Algorismes i estructures de dades

K-means

Aquest algorisme ens serveix per separar els usuaris del sistema en k clústers segons el grau de "proximitat" entre les seves valoracions. En primer lloc, s'assigna a cada usuari un punt en un sistema de coordenades on cada valoració d'un ítem es correspon amb una dimensió. Després, a cada un dels k clústers se li assigna un centroide en el mateix sistema de coordenades (primerament s'agafen les coordenades de k usuaris a l'atzar). A continuació, per a cada usuari es mesura la seva distància als centroides i s'assigna aquell usuari al clúster més proper. Una vegada s'ha fet la primera iteració, es recalculen les coordenades dels k centroides fent la mitjana (d'aquí, k-means) de les coordenades de tots els usuaris que conté el seu clúster. Aquest procés es repeteix fins que els clústers son estables, llavors es considera que cada usuari està al clúster que li correspon i l'algorisme acaba retornant els clústers amb els seus usuaris i els corresponents centroides.

Principals estructures de dades:

- **HashMap<String, Vector<Double>> CoordenadesUsuaris**
    - *És un map que conté, per a tots els usuaris del sistema, les seves coordenades. La clau és l'identificador d'un usuari, i al vector s'emmagatzemen les seves coordenades.*
- **HashMap<String, Vector<Double>>[] clusters**
    - *És un array de k maps on cada un dels maps es correspon amb un clúster. La clau de cada entrada d'un map és l'identificador d'un usuari, i al vector s'emmagatzemen les seves coordenades.*
- **Vector<Double>[] centroides**
    - *És un array de k vectors on cada un dels vectors representa les coordenades d'un centroide, de manera que centroides[i] conté les coordenades del centroide que correspon a clusters[i].*
- **Vector<UsuariActiu>[] clustersUsuaris**
    - *S'usa per fer el retorn de l'algorisme. Consisteix en un array de k vectors d'usuaris, un per a cada clúster. El contingut de clustersUsuaris[i] són els usuaris identificats per les mateixes claus que hi ha a les entrades de clusters[i].*

Slope one

Aquest algorisme s'usa per estimar o predir la valoració que un usuari en concret donaria a un ítem donat, a partir de les valoracions que altres usuaris han fet d'aquell ítem. Es complementa amb l'algorisme K-means esmentat anteriorment, de manera les valoracions de l'ítem que s'agafen com a referència siguin les dels usuaris que estan en el mateix clúster (per tant, amb gustos similars). Primerament es calcula, per al usuari donat, a quin clúster pertany amb la funció calculaParticio. Un cop es té el cluster al que pertany, per cada item del map entrat per paràmetre (i per tant per cada item per al que es vol deduir una valoració) es fa el càlcul de slope one descrit a l'enunciat.

Principals estructures de dades:

- **Vector <UsuariActiu>[] clusters** /
    - *S'usa per trobar la pertinença del usuari donat als diferents clústers. Consisteix en un array de k vectors on cada un dels vectors es correspon amb un clúster, i conté les diferents instàncies d'usuaris que pertanyen a la partició. La clau de les entrades és l'identificador d'un usuari.*

- **Vector<Double>[] centroides**

    - *És un array de k vectors on cada un dels vectors representa les coordenades d'un centroide, de manera que centroides[i] conté les coordenades del centroide que correspon a clusters[i].*

- **Vector<UsuariActiu> uparticio**

    - *És un vector d'UsuarisActius que representa els usuaris que pertanyen a la mateixa partició que l'usuari entrat per paràmetre a Slope One.*

- **TreeSet<Valoració> result**

    - *Aquesta estructura és la que s'usa per el return de Slope One. Representa un conjunt de Valoracions predites per a l'usuari entrat per paràmetre. Tot i que bastaria retornar els id dels items i després ordenar-los per la puntuació predita, es retorna un TreeSet de Valoracions. Una valoració té, un usuari actiu, un item i una puntuació, d'aquesta manera retornant un objecte (dins d'un TreeSet) podem obtenir tota la informació necessària per si més endavant es vol treballar amb la valoració. El TreeSet s'usa per poder tenir les Valoracions ordenades, per puntuació, de major a menor. De manera que no cal ordenar més tard els ordre en que es retornen els items.*

K-nearest neighbours

Aquest algorisme ens serveix per cercar, per a cada ítem que li agrada a l'usuari d'entrada, els k ítems més semblants a aquests. Un cop té els k ítems més semblants per cada ítem que li agrada, calcula una Valoració predictiva per aquest ítem semblant, tenint en compte la similitud i la nota de l'ítem agradat per l'usuari, essent la Valoració final la nota de l'ítem agradat afegint-li una desviació tipus calculada a partir de la similitud, de manera que a similitud màxima, la nota seria la mateixa que l'agradat, i a similitud mínima (no s'assemblen en absolutament res) seria una Valoració aleatòria. Finalment s'ordenen per puntuació aquestes valoracions i es retornen.

