

Explicación de Implementación

1. LINKS:

a. Base de Datos Preprocesada:

https://drive.google.com/drive/folders/1G7y2n5Hz7w_DgzQQJxaRBtGVhMSQonfF?usp=sharing

b. Github:

https://github.com/Kath17/Recommendation_System

c. Reporte Técnico:

<https://es.overleaf.com/read/ysgzphykqkhs>

2. ALMACENAMIENTO Y LEVANTAMIENTO DE LA BASE DE DATOS:

Almacenamiento:

Para el manejo de la base de datos se usó el contenedor **map** de la librería estándar **STL** de C++. **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. Usando un **map** de **map**, donde la llave del primer **map** guarda el nombre de los usuarios y el segundo componente guarda otro **map** que contiene los ítems y la calificación de las películas que vio.

```
typedef string UserId;  
typedef string ItemId;  
typedef float RatingVal;  
  
typedef map < UserId, map <ItemId, RatingVal> > db;
```

Levantamiento de la BD:

Para leer la Base de Datos estamos usando la clase `fstream` de la librería estándar STL, la cual nos permite acceder al archivo.

Si el archivo no se encuentra, se imprime un mensaje de error.

Al existir la Base de datos se lee línea por línea, guardandola en el string **line**.

```

db getBD(string fileName, char delimit, int &ultimo_usuario){

    db BDatos;
    ifstream File(fileName.c_str());

    if (!File) {
        std::cout << "Error, could not open file." << std::endl;
    }

    vector <string> temp;
    temp.reserve(3);
    string line;

```

Cada línea es separada mediante su delimitador “,”, guardando cada parte en un vector llamado “**temp**”. Ejemplo:

line = 4598,”Titanic (1984)”,4.5

temp[0] = 4598

temp[1] = Titanic (1984)

temp[2] = 4.5

El valor de temp[2], que es el rating, es guardado como float en la variable **floatVar**, por lo cual se hace la conversión de string a float.

```

while(getline(File,line))
{
    stringstream lineStream(line);

    temp = split_string_nos(line, delimit);

    string QString = temp[2];
    istringstream StrToFloat(QString );
    float floatVar;
    StrToFloat >> floatVar;

    BDatos[temp[0]][temp[1]] = floatVar;
}

```

```

    string QString = temp[0];
    istringstream StrToInt(QString );
    int floatVar;
    StrToInt >> floatVar;

    ultimo_usuario = floatVar;

    File.close();
    return BDatos;
}

```

Finalmente se agrega cada línea dentro del map de map de la siguiente manera:

```
BDatos[temp[0]][temp[1]] = floatVar;
```

Las líneas de código luego del while sólo sirven para obtener el ID del último registro en la Base de datos.

3. IMPLEMENTACIÓN K-NN

El **K-nn** nos devuelve los k vecinos más cercanos de un usuario. En la implementación tenemos un método que se encarga de esto y se llama *vecino_cercano*. Este método nos devuelve un vector de duplas, donde el primer valor es la distancia de usuarios y el segundo valor es el usuario con el que se está comparando. Este método recibe 3 entradas: *user*, *k*, algoritmo, donde *user* es el usuario del cual se quieren calcular los vecinos más cercanos, *k* es el número de vecinos más cercanos que calculares y algoritmo es la métrica de distancia que se está usando.

Primero creamos un vector llamado *distanciasTodo* que contendrá las distancias con todos los vecinos. Luego se elige la métrica de distancia que se utilizará (Manhattan, Euclidean, Pearson, Similitud del Coseno). Posteriormente se recorre todos los usuarios cuidando que no se haga la comparación consigo mismo, todos los resultados de estas operaciones son almacenadas en *distanciasTodo*. Posteriormente se usa el **sort** de *STL* para ordenarlo en forma creciente si la métrica de distancia es de Manhattan o Euclidean y decreciente si la métrica de distancia es Correlación de Pearson o Similitud de Coseno. Finalmente se crea un vector *distancias* que contendrá la salida con los *K* vecinos más cercanos que son extraídos de *distanciasTodo* que ya es un vector ordenado dependiendo de la

