Informe: Sistema de recomendación

Margarita Lacuaña Luis Bernal

Mayo 2019

1. Introduction

La aparición y el crecimiento de los mercados en línea han tenido un impacto considerable en los hábitos de los consumidores, al brindarles acceso a una mayor variedad de productos e información sobre estos productos. Si bien esta libertad de compra ha convertido al comercio en línea en una industria multimillonaria, también dificulta a los consumidores seleccionar los productos que mejor se ajusten a sus necesidades. Una de las principales soluciones propuestas para este problema de sobrecarga de información son los sistemas de recomendación, que brindan sugerencias automatizadas y personalizadas de productos a los consumidores.

Este trabajo presenta la implementación de un sistema de recomendación basado en filtro colaborativo. Utilizamos el algoritmo de los K-Vecinos más cercanos*(en inglés k-nearest neighbors - KNN)*

2. Detalles de implementación

2.1. Lenguaje Utilizado

Se utilizó el lenguaje C++, en su especificación C++ 17, junto con su librería estándar STL para el uso de "maps", "multimaps", "pairs" y "lists".

2.2. Almacenamiento

Se realizaron las pruebas con la base de datos de *movielens* de 27 y 20 millones. Se almacenaron en memoria principal ocupando:

	Memoria ocupada	Tiempo de carga
20 Millones	2.6 GB	1 min 27 seg
27 Millones	3.3 GB	2 min 20 seg

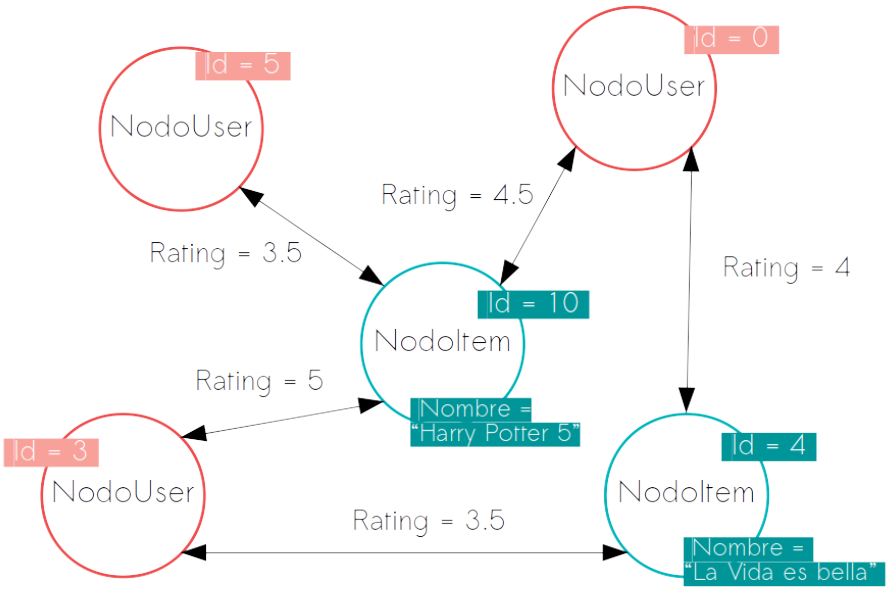

Figura 1: Representación de la estructura planteada

2.3. Estructura

El enfoque utilizado en esta implementación es un grafo donde existen dos tipos de nodos: NodoItem y NodoUser Fig.1. Donde:

- **NodoUser** tiene como dato miembro id y map con clave puntero a NodoItem y valor el rating.
- **NodoItem** tiene como datos miembros id, Nombre y map con clave puntero a NodoUser y el valor del rating.

Al tener NodoUser y NodoItem un map con punteros hacia los dos y los ratings permite moverse a travez del grafo en cualquier dirección.

