PROP

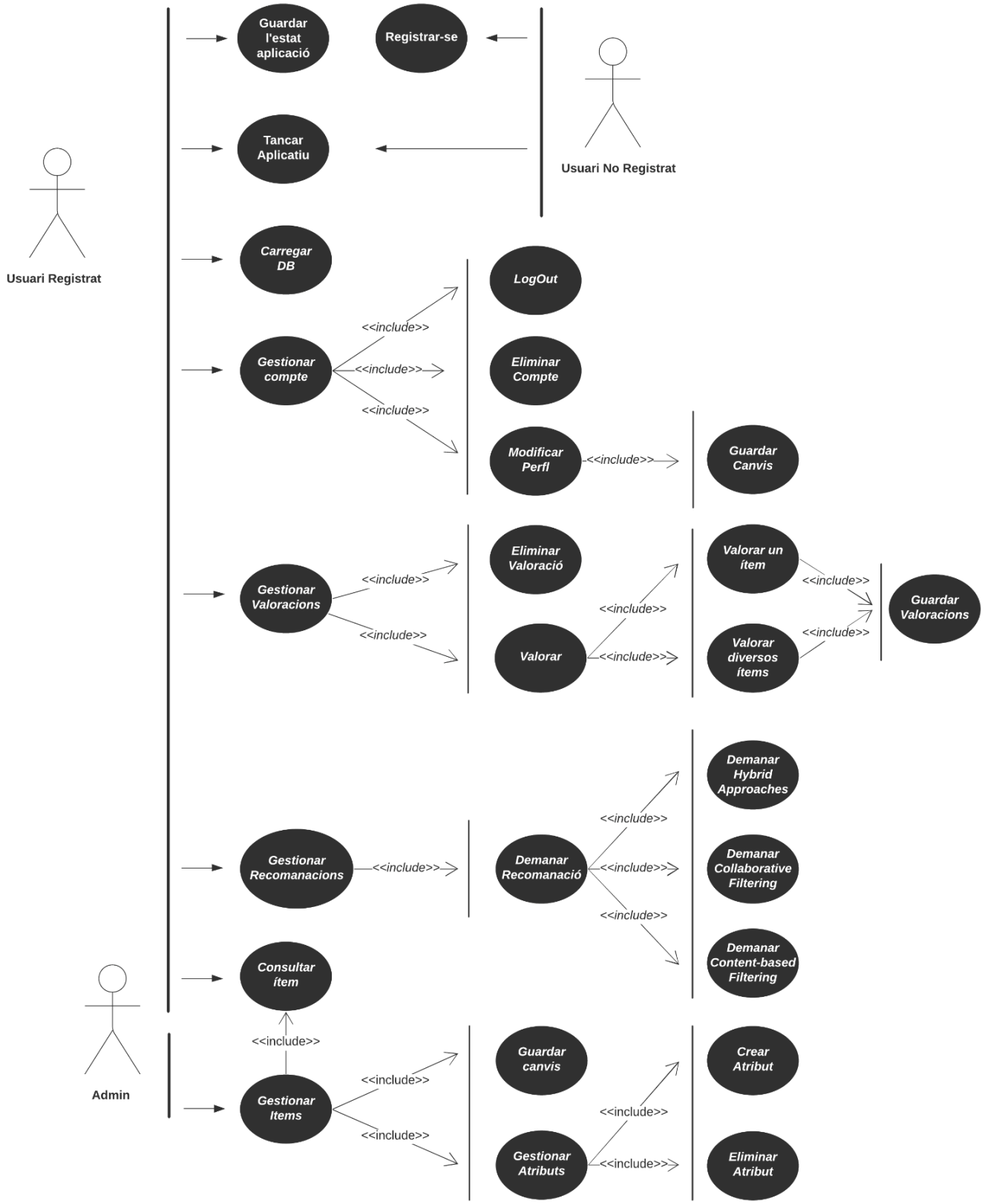
| Sistema Recomanador

PRIMERA ENTREGA

Abdelali El Attar Yalaoui
Miguel García Soler
Ricard Guixaró Trancho
Sergi Casau Pueyo

