UNIVERSIDAD DEL VALLE DE GUATEMALA

CC2006 - Algoritmos y Estructura de Datos

Sección 10

Proyecto 2 fase 1

Oscar Escribá, 21474

GUATEMALA, 07 de Mayo de 2024.

Investigación de algoritmos existentes

El funcionamiento de los sistemas de recomendación ha evolucionado gracias al machine learning. Anteriormente los motores de búsqueda, plataformas de contenido y ventas de producto funcionaban con rankings o listas de popularidad. Los sistemas de recomendación según su tipo de funcionamiento, funcionan gracias a algunos algoritmos que optimizan el análisis de datos para construir las recomendaciones. Uno de los algoritmos más utilizados para regir un sistema de recomendaciones es el conocido como algoritmo de vecinos cercanos. Este determina según los datos proporcionados un patrón de gustos y preferencias y utiliza los datos de un vecino cercano con características similares al inicial y partiendo de estos datos genera las recomendaciones.

Estos datos también pueden ser detectados o calculados con otro algoritmo reconocido como es el de correlación de pearson. Este es un algoritmo de similaridad que recolecta los datos de preferencias de los usuarios y determina un peso de similitud para estimar la relación que existe entre dos usuarios y crear recomendaciones de contenido en base a dicha similaridad. (Graph everywhere, n.d.)

Los algoritmos de recomendación son técnicas computacionales que se utilizan para sugerir elementos relevantes para los usuarios. Dentro de los algoritmos más comunes utilizados en sistemas de recomendación tenemos el filtro colaborativo, Filtrado basado en contenido, sistemas híbridos, factorización de matrices, aprendizaje profundo, modelos de series temporales y algoritmos basados en grafos. Nosotros trataremos de enfocarnos en el algoritmo basado en grafos, los cuales utilizan estructuras de grafos para representar relaciones entre los usuarios, elementos y características, y luego utilizan algoritmos de grafos para realizar recomendaciones.
Este algoritmo es un conjunto de instrucciones que recorren los nodos de un grafo. Estos son muy útiles para modelar varios problemas. Algunos de los algoritmos de grafos más comunes son: Búsqueda de Amplitud o anchura, búsqueda de profundidad, Dijkstra, algoritmo de floy-warshall. (Pelayo, 2023)

Design Thinking

- **Empatía**: en esta etapa de empatía se realizaron una serie de encuestas para interactuar con los usuarios, todo esto con el fin de entender las principales necesidades de los usuarios. Las preguntas fueron basadas en sistemas de recomendación que los mismos han utilizado anteriormente. Para así comprender las verdaderas motivaciones y poder atacarlas directamente. A continuación presentaré las preguntas y las respuestas de las personas con las que platiqué.

Nombre

5 respuestas

Juan Contreras

Kathy Contreras

juan Rios

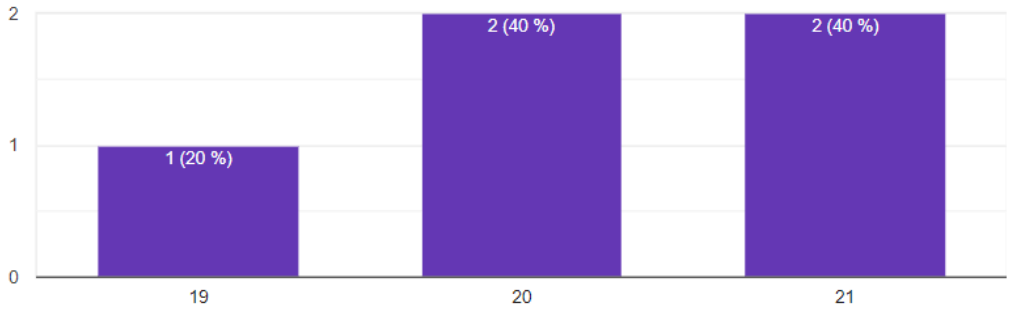
Lucia Rodas

Angela Quezada

Edad

Copiar

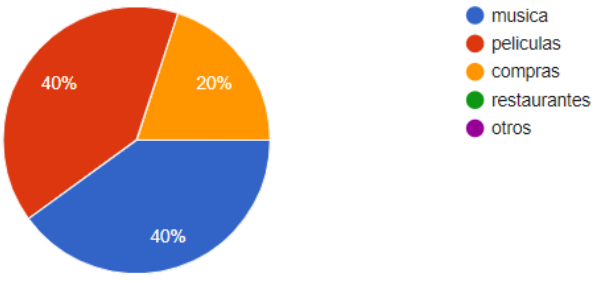
5 respuestas



¿Qué tipo de recomendaciones utilizas más frecuentemente en tus aplicaciones diarias (como música, películas, compras, restaurantes, etc.)?

