Descripció del diagrama de classes

Classe Persona (+ subclasses)

Aquesta classe fa referència a les persones que usaran la app. Aquestes persones poden ser o bé Admin (administrador) o bé usuaris actius del sistema. Les persones tenen un nom d'Usuari, que les identifica, i una contrasenya. Atributs amb els quals podrà iniciar sessió.
L'atribut usuari es string ja que, si fos int, seria més difícil recordar per una persona quin és el seu usuari. D'aquesta manera permetem que usin qualsevol nom d'usuari més senzill.

Les operacions de la superclasse són bàsicament getters i setters dels seus propis atributs i operacions abstractes que implementaran les subclasses.

De la subclasse UsuariActiu l'operació més complexa és:

- getCoordenades(Map<String,Item> allitems) :: Vector<Double>
    - *Retorna el vector amb les coordenades d'un usuari, en un sistema de coordenades on cada ítem és una dimensió i la valoració per part de l'usuari d'aquell ítem és el seu valor. Per als ítems no valorats retorna un valor a l'atzar dintre del rang acceptable de valoracions.*

A la subclasse Admin trobem també les implementacions d'algunes operacions abstractes de Persona:

- getRecomanacio(Recomanacio r, UsuariActiu ua, Map<String,Item> cjtItems, int n) :: Vector<String>
    - *Serveix per demanar una recomanació d'ítems per a l'usuari passat com a paràmetre. Retorna un vector amb els identificadors d'ítem, ordenats decreixentment segons la puntuació estimada que l'usuari donaria a aquell ítem.*

- getValoracioRecomanacio(Recomanacio r, UsuariActiu ua, Vector<String> recomanacio, HashMap <String, UsuariActiu> Unknown) :: Double
    - *Retorna el càlcul del dcg corresponent a la recomanació que es passa com a paràmetre per tal de valorar numèricament la seva qualitat.*

Classe Valoració

Aquesta classe serveix per enregistrar les valoracions que els usuaris fan sobre determinats ítems. Entre els seus atributs trobem la instància de l'usuari que ha fet la valoració, la instància de l'ítem que s'ha valorat i un double per representar la puntuació que dóna a aquell ítem. S'identifica per l'usuari i el item. A més, compta amb un atribut booleà per diferenciar les valoracions "predites" pels algorismes de recomanació, de les valoracions que han fet persones reals. Finalment, té dos atributs de tipus double estàtics per tal que la classe pugui saber quina és la puntuació màxima i la mínima que tenen els ítems (de manera que els algorismes de recomanació tampoc puguin predir puntuacions fora d'aquest rang).

Pel que fa a les operacions públiques, bàsicament són getters i setters dels atributs esmentats en el paràgraf anterior.

Classe Atribut (+ subclasses)

La classe Atribut representa els diferents atributs que pot tenir un item. Un atribut té un contingut que pot ser de tres tipus diferents: un string que representa categories i noms entre d'altres, un nombre que representa dades numeriques i un booleà que representa el compliment de diferents condicions o propietats. És per això que hi ha les tres subclasses de atribut; Atribut Categòric, Atribut Numèric i Atribut Booleà.

Un atribut és d'un ítem i pertany a una columna, de manera que si tenim les columnes Any, NomPelicula, TéLimitacionsDeEdat, un atribut numèric podria pertànyer per exemple a la columna amb nom 'Any', un categòric podria pertànyer a NomPelicula i un booleà podria pertànyer a la columna 'TéLimitacionsDeEdat'.

El contingut dels atributs son conjunts de valors ja que es pot donar que per una mateixa columna un item tingui diversos valors d'atribut. El set ens permet emmagatzemar-los junts.

Pel que fa a les operacions de la superclasse, trobem:

- esClau() :: boolean
    - *Retorna vertader si aquell atribut és la clau o identificador de l'ítem al què pertany.*

- ComparaAtributs(Atribut a) :: double
    - *Retorna el grau de similitud entre l'Atribut sobre el que s'invoca la funció i l'Atribut que es passa com a paràmetre. Té una implementació diferent en cada subclasse, segons el seu tipus.*

Classe Columna

Una columna representa, com bé diu el nom, una columna a la 'matriu' d'items.csv, on una fila correspon als valors dels atributs d'un item donat i les columnes a les diferents categories d'atributs del que està compost l'ítem. S'identifica per un número.

Una columna té un nom, un número de columna (que va d'acord a la posició en la que es troba en la 'matriu' d'items.csv) i altres atributs necessàris com el valor màxim i mínim en cas de que la columna guardi atributs numèrics o un booleà que és cert si aquella columna representa la clau externa d'un ítem.

Les seves operacions son constructores, getters i setters.

Classe Item

Els objectes d'aquesta classe representen els ítems que els usuaris poden valorar i rebre a les recomanacions. Cada instància d'Item compta amb un identificador de tipus String, un vector d'atributs amb els Atributs que té aquell ítem, un set de Valoracions on es guarden les valoracions que s'han fet d'aquell ítem, i també un set de recomanacions que conté les Recomanacions on apareix aquell ítem.