ÍNDEX

1. CASOS D'ÚS - Esquema



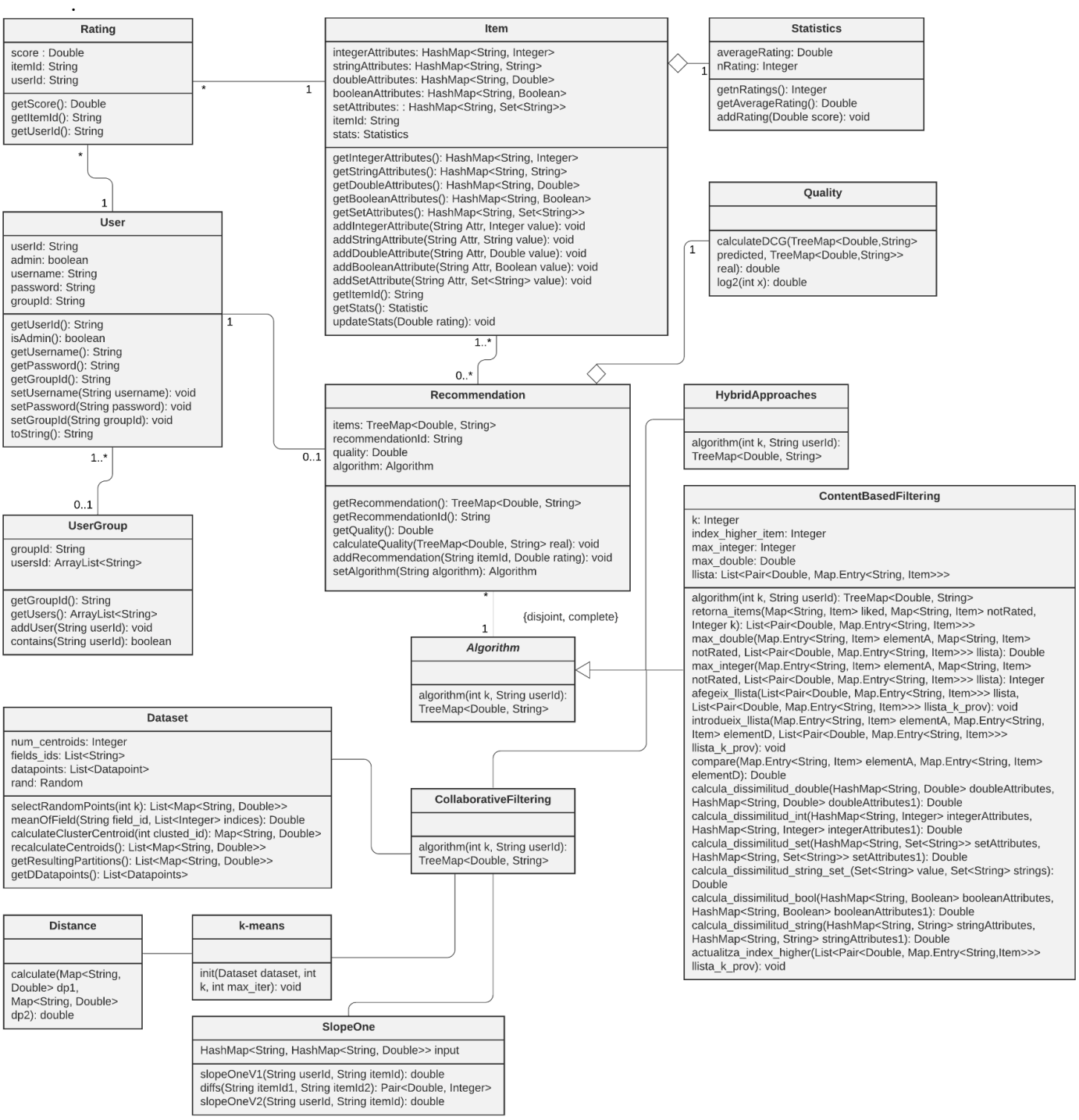
2. CASOS D'ÚS - Explicació

Cas d'ús		Actor	Descripció	Errors
Carregar DB		Usuari/Admin	L'administrador carrega la base de dades i el sistema defineix automàticament el tipus d'ítem.	Ja hi ha un tipus creat, el sistema informa de l'error.
Registrar-se		Usuari no registrat	L'usuari introdueix el username i el password. El sistema valida les dades i crea el nou perfil.	Passwords no coincideixen, el sistema informa de l'error.
Gestionar compte		Usuari/Admin	L'usuari accedeix a un menú amb diverses opcions.	
-	***LogOut***	Usuari/Admin	L'usuari selecciona l'opció de tancar la sessió. El sistema tanca la sessió i torna a l'inici.	El sistema informa l'usuari de l'opció de guardar les dades o sortir sense desar.
-	***Eliminar compte***	Usuari/Admin	L'usuari selecciona eliminar el compte i torna a l'estat inicial.	El sistema li torna a demanar a l'usuari si realment vol eliminar el seu compte.
-	***Modificar perfil***	Usuari/Admin	L'usuari decideix modificar el perfil (nom, correu, etc.).	
-	***Guardar canvis***	Usuari/Admin	L'usuari demana al sistema guardar els canvis.	Password massa simple, username ja en ús, etc. el sistema informa de l'error.
Gestionar valoracions		Usuari/Admin	L'usuari accedeix a un menú amb diverses opcions.	
-	***Valoració***	Usuari/Admin	L'usuari demana al sistema valorar un o diversos ítems.	
-	***Valorar un ítem***	Usuari/Admin	L'usuari avalua un únic ítem.	Ítem ja valorat.
-	***Valorar diversos ítems***	Usuari/Admin	L'usuari avalua un conjunt d'ítems.	Ítem ja valorat.
-	***Eliminar valoració***	Usuari/Admin	L'usuari pot eliminar una valoració que ell hagi fet.	

-	***Guardar valoracions***	Usuari/Admin	L'usuari demana al sistema guardar els canvis respecte a les valoracions. El sistema té en compte els canvis.	
Gestionar ítems		Admin	L'usuari accedeix a un menú amb diverses opcions.	
-	**Consultar ítem**	Usuari/Admin	L'usuari pot consultar (amb condicions) un ítem i veure les estadístiques associades.	No existeix cap ítem que compleixi les condicions especificades per l'usuari
-	***Gestionar atributs***	Admin	L'usuari accedeix a un menú amb diverses opcions.	
-	***Crear atribut***	Admin	L'usuari crea un nou atribut pels ítems (derivat o calculat de la resta).	Nom de l'atribut invàlid, ja existent.