Principals estructures de dades:

- **Map<String,Valoració>**
    - Aquesta estructura de dades s'utilitza perquè és l'atribut que conté l'usuari actiu per tal d'accedir a les valoracions d'aquest. El que es fa a l'algorisme es simplement recorrer-ho per tal de trobar les millors valoracions.
- **Vector<Valoració>**
    - Aquest vector és simplement per emmagatzemar les millors valoracions de l'usuari actiu per després tractarles.
- **HashMap <Item, Pair<Integer, Double> >**
    - Aquest HashMap serveix per emmagatzemar, per tots els items millor valorats per l'usuari actiu, els k més semblants amb la seva similitud (el double) i, com que és possible que un altre item sigui semblant a aquest, l'integer és el nombre de repetits. És fa servir aquest HashMap per maximitzar la eficiència de les cerques per detectar repetits.
- **PriorityQueue< Pair<Double,Item> >**
    - Utilitzem la priority queue per, primerament ficar tots els items amb els que es compara l'item comparat i, finalment, extreure els k més semblants, amb l'aventatge d'aquesta estructura de que el pas de extreure els k més semblants és de cost O(k) i que les insercions també són prou barates en temps.

- **TreeSet<Valoració>**
    - Finalment fiquem les valoracions a un TreeSet per tal de que quedin ordenades per puntuació i les retornem al metode de Filtering.

Valorar recomanació (DCG)

El DCG (Discounted Cumulative Gain) s'usa per mesurar la qualitat d'una recomanació oferida per un algorisme a un usuari, consistint aquesta en una llista dels ítems "recomanats". Aquesta llista d'ítems se suposa ordenada de forma descendent, segons la puntuació amb què l'algorisme prediu que l'usuari valoraria aquell ítem. Per aquest algorisme és necessari conèixer la valoració "real" que hauria de predir l'algorisme per als ítems de la llista. Per calcular el dcg, hem usat la fórmula següent:

$$DCGp = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

On rel_i és la puntuació real de l'ítem que està en la posició ***i*** a la llista de recomanacions. El nombre d'ítems de la llista de recomanats és ***p*** i, ***i*** correspon a la posició de dit element a la llista començant per 1. Aquest sumatori serà major com més encertada sigui l'ordenació dels ítems a la llista de recomanació (de manera que es penalitza si un element amb puntuació "real" baixa és recomanat abans a la llista que un altre ítem amb puntuació "real" major).

Principals estructures de dades:

- **Vector<String> itemsRecomanats**
    - *És un vector que conté els noms del ítems recomanats per un algorisme a un usuari, ordenats descendentment segons la puntuació que prediu que aquest usuari els donaria.*
- **Map<String, Valoració> LT**
    - *És un map on la clau és el nom d'un ítem i que conté les valoracions "reals" amb què l'usuari els puntuaria.*

Altres estructures de dades rellevants del sistema:

També hi ha altres estructures de dades rellevants que s'usen en diferents classes del sistema (la majoria, dins del controlador de la capa de Domini), i són les que s'esmenten a continuació:

- **Map<String, Item> Items**
    - *És un map on la clau és l'identificador d'un ítem i el valor és l'Item en qüestió.*
- **Vector<Columna> Columnes**
    - *És un vector on es guarden les Columnes corresponents als Atributs que defineixen un Item. És a dir, per a cada Atribut diferent que pot tenir un Item, hi ha un objecte Columna que l'emmagatzema.*
- **HashMap<String, UsuariActiu> Usuaris**
    - *És un map on la clau és l'identificador d'un usuari i que conté tots els usuaris que apareixen al fitxer de ratings.*
- **HashMap<String, UsuariActiu> UsuarisKnown**
    - *És un map on la clau és l'identificador d'un usuari i que conté tots els usuaris que apareixen al fitxer de Known.*
- **HashMap<String, UsuariActiu> UsuarisUnknown**
    - *És un map on la clau és l'identificador d'un usuari i que conté tots els usuaris que apareixen al fitxer d'Unknown. Tot i que l'usuari en si apareix al map de UsuarisKnown, en aquest map la instància té unes valoracions otorgades diferents, les de unknown. Aquesta manera de guardar la informació de unknown ens permet un accés fàcil a l'hora d'accedir per usuari a les dades que es troben en el fitxer. Serveix per diferenciar clarament quina informació es "coneguda" d'un usuari i quina es "desconeguda".*
- ***Pair<T,T>***
    - *Durant la programació de les funcionalitat vam trobar que l'ús de pairs facilitaria molt la feina a fer i faria més fàcil els retorns de algunes funcions. És per això que es va crear una classe Pair genèrica.*