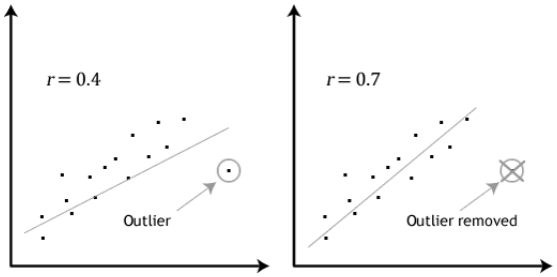

Fig. 1. Valores atípicos

Si los valores atípicos no pueden ser removidos, es mejor usar otra correlación.

- Utilidad: La correlación de Pearson no funciona correctamente para los conjuntos de datos de baja dimensión pero muestra mejor resultados para conjuntos de datos de alta dimensión. [8]
  En minería de datos, si los datos a medir están sujetos a la inflación de grado (diferentes individuos pueden estar usando diferentes escalas) es recomendable usar Pearson.

3) Similitud del Coseno: La similitud de coseno mide la similitud entre dos vectores de un espacio de producto interno. Se mide por el coseno del ángulo entre dos vectores y determina si dos vectores apuntan aproximadamente en la misma dirección. A menudo se utiliza para medir la similitud de documentos en el análisis de texto.

La similitud del coseno se define por la siguiente ecuación:

$$cos(x, y) = \frac{x \cdot y}{||x|| \times ||y||}$$

- Utilidad: La similitud de coseno es una medida de similitud de dos vectores no binarios. El ejemplo típico es el vector de documento, donde cada atributo representa la frecuencia con la que aparece una palabra en particular en el documento. Cada vector de documento es escaso por sus pocos atributos distintos de cero. Por lo tanto, la similitud de coseno es útil cuando se tienen valores dispersos, ya que ignora las coincidencias 0-0.

E. Algoritmo

Para generar las predicciones/sugerencias los sistemas de recomendación usan el total o gran parte de la base de datos. Los usuarios con intereses similares forman parte de un grupo. Identificando a os vecinos mas cercanos nos ayuda a predecir los *items* que se le recomendaran a usuario en cuestión.

1) K-Nearest Neighbors: KNN es el algoritmo mas usado para los sistemas de recomendación de filtrado colaborativo. El articulo de GrupuLens fue introducido por primera vez [7]. Existen dos tipos de KNN:

- Vecinos Cercanos Basados en el Usuario:
  - Usando la medida de similitud seleccionada se calculan los vecinos más cercanos.
  - Para predecir usualmente se usa un promedio, suma de pesos o agregación de pesos.
  - Para obtener las n mejores recomendaciones escogemos los que mayor calificación tengan.
- Vecinos Cercanos Basados en el Item.

Este enfoque investiga el conjunto de ítems calificados por el usuario objetivo y calcula su similitud con el elemento objetivo i y luego elige k los elementos más similares i_1, i_2, ..., i_k.

III. PROPUESTA

Para la implementación del sistema de recomendaciones con filtrado colaborativo basado en usuarios optamos en un primer momento por la programación en el lenguaje *Python*. Para cargar los datos usamos la librería Pandas y para manejarlos fueron almacenados en diccionarios(tablas hash), ya que las consultas en esta estructura son de tiempo O(1).El principal problema que se nos presentó fue la carga de la base de datos. Debido a la naturaleza de los datos, la matriz que los representa es muy esparza y Pandas no maneja de manera eficiente este acontecimiento. Lo que esta librería hace es crear una matriz llena de tamaño nxm, donde n representa al número de usuarios y m representa al número de ítems. Al crear esta matriz, la memoria es la principal afectada debido a que en los lugares donde el usuario no calificó un ítem son llenados con valores *Nan* que ocupan un espacio de memoria innecesario. Otro problema

se presentó al momento de recorrer los datos, con el objetivo de hacer búsquedas en el diccionario, pero al no estar ordenados esta operación no es factible de forma eficaz.

Por lo expuesto anteriormente, se buscó una solución alternativa. Para ello, optamos por la implementación en el lenguaje de programación de C++. Para el manejo de los datos optamos por un *map*, el cual tiene un funcionamiento similar al de un diccionario, pero la implementación interna de este contenedor esta hecha con un árbol binario *Red-Black* el cual tiene una búsqueda de O(log n), permitiéndonos solucionar el problema de recorrido entre sus datos, ya que es ordenado.