-	***Eliminar atribut***	Admin	L'usuari elimina un atribut a tots els ítems.	L'atribut seleccionat és l'identificador, o és necessari per als algorismes o la identificació del tipus d'ítem
-	**Guardar canvis d'ítems**	Admin	L'usuari demana al sistema guardar els canvis respecte als ítems.	El valor d'un camp modificat és invàlid.
Gestionar recomanacions		Usuari/Admin	L'usuari accedeix a un menú amb diverses opcions.	
-	***Demanar recomanació***	Usuari/Admin	L'usuari demana la recomanació al sistema. El sistema ofereix l'opció a l'usuari d'escollir l'algorisme.	
-	***Demanar Hybrid approaches***	Usuari/Admin	L'usuari selecciona el sistema de recomanació Hybrid approaches	
-	***Demanar Collaborative filtering***	Usuari/Admin	L'usuari selecciona el sistema de recomanació Collaborative filtering	
-	***Demanar Content-based filtering***	Usuari/Admin	L'usuari selecciona el sistema de recomanació Content-based filtering.	

Guardar estat de l'aplicació	Usuari/Admin	L'usuari demana al sistema guardar tots els canvis efectuats a l'aplicatiu.	Algun canvi no és correcte.
Tancar Aplicatiu	Usuari/Admin	L'usuari selecciona l'opció de sortir del sistema. Se surt del sistema.	El sistema informa l'usuari de l'opció de guardar les dades o sortir sense desar.

3. DIAGRAMA CLASSES UML - Esquema



4. DIAGRAMA CLASSES UML - Explicació

- **Rating:**

Un *Rating* fa referència a un ítem *itemId* i a un usuari *userId*. Conté la valoració en forma de `Double` que ha fet un usuari a un determinat ítem.

- **User:**

Un *User* està identificat per *userId*. Conté informació de l'usuari (si és administrador del sistema o és un usuari corrent, el seu username, la seva password i el grup d'usuaris al qual pertany).

- **UserGroup:**

Un *UserGroup* està identificat per *groupId*. Conté els identificadors d'aquells usuaris que pertanyen al grup, permet consultar si un *User* pertany al grup o no.

- **Item:**

Aquesta classe conté la informació d'un ítem juntament amb tots els seus conjunts d'atributs (de tots els tipus; `String`, `Integer`, `Double`, `Boolean` i `Set<String>`). Un Item està identificat pel seu identificador, *itemId*. També té associada informació sobre les seves estadístiques.