Copiar

5 respuestas



¿Qué características consideras esenciales en un sistema de recomendaciones para que sea útil y relevante?

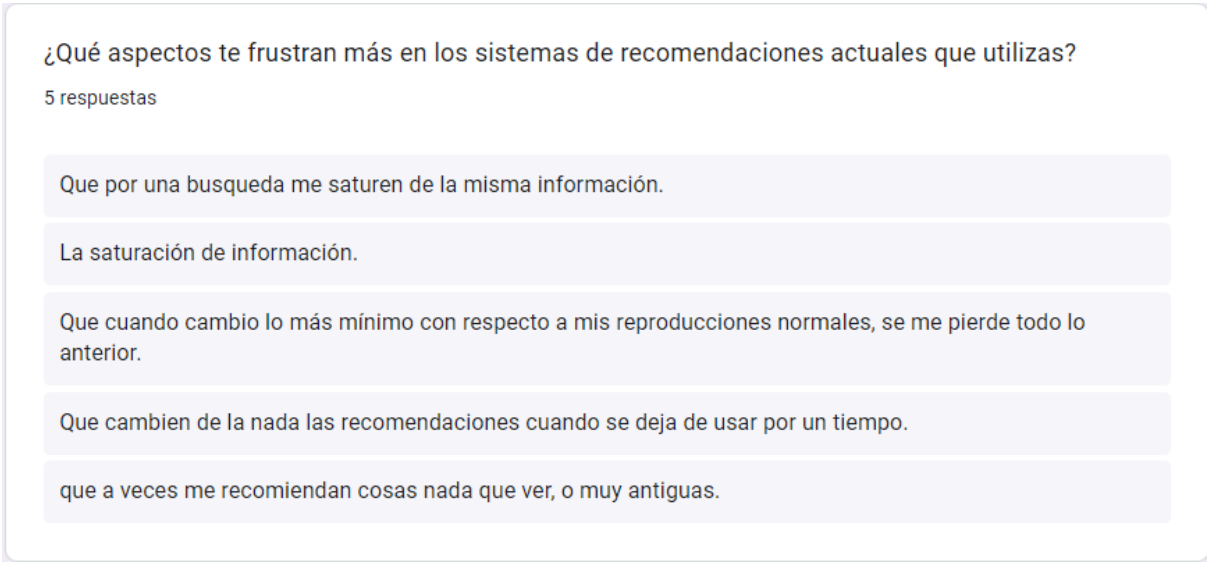
5 respuestas

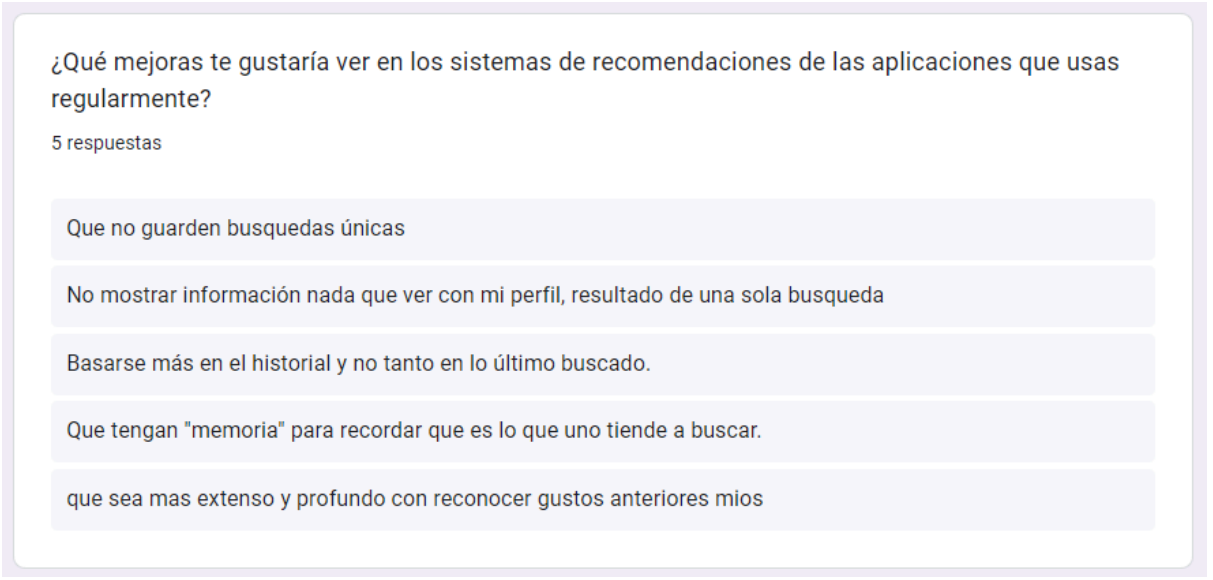
- Busquedas anteriores
- los elementos con los que mas interactuo
- el historial de uso de una aplicación o iformación dentro de una aplicación
- Las últmas busquedas.
- Pues normalmente solo tiendo a utilizar estos sistemas en programas como spotify o netdlix, por lo que piesnso que las útimas reproducciones son lo más importante.

¿Puedes recordar alguna ocasión en la que una recomendación te resultó especialmente útil o sorprendente? ¿Qué fue lo que la hizo tan efectiva?

5 respuestas

- En PedidosYa, cuando intento buscar algo para comer. Siempre me resulta efectivas las recomendaciones que realiza la aplicación al momento de ingresar.
- Hace poco en spotify me salio una nueva canción en mis daily mix, los cuales están hechos con artistas que escucho frecuentemente
- cuando miro netflix siempre va de aceurdo a mis gustos y lo que miro con frecuencia.
- buscando articulos por facebook marketplace, ya que se acoplaba mucho a lo último que había buscado,
- escuchando musica, me ha recomendado artistas que se asemeja mucho al tipo de música que escucho normalmente.





- **Definición**: Luego de conocer las distintas necesidades en la etapa anterior, se definen cuáles son las principales. Luego de determinar las necesidades modulares, será posible plantear medidas para llegar a una solución definitiva.
    - **Discusión acerca de los problemas encontrados:** Como se puede observar en los resultados presentados en la encuesta realizada a los usuarios que decidieron colaborar conmigo en la etapa anterior, los algoritmos de recomendaciones en aplicaciones como Netflix y Spotify son utilizados con gran frecuencia. Estos algoritmos les ayudan a descubrir nuevo contenido basado en sus búsquedas recientes y en el contenido previamente consumido.