A. Arquitectura

Para el manejo de la base de datos se opto por usar el contenedor *map* de la librería estándar *STL* de C++ [9]. *Map* es un contenedor donde cada elemento tiene una llave y un elemento mapeado. Dos elementos no pueden ser mapeados por la misma llave. Aprovechando estas características se diseñó una estructura que represente a una matriz, que es como se trabaja los datos para el sistema de recomendación de filtrado colaborativo. Mediante un *map* de *map*, donde la llave del primer *map* guarda el nombre de los usuarios y el segundo componente guarda otro *map* que a su vez tiene como llave el ítem que el usuario del primer *map* calificó, junto a su respectiva calificación.
Al usar este contenedor se obvian los valores inexistentes, situación que complicaba la carga de datos mediante Pandas en Python, ya que al no encontrar valores la librería mencionada anteriormente llenaba los espacios vacíos con valores nulos o llamados *Nan*, lo cual generaba un problema de almacenamiento. Gracias a los *map*, este problema es solucionado ya que no sobrecarga con valores nulos a la estructura, haciendo que la matriz generada simule una matriz esparza.

```
#include <map>

typedef string IdUser
typedef string IdItem
typedef floar Rating

map<IdUser ,map<IdItem, Rating>>

BD_Structure;
```

El $IdUser$, $IdItem$ y el $Rating$ representan a los usuarios, los ítems y la calificación de los ítems dada por los usuarios respectivamente.

Para la funcionalidad del sistema de recomendaciones, aprovechando las propiedades antes descritas de los *map* de C++, se implementó una clase con los siguientes métodos: *Vecino Cercano*, *RecomendarPorKUsuarios* y *Probabilidad Ítem*. Cada uno de estos métodos con la finalidad de satisfacer las características de un sistema de recomendaciones de filtrado colaborativo basado en el usuario. Para el cálculo de las similitudes entre los usuarios se implementaron las siguientes métricas de distancia: Manhattan, Euclidean, Correlación de Pearson y Similitud del coseno.

1) Vecino Cercano: Sus entradas son: $user$ que es el usuario del cual se calcularan los vecinos cercanos, k el número de vecinos que se obtendrá y $métrica$ que es la métrica de distancia a emplearse.

Este método busca a los k vecinos mas cercanos usando una de las cuatro métricas de distancia implementada. Aprovechando que el *map* es iterable, se recorren por los ítems que los usuarios calificaron O(N) y se obtiene la distancia respectiva entre dos usuarios O(N), donde N es el número de películas que vio el primer usuario. Estas distancias son almacenadas en un vector para posteriormente ser ordenadas y extraer los k primeros elementos, dándonos así la salida deseada.

2) RecomendarPorKUsuarios: Sus entrada son: $user$ que es el usuario al cual se le recomendaran películas, $métrica$ que es la métrica de distancias a emplearse, k que son la cantidad de vecinos que se usaran para los cálculos y $Umbral$ que es el valor mínimo que debe tener un ítem para ser recomendado.

Este método nos devuelve las recomendaciones que le daremos a un usuario basado en los k vecinos más cercanos obtenidos del método

III-A.1. Para ello obtenemos los vecinos más cercanos, y los ordenamos en forma descendente respecto al puntaje y en orden alfabético respecto al nombre del ítem. Posteriormente se obtiene todas los ítems que pasen el umbral, se toma las calificaciones de todos los vecinos para dicho ítem y se hace un promedio que sera la calificación con la que se recomendara el ítem al usuario.

Excepciones a tomar en cuenta: si no se encuentran vecinos cercanos, no se puede recomendar.

3) Probabilidad ítem: Sus entradas son: $user$ que es el usuario del cual se proyectara su calificación, $item$ que es el ítem del cual se hara la proyección de calificación, k el número de vecinos que se usara y $métrica$ que es la métrica de distancia a emplearse.

Este método es usado para calcular la calificación que un usuario, basado en las influencias de los k vecinos mas cercanos, le daría a un ítem que aun no ha calificado. Para esto se calcula los k vecinos mas cercanos que hayan calificado el ítem del cual se quiere saber la calificación que le daría el usuario. Posteriormente se calcula el grado de influencia que cada uno de los vecinos tiene sobre el ítem en cuestión. Finalmente se halla el porcentaje proyectado con una multiplicación de los grados de influencia contra los calificaciones de los vecinos le dieron al ítem.

Excepciones a tomar en cuenta: si un usuario ya vio una película, ya no se le calcula su proyección.

B. Bases de Datos

Las bases de datos fueron preprocesadas para tener un formato estándar.