métrica de distancia. Se hace una validación de si n es menor que K , donde n es el número de vecinos cercanos, el K se actualice con el valor de n .

```
vector<tuple<float,string>> vecino_cercano(string user,int k,string algoritmo)
{
    vector<tuple<float,string>> distanciasTodo;
    if(algoritmo == "Manhattan" || algoritmo == "Euclidean")
    {
        if (algoritmo == "Manhattan")
        {
            for(auto users: data)
            {
                if (users.first != user)
                    distanciasTodo.push_back(make_tuple(manhattan(user,users.first),users.first));
            }
        }
        else {
            for(auto users: data)
            {
                if (users.first != user)
                    distanciasTodo.push_back(make_tuple(euclidean(user,users.first),users.first));
            }
        }

        sort(distanciasTodo.begin(),distanciasTodo.end());

        vector<tuple<float,string>> distancias;
        distancias.reserve(k);
        int n = distanciasTodo.size();
        if (n < k)
            k = n;
        for(int i=0; i < k; i++)
            distancias.push_back(distanciasTodo[i]);
        return distancias;
    }

    else if (algoritmo == "Pearson" || algoritmo == "Similitud del Coseno")
    {
        if (algoritmo == "Pearson")
        {
            for(auto users: data)
            {
                if (users.first != user)
                    distanciasTodo.push_back(make_tuple(pearson(user,users.first),users.first));
            }
        }
        else {
            for(auto users: data)
            {
                if (users.first != user)
                    distanciasTodo.push_back(make_tuple(sim_coseno(user,users.first),users.first));
            }
        }

        sort(distanciasTodo.rbegin(),distanciasTodo.rend());
    }
}
```

```

vector<tuple<float,string>> distancias;
distancias.reserve(k);
int n = distanciasTodo.size();
if (n < k)
    k = n;
for(int i=0; i <k; i++)
    distancias.push_back(distanciasTodo[i]);
return distancias;
}
else{
    cout<<"Ingrese una distancia valida"<<endl;
}
}

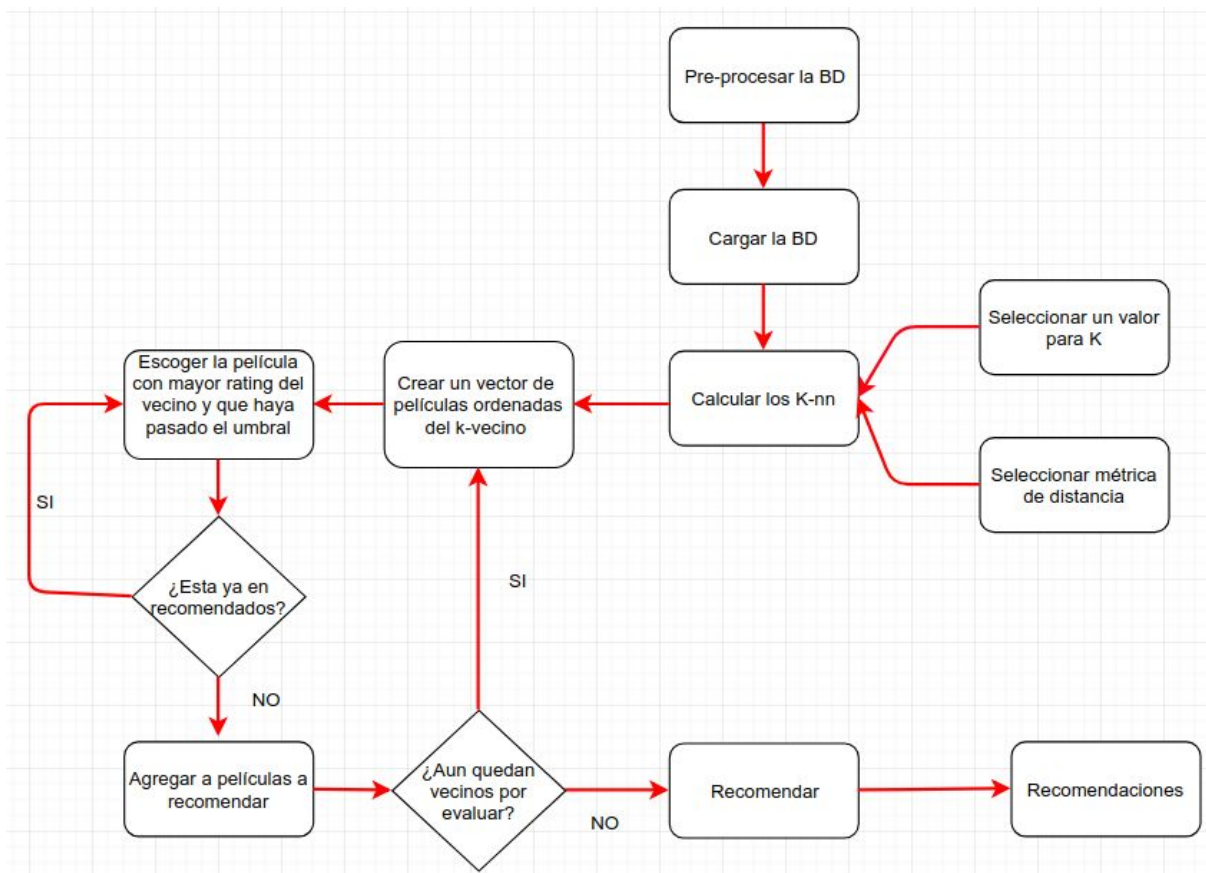
```