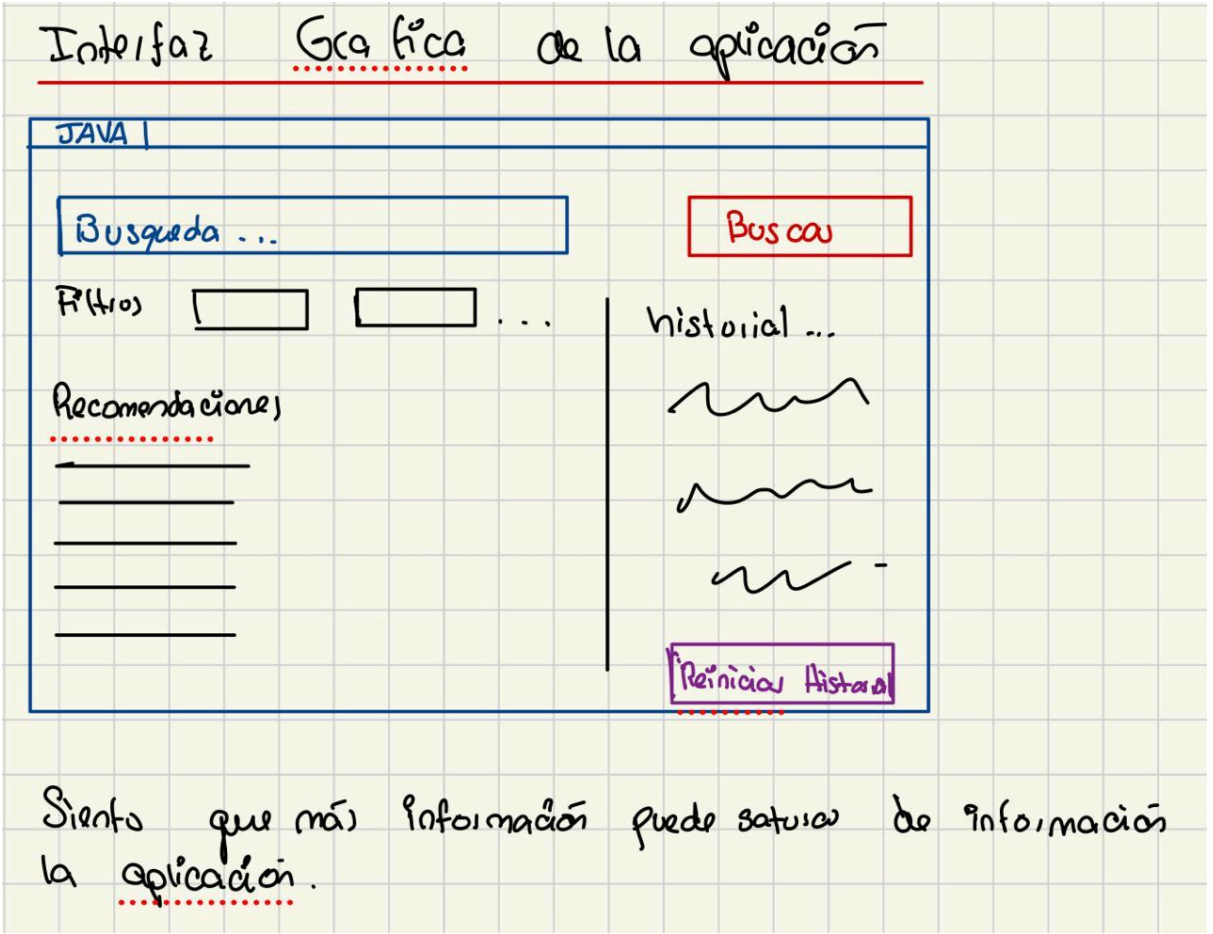
Sin embargo, a pesar de ser herramientas extremadamente útiles, algunos usuarios han expresado ciertas inconformidades respecto a su funcionamiento. Uno de los problemas más mencionados es que el algoritmo tiende a reordenar todo el contenido basado principalmente en la última búsqueda realizada. Esto resulta ser una molestia para muchos, ya que frecuentemente realizan búsquedas ocasionales de contenido que no consumen regularmente. En consecuencia, las recomendaciones generadas no siempre resultan útiles y desplazan el contenido que los usuarios suelen reproducir con mayor frecuencia.

  - **Declaramos el problema:** dentro de los principales problemas tenemos: la reordenación basada en la última busqueda, ya que el algoritmo tiende a reordenar todo el contenido basandose principalmente en la última busqueda. Las recomendaciones no relevantes, el desplazamiento de contenido preferido por saturación de información y la falta de control del usuario.

- **Ideación**: en esta etapa, el equipo debe pensar creativamente y lanzar más de una idea para solucionar aquellos problemas específicos que identificó en la fase previa. En este proceso de pensamiento divergente está permitido equivocarse. Además, para llevarlo a cabo de la mejor forma, se pueden utilizar técnicas para estimular la creatividad y el pensamiento libre.
  Después de haber realizado una lluvia de ideas llegamos a las siguientes soluciones:
    - Peso de las búsquedas: ajustar un algoritmo para que asigne un peso menor a las búsquedas únicas o esporádicas, dando más relevancia al historial de consumo continuo.
    - Preferencias personalizadas.
    - Filtro de recomendaciones: implementar un filtro donde los usuarios puedan elegir ver recomendaciones basadas únicamente en su historial.
    - Etiquetado de búsquedas: etiquetar ciertas búsquedas como ocasionales o frecuentes.

- Prototipo: consiste básicamente en materializar las ideas seleccionadas. En ocasiones, el prototipo puede ser digital (una web beta, por ejemplo) o físico (como un dibujo o diseño).