- Movie Ratings:
  - Originalmente: Conformada por un archivo *"MovieRatings.csv"* en formato de matriz de 26x26 (26 filas con las películas y 26 columnas con los usuarios).
  - Preprocesada: Contiene 441 filas y 3 columnas que almacenan:
    - Usuarios: Especificando el nombre de cada uno.
    - Películas: Especificando el título de las películas vistas por cada usuario.
    - Ratings: Valor entre 1 y 5
- Libros:
  - Originalmente: Conformada por tres archivos: *"BX-Book-Ratings.csv"* (Con calificaciones expresadas en una escala del 1 al 10), *"BX-Books.csv"* (Con el ID del libro, autor, año de publicación, etc.), *"BX-Users.csv"* (con los ID de usuarios, locación y edad).
  - Preprocesada: Contiene más de un millón de registros y 3 columnas que almacenan:
    - Usuarios: Especificando el ID de cada usuario
    - Películas: Especificando el título de las películas vistas por cada usuario.
    - Ratings: Valor entre 1 y 10.
- Movie Lens (2 Bases de Datos)
  - Originalmente: Conformada por seis archivos. Entre ellos: *"ratings.csv"* (Con el ID del usuario, el ID del libro, la calificación con escala de 1 a 5 y un *timestamp*) y *"movies.csv"* (Con el ID de la película, el título y el género).
  - Preprocesada: El primer archivo contiene aprox. 20 millones de registros y el segundo 27 millones de registros. Ambos tienen 3 columnas que almacenan:
    - Usuarios: Especificando el ID de cada usuario
    - Películas: Especificando el título de las películas vistas por cada usuario.
    - Ratings: Valor entre 1 y 5.

IV. Resultados

Para la evaluación del sistema de recomendaciones implementado se tomo dos bases de datos, se puso a prueba los métodos implementados y los resultados se muestran en las siguientes tablas:

Base de datos: MovieRatings.csv			0.001882 s
Características	Esta base de datos contiene pocos registros		
K - vecinos más cercanos User: Patrick C k: 4	Euclidean	0.001583 s	1) Josh - 2 2) Katherine - 2.64575 3) Jessica - 3.31662 4) Vanessa - 3.4641
	Pearson	0.003061 s	1) Ben - 0.697721 2) Chris.1 - 0.666241 3) Chris - 0.639419 4) Greg - 0.59318
	Sim. Coseno	0.013297 s	1) Valerie - 0.872573 2) Ben - 0.819582 3) Zwe - 0.814696 4) Brian - 0.813954
Recomendar películas User: Patrick C k: 4 Umbral = 4	Euclidean	0.006619 s	1) Jaws - 3.33 2) Gladiator - 3.67 3) Village - 3 4) Pulp Fiction - 4 5) Scarface - 4
	Pearson	0.003644 s	1) Gladiator - 4 2) Pulp Fiction - 4.5 3) Alien - 3 4) Scarface - 3.25 5) Jaws - 2.75
	Sim. Coseno	0.007183 s	1) Scarface - 4 2) Gladiator - 4 3) Pulp Fiction - 4 4) Village - 3.25 5) Alien - 4
Calificación Proyectada User: Patrick C k: 4 Item: Village	Pearson	0.004134 s	El puntaje aproximado de Patrick C al ítem: Village es: 2.26423
	Sim. Coseno	0.01322 s	El puntaje aproximado de Patrick C al ítem: Village es: 3.22768

Fig. 2. Resultados del sistema de recomendaciones de filtrado colaborativo basado en el usuario aplicado a una Base de Datos densa donde podemos apreciar que las métricas de Euclidean y Pearson trabajan de manera efectiva mientras que el coseno actúa de manera diferente, esto ayuda a fortalecer el concepto de que la similitud del coseno esta orientado a bases de datos esparzas.

Base de datos: BDLibros.csv			2.43104 s
Características	Esta base de datos es muy esparza		
K - vecinos más cercanos User: 35050 k: 4	Euclidean	2.43344 s	Las distancias de los resultados tienden a ser 0.
	Pearson	5.29875 s	1) 4600 - 0.461539 2) 193499 - 0.390758 3) 239594 - 0.362104 4) 170264 - 0.362104
	Sim. Coseno	8.91524 s	1) 262585 - 0.498974 2) 221960 - 0.407411 3) 93269 - 0.406809 4) 45075 - 0.384777
Recomendar películas User: 276872 k: 4 Umbral: 4	Euclidean	3.18402 s	1) The Hunt for Red October - 10 2) Harry Potter and the Chamber of Secrets (Book 2) -10 3) Harry Potter and the Sorcerer's Stone (Harry Potter (Paperback)) -10 4) 31 resultados más
	Pearson	8.08689 s	1) A Child Called \It\: One Child's Courage to Survive - 10 2) A Gentle Madness : Bibliophiles, Bibliomanes, and the Eternal Passion for Books - 10 3) A Gracious Plenty : A Novel -10 4) 1475 resultados
	Sim. Coseno	13.1635 s	Sin resultados.
Calificación Proyectada User: 276872 k: 4 Item: Oceano Mare	Pearson	8.02049 s	El puntaje aproximado de 276872 al ítem: Oceano Mare es: 8
	Sim. Coseno	13.1814 s	El puntaje aproximado de 276872 al ítem: Oceano Mare es: 8

Fig. 3. Resultados del sistema de recomendaciones de filtrado colaborativo basado en el usuario aplicado a una base de datos demasiado esparza

V. Conclusiones

Al construir un sistema de recomendaciones de filtrado colaborativo basado en el usuario nos damos cuenta que no todos los problemas presentados al inicio pueden ser cubiertos por este enfoque.