- **Statistics:**

És una classe auxiliar d'*Item* que emmagatzema les estadístiques associades als ítems. Està identificada per l'*Item* i permet ser actualitzada cada vegada que s'afegeix un nou *Rating*.

- **Quality:**

És una classe auxiliar de *Recommendation* que calcula la qualitat d'una recomanació. Està identificada per *Recommendation*. La funció principal calcula el *Discounted Cumulative Gain* de la *Recommendation* calculada en funció d'una *Recommendation* real.

- **Recommendation:**

Una *Recommendation* està identificada pel seu identificador, *recommendationId*. Aquesta classe emmagatzema un conjunt ordenat d'itemId's

i preferències. També guarda la qualitat d'una *Recommendation*, que es calcula mitjançant la funció: `private void public void calculateQuality(TreeMap<Double, String> realRecommendation)`. A més a més, crea la classe *Algorithm* en funció d'un atribut.

- **Algorithm:**

Classe abstracta on es crea una *Recommendation*.

- **Collaborative Filtering:**

Classe on es crea una *Recommendation* seguint l'estratègia Collaborative *Filtering*.

- **K-Means:**

Classe que classifica l'usuari actiu en un grup d'usuaris semblants a ell.

- **Dataset:**

Classe que abstrau el concepte d'unitat d'informació i conté una sèrie de mètodes auxiliars per facilitar el càlcul del KMeans.

- **SlopeOne:**

Classe que prediu la valoració que faria l'usuari actiu a un determinat *Item*, en funció dels *Ratings* que han fet usuaris del seu *UserGroup* a aquell *Item*.

- **Content-based Filtering:**

Classe on es crea una *Recommendation* seguint l'estratègia Content-based Filtering. Implementa un algorisme *k*-nearest neighbours (*k*-NN).

- **Hybrid Algorithm:**

Classe on es crea una *Recommendation* seguint una estratègia que combina tècniques basades en les dues aproximacions anteriors.

- **Distance:**

Classe que calcula la distància entre dos usuaris basant-se en els *Rating*'s de tots els *Item*'s que han valorat en comú.

5. DESCRIPCIÓ FUNCIONALITATS PRINCIPALS

5.1. Collaborative Filtering

L'estratègia del Collaborative Filtering es basa a recomanar a l'usuari aquells ítems que han agradat a usuaris semblants. Està implementada en dues parts:

a. K-Means:

L'algoritme necessita els següents paràmetres:
- El número de centroides inicial.
- El número màxim d'iteracions que pot realitzar l'algoritme.
- El *dataset* sobre el qual aplicar l'algoritme.

Primerament, se seleccionen els centroides inicials escollint aleatòriament K usuaris, això garanteix que inicialment no hi hagi cap partició buida. Aquesta part de l'algoritme té un cost O(k).

A partir d'aquí se segueixen els mateixos passos fins que l'algoritme finalitza (arribem al màxim d'iteracions o les particions s'estabilitzen). Per cada *datapoint* (usuari) es calcula de quin dels K centroides es troba més a prop fins que tots es troben assignats a un clúster. Això té un cost O(K * n * m), on *n* és el número de datapoints i *m* el número d'atributs (ítems valorats) per usuari.

A continuació es recalculen els centroides fent la mitjana de tots els *datapoints* de cada clúster i repetim fins a acabar l'algoritme. Aquest últim pas té un cost proporcional a O(n * m).

b. SlopeOne:

La funció rep diversos paràmetres:
- L'identificador de l'usuari que vol rebre una recomanació.
- L'identificador de l'ítem pel qual es vol fer la predicció.

A més a més, també s'accedeix al conjunt de valoracions a través del Controlador de Domini.

El primer pas és recórrer aquells ítems que ha valorat l'usuari actiu i, si no són l'ítem pel qual s'està fent la predicció, calculem la desviació mitjana i la quantitat

de valoracions. Per fer-ho, s'accedeix a les valoracions d'aquells usuaris que pertanyin al grup d'usuaris de l'usuari actiu i que hagin valorat tots dos ítems.

Per cada desviació, aquesta es divideix entre el nombre de valoracions, se suma a la valoració que ha fet l'usuari actiu a aquest segon ítem i es multiplica per n.