El prototipo mostrado a los usuarios fue el siguiente:

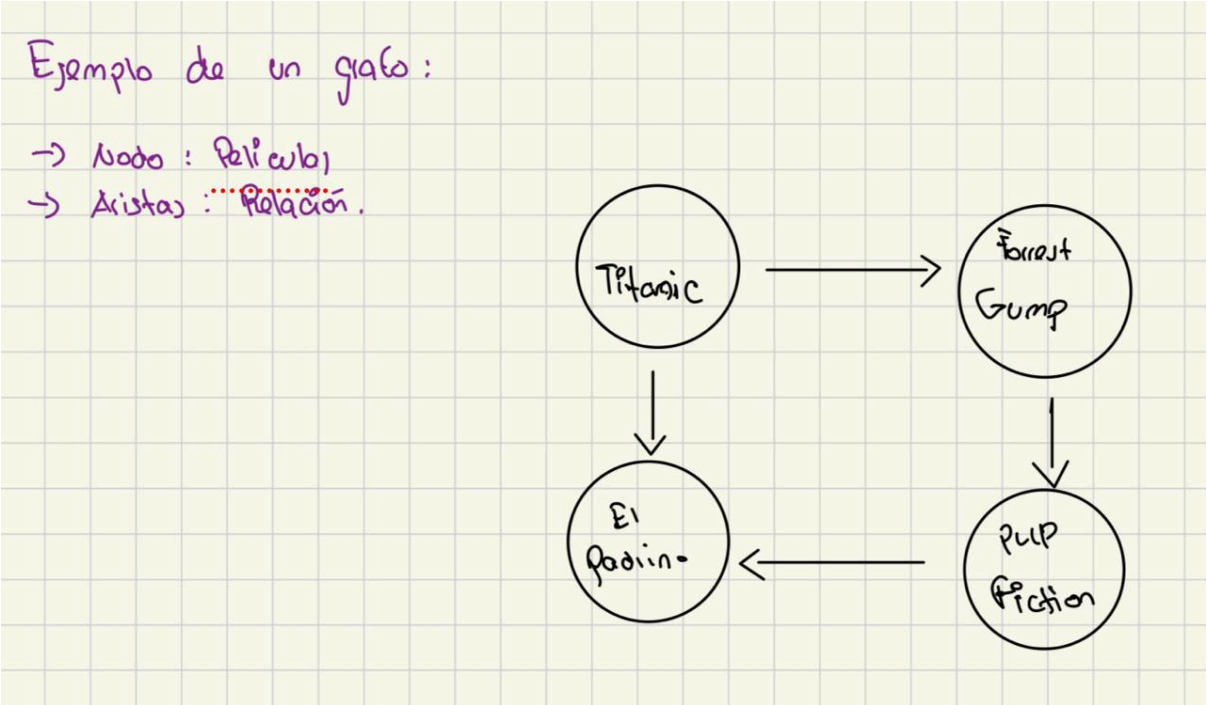


los usuarios mostraron una gran aceptación hacia este prototipo, debido a la sencilla presentación de la información. Contiene todo lo necesario para darle control al usuario acerca de las recomendaciones que este desea recibir.

- **Testing**: Algunos de los cambios solicitados fueron los siguientes: Agregar detalles de las películas y controles de navegación. Todo esto con el fin de tener una interfaz amigable para los usuarios. De lo contrario quedaron encantados con la simplicidad para presentar la información.

Pseudocódigo

Antes de presentar el pseudocódigo me gustaría presentar un ejemplo de grafo, que representa el “pensamiento” de nuestro algoritmo.



El diagrama de grafos representa las relaciones entre las películas en el sistema de recomendación. Cada nodo en el grafo representa una película, y cada arista representa una relación entre las películas.

- Nodos (Películas): Los nodos del grafo son círculos etiquetados con los títulos de las películas. Cada nodo representa una película en la base de datos del sistema.
- Aristas (Relaciones): Las aristas del grafo son las flechas que conectan los nodos. Cada arista representa una relación entre dos películas. En este caso, la relación puede ser la similitud entre las películas, lo que significa que una película puede ser recomendada en función de otra.
- Dirección de las Aristas: Las flechas indican la dirección de la relación entre las películas. Por ejemplo, si hay una flecha de la película A a la película B, significa que la película A está relacionada con la película B, pero no necesariamente al revés.
- Conexiones entre Películas: Las conexiones entre las películas en el grafo representan la similitud o relación entre ellas. Por ejemplo, en el diagrama proporcionado, "Titanic" está conectado con "Forrest Gump", lo que sugiere que estas dos películas están relacionadas de alguna manera, quizás porque comparten un género similar o un actor en común.