El problema de que a los usuarios nuevos no se le puede recomendar películas persiste, ya que para este tipo de sistema de recomendación, la interacción del usuario con los ítems es fundamental. Las personas cambian de gustos, esto genera que puedan salir usuarios similares pero con el comportamiento pasado de un usuario, dando lugar a recomendaciones que ya no siguen el patrón actual del usuario. El sistema puede ser vulnerado por usuarios *falsos* que se creen con la finalidad de corromper el sistema a su favor, calificando ítems muy resaltantes, generando esto que algún ítem que quieran promover sea recomendado a usuarios que han

interactuado con los ítems muy resaltantes. Esto no es solucionado en otros enfoques, por ende la construcción de un sistema de recomendaciones híbrido es necesaria, donde los enfoques presentes en este cubran las desventajas que otros tienen.

Dependiendo del estado (densa, esparza, demasiada esparza) de la base de datos se optara por tomar una u otra métrica para calcular la similitud entre los usuarios. Si la base de datos es densa, se usan métricas como Manhattan o Euclidean, pero si la matriz es esparza usamos la similitud del coseno y si hay diferentes escalas en las puntuaciones usaremos la Correlación de Pearson. De los resultados podemos ver que las recomendaciones cuando se trabaja con grandes cantidades de datos no son aceptables, esto debido a que las métricas de distancia no cubren de manera eficiente el hecho de que la matriz de calificaciones sea demasiada esparza, esto generando problemas en el calculo de las distancias, por ende se tiene que investigar nuevas métricas para solucionar estos problemas.

El cálculo de los vecinos mas cercanos al ser lineal demorara más a medida que la base datos crezca y sea mas densa.

Observando los resultados de las pruebas ejecutadas, se concluye que en bases de datos pequeñas y muy densas las distancias como: Euclidean y Pearson son efectivas. A diferencia de las bases de datos grandes y esparzas, como la de Libros y MovieLens donde es mejor usar la similitud del coseno ya que en lugar de ignorar los items no calificados los reemplaza por 0.

VI. Trabajos Futuros

En los resultados pudimos ver que uno de los principales problemas con respecto al tiempo de respuesta del sistema de recomendaciones es la carga de la base de datos. Dada la naturaleza del proceso que usamos para cargar los datos y la estructura donde son almacenados, podemos apreciar que este proceso es paralelizable, ya sea usando GPU o con el mismo CPU, esto reduciría drásticamente el tiempo de carga de los datos. El almacenamiento en la nube es una opción por la que grandes empresas como Netlix se han inclinado, debido a las características que estos servidores ofrecen, ya sea en almacenamiento o poder de procesamiento mayor. Por ende, explorar estos servicios con la finalidad de optimizar los tiempos y la escalabilidad serán motivo de estudio.

El calculo de los k vecinos más cercanos es otro de los principales problemas que generar demora en la obtención de resultados. Las estructuras métricas manejan de una mejor manera este proceso, entonces se tendrá que estudiar estas estructuras como opción para calcular los KNN.

El sistema de recomendaciones de filtrado colaborativo basado en el usuario tiene falencias que pueden ser cubiertas por otros enfoques, como trabajo futuro queda la construcción de un sistema de recomendaciones híbrido.

References

[1] Recommender system application developments: A survey. Jie Lu, DianshuangWu, Mingsong Mao,Wei Wang, Guangquan Zhang , Australia, 2015.

[2] Research-paper recommender systems: a literature survey. Joeran Beel, Bela Gipp, Stefan Langer and Corinna Breitinger 2015.

[3] Hybrid Recommender Systems: Survey and Experiments. California: Fullerton, CA 92834, 2002.

[4] A Survey of Recommendation System: Research Chanllenges. Lalita Sharma and Anju Gera, 2013.

[5] A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering. SongJie Gong, China, 2010.

[6] Recommender Systems: Introduction and Chanllenges. Francisco Ricci, Lior Rokach, and Bracha Shapira, Italia, 2015.

[7] Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. Pooman B. Thorat, R. M. Goudar, and Sunita Barve, Enero 2015.

[8] Shirkhorshidi AS, Aghabozorgi S, Wah TY A Comparison Study on Similarity and Dissimilarity Measures in Clustering Continuous Data *(2015)*

[9] http://www.cplusplus.com/reference/map/map/at/