Entre les operacions principals, a banda de getters i setters, podem trobar:

- ComparaItem(Item comparat) :: double
    - *Serveix per comprovar la distància/grau de similitud entre dos ítems.*

- ComparaItemEucl(Item comparat) :: double
    - *Serveix obtenir la distància euclidiana entre dos ítems.*

Classe Recomanació

Una recomanació representa el resultat de executar els algorismes de filtering. La recomanació retorna unes Valoracions predictives sobre uns items concrets per a un usuari concret (qui demana la recomanació a través de l'associació "demana" o bé l'usuari que l'administrador indica per a poder provar la qualitat dels algorismes a través de l'associació "valora"). Els seus atributs són una instància de la Strategy que s'ha usat per obtenir aquella recomanació, l'usuari a qui va adreçada i un set de Valoracions que correspon al resultat de la recomanació calculada per l'algorisme corresponent a l'estratègia escollida.

L'única operació que no és getter ni setter és:

- valoraRecomanacio(Vector<String> itemsRecomanats, UsuariActiu u, HashMap<String, UsuariActiu> UsuarisUnknown) :: double
    - *Aquesta operació calcula el dcg (més informació sobre aquesta mesura en el document d'algorismes usats en el sistema) de la llista d'ítems que conformen una recomanació, segons el grau d'encert en el seu ordenament.*

Classes Strategy

Interfícies que s'usen per poder aplicar el patró Estrategia. Aquest patró permet canviar en temps d'execució la estrategia que es fa servir. Aquestes interficies tenen unes operacions que obligatòriament cada classe que les usi ha d'implementar.

Concretament, la classe Strategy es la que s'usa per l'estratègia de filtering i té l'operació : Filtering (UsuariActiu ua, Map<String,Item> items) ::TreeSet<Valoració>

La classe StrategyDistancia s'usa per l'estratègia del calcul de distancies entre usuaris i té l'operació: distancia(Vector<Double> coordenades1, Vector<Double> coordenades2) ::Double.

Classe CollaborativeFiltering

Aquesta classe implementa una estratègia de filtering que combina els algorismes de k-means i Slope one, que apareixen detallats en el document d'algorismes i estructures de dades utilitzats. Els atributs d'aquesta classe consisteixen en un String amb el nom del filtering, una instància de StrategyDistancia (que serà la corresponent al tipus de càlcul de distància configurat) i un atribut "k" de tipus int per guardar el nombre de clústers usats en el k-means.

Pel que fa a les operacions principals, trobem:

- calculaParticio(UsuariActiu ua, Vector<Double>[] centroides, Map<String,Item> items) :: int
    - *Serveix per determinar a quin clúster ha d'anar un usuari ua. Retorna un enter entre 0 i k-1 corresponent al número de clúster.*

- Filtering (UsuariActiu ua, Map<String,Item> items) ::TreeSet<Valoració>
    - *Correspon a la implementació del mètode definit a la interfície Strategy. També conté la implementació de l'algorisme Slope One, que més endavant tindrà una funció pròpia. Retorna el conjunt de valoracions predites pels algorismes de filtering, ordenades de major a menor usant com a comparació les puntuacions estimades d'aquestes.*

- kmeans(HashMap<String, UsuariActiu> allUsers, Map<String, Item> allItems) :: Pair< Vector<UsuariActiu>[] , Vector<Double>[] >
    - *Implementa l'algorisme k-means per separar tots els usuaris en k clústers segons la proximitat de les seves valoracions. Retorna un parella d'arrays de vectors on el primer array correspon als diversos clústers (amb els seus respectius usuaris), i el segon array conté les coordenades del centroide de cada clúster.*

Classe ContentBasedFiltering

Aquesta classe implementa una estratègia de filtering usant l'algorisme k-nearest neighbours, també detallat al document d'algorismes utilitzats. Els atributs principals són un String amb el nom de l'estratègia de filtering, i un enter k que representa com a màxim quants ítems més semblants al que s'està comparant s'han de considerar.

Entre les operacions, trobem:

- Filtering (UsuariActiu ua, Map<String,Item> items) ::TreeSet<Valoració>
    - *Correspon a la implementació del mètode definit a la interfície Strategy. Retorna el conjunt de valoracions predites pels algorismes de filtering, ordenades de major a menor usant les puntuacions estimades per cada item.*

- KNearests(Item donat, Map<String,Item> tots, int k) :: PriorityQueue< Pair<Double,Item> >

    - *Implementa l'algorisme k-nearest neighbours explicat de forma més detallada en el document d'algorismes utilitzats. Retornar una cua ordenada de forma decreixent per similitud entre items dels com a molt k ítems més semblants a l'ítem donat.*

- calcularNota(double similitud, double notaItemSemblant) :: Double
    - *Serveix per fer una predicció/estimació de la puntuació d'un ítem per part d'un usuari en funció de la puntuació coneguda que ha fet l'usuari a un item i segons el seu grau de similitud del item que s'intenta predir amb la del item que la puntuació és coneguda.*

- ponderarItem(Pair<Integer, Double> p) :: Double
    - *Serveix per, tenint en compte el nombre de vegades que s'ha repetit un item (és a dir, el nombre de repeticions és el nombre de vegades que ha sortit que aquest item és similar a items que a l'usuari li agraden i per tant el nombre d'items agradats semblants a aquest) i la suma de les notes predites en funció de la nota al item semblant i la similitut a aquest, calcular la nota que es preveu que l'usuari donarà a aquest item, fent una mitjana de les diferents notes predites amb calcularNota i aplicant-li una bonificació en funció del nombre de repeticions.*

Classe HybridApproaches

Aquesta classe encara no està implementada en la primera entrega, però correspon a una tercera estratègia de filtering que combina Collaborative Filtering i Content-based Filtering per tal de fer les prediccions de les valoracions de certs ítems necessàries per construir les recomanacions.