Finalment, les desviacions acumulades es divideixen entre les ns acumulades, obtenint així la predicció que faria l'usuari actiu a un determinat ítem.

5.2. Content Based Filtering

L'estratègia implementada està basada en l'algorisme "*k*-nearest neighbours (*k*-NN)". Bàsicament, estudia quins són els ítems més semblants als que el nostre *User* actiu ha valorat positivament. Després d'haver recorregut tot el conjunt d'ítems, només ens quedarem amb els *k* ítems més similars.

Per tant, el primer que fem és, treballar amb dos *HashMap*s, un que representa el conjunt d'ítems que li agraden a l'usuari actiu i l'altre que representa el conjunt d'ítems no valorats per l'usuari.
La funció, per tant, es basa en fer un doble bucle, on comparem cada ítem que li agrada a l'usuari amb cada ítem de la base de dades no valorat. En aquesta comparació ens quedem amb tots els ítems i la seva puntuació, la qual indica més probabilitat de ser recomanat com més propera a 0. Les següents iteracions fem el mateix, però de manera que actualitzem les puntuacions dels ítems. Aquesta puntuació es calcula a partir de la no similitud entre els atributs de l'ítem i de la valoració mitjana complementaria. Pondera per ser concrets un 60% la no similitud i un 40% la valoració mitjana complementària. Les següents iteracions fem el mateix, però de manera que actualitzem les puntuacions dels ítems.

Tots els ítems que hem introduït a la llista ara tenen una puntuació associada. I, per tant, finalment, l'únic que hem de fer és quedar-nos amb els k elements de la llista amb menys puntuació. Que seran els més similars i amb valoració mitjana més elevada.

5.3. Hybrid Approaches

L'estratègia del Hybrid Approaches és simple, recomanem els millors Ítems segons l'algorisme Collaborative i els millors segons el Content Based. Després fem la unió d'aquestes dues recomanacions en funció de la rellevància dels ítems recomanats.

5.4. Avaluació Recomanacions

La funció rep dos paràmetres:

- Una llista amb la recomanació calculada.
- Una llista amb la recomanació real.

L'algorisme recorre els ítems de la recomanació calculada, i per a cada un d'ells busca la posició que aquest ocupa en la recomanació real.

Si aquest ítem pertany a la recomanació real, s'obté la valoració associada, altrament, la rellevància pren com a valor 0.

A partir d'aquí, s'aplica la fórmula del Discounted Cumulative Gain tenint en compte que, si dos ítems tenen la mateixa valoració, tots dos ocupen la mateixa posició dintre de la recomanació, i el següent ítem de la llista també estarà en una posició inferior (més rellevància).

El nostre sistema no calcula el Normalized Discounted Cumulative Gain.

6. DESCRIPCIÓ ESTRUCTURES DE DADES

Estructures principals utilitzades:

- Recomanació:

L'estructura de dades escollida per emmagatzemar una **recomanació** és un TreeMap en ordre invers (decreixent) on la '*key*' és el valor de la valoració i el '*value*' és l'identificador de l'ítem.

```
TreeMap<Double, String> recommendation;
```

Aquesta implementació ens evita haver d'ordenar el conjunt d'ítems recomanats constantment. Això fa que en termes d'eficiència, el `TreeMap` es quedi enrere en comparació amb `HashMap`'s i `LinkedHashMap`'s, però com tampoc haurà d'emmagatzemar quantitats immenses de dades, la pèrdua en eficiència compensa l'ordenació automàtica.

- Conjunt de valoracions:

Per tal de guardar tots els ratings de forma que aquests després siguin fàcilment accessibles, s'ha decidit implementar el conjunt de la següent manera:

```
HashMap<String, HashMap<String, Double>> ratings;
```

On cada usuari té un *`HashMap<String, Double>`* on la '*key*' és l'identificador de l'ítem, i el '*valor*' la respectiva valoració en format *double*. Això ens permet buscar fàcilment els ratings associats a un usuari, condició capital per a poder executar els algorismes de recomanació eficientment.

- ContentBased:

També hem utilitzat una llista de parells per emmagatzemar les recomanacions amb el nivell de similitud:

```
List<Pair<Double,Map.Entry<String, Item>>> llista;
```

Hem ordenat la llista de forma ascendent segons el Double, obtenint així els elements més similars al principi de la llista. Cada Double té com a parella l'Item a recomanar i el seu identificador.