4. LENGUAJE DE PROGRAMACIÓN, LIBRERÍAS Y QUÉ HACEN

- Lenguaje de programación: C++
 - En un primer momento se desarrollo con python, pero hubo problemas con el manejo de la data. Entonces se optó por C++ que es más rápido porque es compilado y no interpretado. Además de que nos ofrece **map** que es un contenedor que nos ayuda en el momento de la construcción de la arquitectura.
- Librería: Librería estándar de STL:
 - `#include <map>`
 - Es un contenedor donde cada elemento tiene una llave y un elemento mapeado. Dos elementos no pueden ser mapeados por la misma llave.
 - `#include <vector>`
 - son los mismos que los arreglos dinámicos con la capacidad de redimensionarse automáticamente cuando se inserta o elimina un elemento, y su almacenamiento se maneja automáticamente por el contenedor
 - `#include <tuple>`
 - son objetos que agrupan elementos de posiblemente diferentes tipos en un solo objeto, al igual que los objetos de pares para pares de elementos, pero generalizados para cualquier número de elementos.
 - `#include <algorithm>`

- define una colección de funciones especialmente diseñadas para ser utilizadas en rangos de elementos.
- `#include <fstream>`
 - Clase de flujo de entrada / salida para operar en archivos.
- `sort`
 - Ordena los elementos en el rango [primero, último] en orden ascendente.

5. EL PROCESO QUE SIGUE EL SISTEMA DE RECOMENDACIÓN



6. CAPTURAS

a. RECOMENDACIONES

- i. Usuario: 4598, K= 10, Umbral=4, Distancia: Coseno

Recommendation System

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv ...

Opción:

Distancias:

Se cargó la Base de datos en:
130.293 s

Se realizó la operación en:
158.963 s

Respuesta:
Su Umbral es de :4

Usuario n°1:
4598

Usuario n°2:

Ítem:

K:
10

Umbral:
4.00

Vecino	Distancia	Pelicula	Puntaje
1		12 Years a Slave (2013)	5
2		Birdman: Or (The Unexpe...	5
3		Grand Budapest Hotel, Th...	4.66667
4		Monty Python and the Ho...	4.5
5		Godfather, The (1972)	4.5
6		Catch Me If You Can (2002)	4.5
7		Dark Knight Rises, The (20...	4.3
8		Shawshank Redemption, ...	4.27778
9		Dead Poets Society (1989)	4.25
10		Gladiator (2000)	4.25

ii. Usuario: 16006, K= 5, Umbral=3, Distancia: Coseno

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv ...