La estructura básica de grafos está inspirada en el gestor de base de datos Neo4j [1].

```
class Grafo {
public:
    Grafo();
    map<int, NodoUser*>    index_users;
    map<int, NodoItem*>    index_items;
```

Así cada nodo es indexado usando su id, permitiendo la búsqueda de cualquier user o item en orden O(log(n))

Además cada NodoUser y NodoItem tiene la forma:

```
class  NodoUser{
  public:
    int id ;
    //puntero a nodo item, float rating
    map<NodoItem * ,float>   items; //Relaciones
    NodoUser(int id);
```

y el NodoItem:

```
class  NodoItem {
    public:
      int id;
      string nombre;
      map<NodoUser *,float >  users;
      NodoItem(int);
```

2.4. KNN

Por lo tanto realizar un KNN quedaría de la siguiente forma:

1. Encontramos el NodoUser deseado, en el index_users(Retorna un puntero a NodoUser).
2. A partir del nodo, recorremos todos sus items.
3. Para cada item, recorremos todos los usuarios que han visto ese item. A su vez se van almacenando los ratings.

```
typedef list<pair<float ,NodoUser*> > k_vec; //k vecinos

void NodoUser::knn(int k,int dist, k_vec &k_vecinos_cercanos){
  map<NodoUser*, list_ratings_xuser > common_users;
  for (auto & item : this->items ){
  //Recorremos todos los items del usuario (this)
  float rating1 = item.second;
     for (auto & user: item.first->users){
     //Recorremos todos los usuarios que han visto el item
    float rating2 = user.second;
    auto it = common_users.find(user.first);
    if(it!=common_users.end()){
    // Y los anadimos a la lista de vecinos comunes ( Common_users )
        it->second.push_back(make_pair(rating1,rating2));
    }
    else{
        list_ratings_xuser temp;
        temp.push_back(make_pair(rating1,rating2));
        common_users[user.first] = temp;
```

```
    }
  }
}
```

2.5. Librerías utilizadas

- Pistache C++: Librería para crear y recibir conexiones HTTP. [2]
- Rapidjson: Librería para parsear JSONs recibidos con el método POST. [3]

3. Distancia Propuesta: Coseno Soft

Dados 2 vectores A, B. con n atributos, donde los elementos de A son diferentes de cero.

$$cos(\theta) = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

4. Interface

Se implementó una interface utilizando HTML, CSS y JS. Utilizando la librería Vue.js para renderizar. Se muestra la interface en la Fig. 2

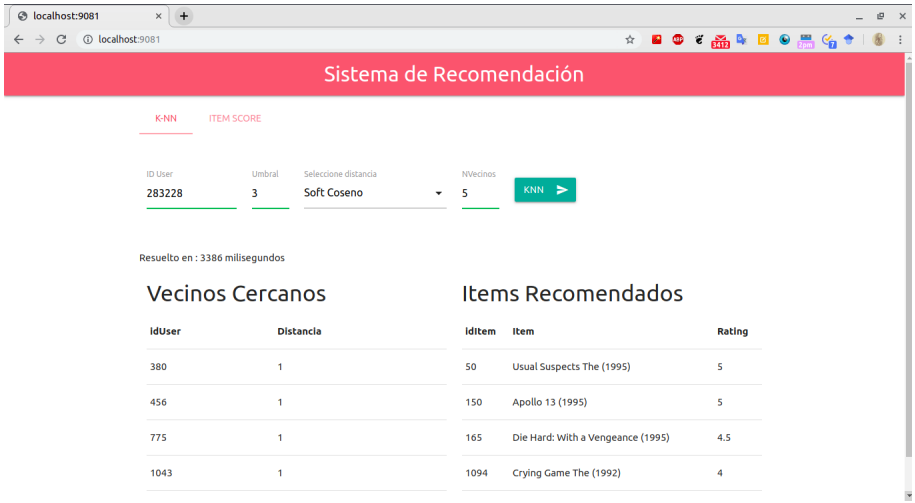

Figura 2: Interface

Referencias

[1] Neo4j. https://neo4j.com/.

[2] Pistache c++. http://pistache.io. Accessed: 2018-05-6.

[3] Rapidjson. http://rapidjson.org/. Accessed: 2018-05-6.