- Relaciones Circulares: En algunos casos, puede haber relaciones circulares entre películas, lo que significa que dos películas están relacionadas entre sí en ambas direcciones. Esto puede deberse a similitudes temáticas o elementos compartidos entre las películas.

Ahora si a continuación presentamos lo que vendría siendo una versión muy simplificada del pseudocódigo

```
FUNCION recomendarPeliculas(usuario, soloHistorial)
    peliculasRecomendadas = NUEVA LISTA()

    historialBusquedas = obtenerHistorialBusquedas(usuario)
    pesoTotalBusquedas = calcularPesoBusquedas(historialBusquedas)

    PARA CADA pelicula EN baseDeDatosPeliculas
        similitud = calcularSimilitud(pelicula, historialBusquedas)
        pesoAjustado = similitud * (1 - pesoTotalBusquedas)

        SI pesoAjustado > UMBRAL_SIMILITUD ENTONCES
            AGREGAR pelicula A peliculasRecomendadas
        FIN SI
    FIN PARA

    SI soloHistorial ES VERDADERO ENTONCES
        peliculasRecomendadas = INTERSECCION(peliculasRecomendadas,
obtenerPeliculasHistorial(usuario))
    FIN SI

    RETORNAR peliculasRecomendadas
FIN FUNCION
```

y esta seria la explicación del mismo:
Definición de la función: La función recomendarPeliculas toma como parámetros un usuario y un booleano soloHistorial.
Inicialización de la lista de recomendaciones: Se crea una lista vacía peliculasRecomendadas para almacenar las películas recomendadas.
Obtención del historial de búsquedas: Se obtiene el historial de búsquedas del usuario mediante la función obtenerHistorialBusquedas.
Cálculo del peso total de las búsquedas: Se calcula el peso total de las búsquedas del usuario usando la función calcularPesoBusquedas.

Iteración sobre las películas en la base de datos: Se recorre cada película en la base de datos de películas.
Cálculo de la similitud: Para cada película, se calcula la similitud con el historial de búsquedas del usuario usando la función calcularSimilitud.
Ajuste del peso: Se ajusta el peso de la película basándose en la similitud y el peso total de las búsquedas.
Filtrado por umbral de similitud: Si el peso ajustado de la película supera un umbral definido (UMBRAL_SIMILITUD), se agrega la película a la lista de recomendaciones.
Aplicación del filtro de historial: Si soloHistorial es verdadero, se filtran las recomendaciones para incluir solo las películas que también están en el historial del usuario.
Retorno de las recomendaciones: Finalmente, se devuelve la lista de peliculasRecomendadas.

Base de datos inicial

Actualmente no estoy seguro si aplicaría base de datos directamente o si solamente tendría un programa en JAVA que me esté almacenando la información de las películas, lo cual no consideraría que fuese lo más profesional pero sí lo más práctico. Es por ello que proporcionaré una base de datos SQL, pero haciendo la salvedad que al final puede ser una simple implementación de JAVA.

```
CREATE TABLE Peliculas (
    id INT PRIMARY KEY,
    titulo VARCHAR(255),
    genero VARCHAR(255),
    sinopsis TEXT
);

CREATE TABLE Usuarios (
    id INT PRIMARY KEY,
    nombre VARCHAR(255)
);

CREATE TABLE HistorialBusquedas (
    id INT PRIMARY KEY,
    usuarioId INT,
    peliculaId INT,
    FOREIGN KEY (usuarioId) REFERENCES Usuarios(id),
    FOREIGN KEY (peliculaId) REFERENCES Peliculas(id)
);
```

```
CREATE TABLE RelacionesPeliculas (
    peliculaId1 INT,
    peliculaId2 INT,
    FOREIGN KEY (peliculaId1) REFERENCES Peliculas(id),
    FOREIGN KEY (peliculaId2) REFERENCES Peliculas(id)
);
```