Opción:

Distancias:

Se cargó la Base de datos en:
126.822 s

Se realizó la operación en:
518.662 s

Respuesta:
Su Umbral es de :3

Usuario n°1:
30503

Usuario n°2:

Ítem:

K:
5

Umbral:
3.00

Vecino	Distancia	Pelicula	Puntaje
1		Duel (1971)	5
2		American P...	5

iii. Usuario: 30503, K= 5, Umbral=3, Distancia: Coseno

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv ...

Opción:

Distancias:

Se cargó la Base de datos en:
126.822 s

Se realizó la operación en:
518.662 s

Respuesta:
Su Umbral es de :3

Usuario n°1:

Usuario n°2:

Ítem:

K:

Umbral:

	Vecino	Distancia	Pelicula	Puntaje
1			Duel (1971)	5
2			American P...	5

iv. Usuario: 283228, K= 8, Umbral=4, Distancia: Coseno

v.

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv ...

Opción:

Distancias:

Se cargó la Base de datos en:
126.822 s

Se realizó la operación en:
268.665 s

Respuesta:
Su Umbral es de :4

Usuario n°1:

Usuario n°2:

Ítem:

K:

Umbral:

	Vecino	Distancia	Pelicula	Puntaje
1			All Quiet on the Western Fr...	5
2			Adventures of Tintin, The (...)	5
3			Bambi Meets Godzilla (1969)	5
4			All About Eve (1950)	5
5			Apollo 13 (1995)	4.83333
6			Amelie (Fabuleux destin d'A...	4.75
7			2001: A Space Odyssey (1968)	4.7
8			African Queen, The (1951)	4.5

b. K-VECINOS

i. Usuario: 16006 , K = 5 , Distancia: Pearson

Ordenados de mayor a menor

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv

Opción:
 K-vecinos más cercanos

Distancias:
 Pearson

Agregar usuario

Usuario n°1:
 16006

Usuario n°2:

Ítem:

K:
 5

Umbral:
 4.00

Se cargó la Base de datos en:
 126.822 s

Se realizó la operación en:
 900.067 s

Respuesta:

Ejecutar

	Vecino	Distancia
1	53960	0.834625
2	1519	0.803766
3	15065	0.771225
4	268345	0.765131
5	175583	0.763198

Cargar datos

ii. **Usuario: 283228, K = 8 , Distancia: Coseno**

Ordenados de mayor a menor

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv

Opción:
 K-vecinos más cercanos

Distancias:
 Similitud del Coseno

Agregar usuario

Usuario n°1:
 283228

Usuario n°2:

Ítem:

K:
 8

Umbral:
 4.00

Se cargó la Base de datos en:
 126.822 s

Se realizó la operación en:
 261.303 s

Respuesta:

Ejecutar

	Vecino	Distancia
1	173357	0.331771
2	220018	0.305973
3	140027	0.29549
4	61675	0.285821
5	90934	0.284196
6	147607	0.27581
7	112712	0.274997
8	8369	0.27381

iii. **Usuario: 283228, K = 8 , Distancia: Manhattan**

Ordenados de menor a mayor

The screenshot shows the 'Recommendation System' application window. The file path is `/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv`. The 'Opción:' dropdown is set to 'K-vecinos más cercanos'. The 'Distancias:' dropdown is set to 'Manhattan'. The 'Usuario n°1:' field contains '283228'. The 'Usuario n°2:' field is empty. The 'Ítem:' field is empty. The 'K:' field is set to '8'. The 'Umbral:' field is set to '4.00'. The 'Agregar usuario' button is visible. The 'Se cargó la Base de datos en:' field shows '126.822 s'. The 'Se realizó la operación en:' field shows '132.059 s'. The 'Ejecutar' button is visible. The 'Respuesta:' field is empty. A table on the right shows the neighbors:

	Vecino	Distancia
1	242683	62.5
2	117490	75.5
3	63783	77.5
4	191028	86.5
5	99662	89
6	133398	97
7	61614	100
8	123100	103.5

iv. Usuario: 30503 , K = 10, Distancia: Euclidiana

Ordenado de menor a mayor

Por la implementación en el map, y al ser de tipo string, los ID están ordenados de izquierda a derecha.

The screenshot shows the 'Recommendation System' application window. The file path is `/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv`. The 'Opción:' dropdown is set to 'K-vecinos más cercanos'. The 'Distancias:' dropdown is set to 'Euclidean'. The 'Usuario n°1:' field contains '30503'. The 'Usuario n°2:' field is empty. The 'Ítem:' field is empty. The 'K:' field is set to '1000'. The 'Umbral:' field is set to '0.00'. The 'Agregar usuario' button is visible. The 'Se cargó la Base de datos en:' field shows '135.716 s'. The 'Se realizó la operación en:' field shows '605.784 s'. The 'Ejecutar' button is visible. The 'Respuesta:' field is empty. A table on the right shows the neighbors:

	Vecino	Distancia	Pelicula	Puntaje
1	100030	0		
2	100034	0		
3	100042	0		
4	100085	0		
5	100094	0		
6	100156	0		
7	100182	0		
8	100184	0		
9	100208	0		
10	100259	0		
11	100276	0		
12	100299	0		

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv

Opción: **K-vecinos más cercanos**

Distancias: **Euclidean**

Agregar usuario

Usuario n°1: 30503

Usuario n°2:

Ítem:

K: 1000

Umbral: 0.00

Se cargó la Base de datos en: 135.716 s

Se realizó la operación en: 605.784 s

Respuesta:

Ejecutar

	Vecino	Distancia	Pelicula	Puntaje
13	100305	0		
14	100354	0		
15	100389	0		
16	100414	0		
17	10042	0		
18	10043	0		
19	100437	0		
20	100446	0		
21	100471	0		
22	100475	0		
23	100477	0		
24	100480	0		
25	100490	0		

Cargar datos

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv

Opción: **K-vecinos más cercanos**

Distancias: **Euclidean**

Agregar usuario

Usuario n°1: 30503

Usuario n°2:

Ítem:

K: 1000

Umbral: 0.00

Se cargó la Base de datos en: 135.716 s

Se realizó la operación en: 605.784 s

Respuesta:

Ejecutar

	Vecino	Distancia	Pelicula	Puntaje
25	100490	0		
26	100542	0		
27	100599	0		
28	100609	0		
29	100611	0		
30	100623	0		
31	100700	0		
32	100719	0		
33	100731	0		
34	100771	0		
35	100773	0		
36	100792	0		
37	100810	0		

Cargar datos

v. **Usuario: 4598** , **K = 10**, **Distancia: Pearson**
 Ordenados de mayor a menor

MainWindow

/home/kat/Documentos/Recommendation_System/RecommendationSystem/BD/BDMovieLens_27m.csv

...

Opción:

K-vecinos más cercanos

Usuario n°1:

4598

Distancias:

Pearson

Agregar usuario

Se cargó la Base de datos en:

126.822 s

Se realizó la operación en:

153.175 s

Respuesta:

Usuario n°2:

Ítem:

K:

10

Umbral:

4.00

Ejecutar

	Vecino	Distancia	
1	58152	0.973225	
2	199302	0.97313	
3	254456	0.971132	
4	278653	0.967643	
5	215490	0.965582	
6	127451	0.965435	
7	66471	0.962177	
8	53302	0.960696	
9	166710	0.952486	
10	238991	0.948594	

Cargar datos