ALGORISMES I ESTRUCTURES DE DADES EMPRATS

Driver Algorisme

Aquesta classe fa servir l'estructura de dades *ArrayList* per realitzar la càrrega de conjunts de dades. També crea una llista de la classe *ItemValoracióEstimada*, que és una classe que hem creat que emmagatzema un *Item* i una valoració, i re-implementa la funció de comparació per així permetre l'ordenació segons l'atribut valoració. Aquesta classe l'hem feta servir per ordenar les valoracions dels usuaris a *Unknown* i per emmagatzemar els resultats dels algorismes recomanadors.

Els algorismes emprats directament són el càlcul de DCG, IDCG i NDCG descrits a la documentació (que fan servir el càlcul de potències i logaritmes de la llibrería *Math*) i l'algorisme d'ordenació de llistes que ens dona la llibreria *Collections*, que hem usat per ordenar les valoracions estimades d'*Unknown* com prèviament descrit.

ItemValoracióEstimada

Aquesta classe senzilla no empra cap estructura de dades ni algorisme, excepte una funció que donat un ItemValoracióEstimada retorna el major entre l'instància actual i el parametre.

Controlador Domini Algorisme

Aquesta classe fa servir l'estructura de dades *ArrayList* per emmagatzemar els resultats dels algorismes recomanadors. No implementa cap algorisme donat que aquest el realitzen les respectives classes.

CollaborativeFiltering

Aquesta classe empra les estructures de dades *ConjuntItems*, *ConjuntUsuaris* i *ConjuntRecomanacons*, que són extensions de la classe *ArrayList*. També fa servir un array de la classe privada *Centroide,* que cadascuna emmagatzema tres *HashMaps*: valoració, sum i quant. Aquests mapes emmagatzemen una variable per a cada item, i ens serveixen per calcular les distàncies i recalcular les mitjanes. També fem servir un *HashMap* en l'atribut *ClosestCentroid*, que ens guarda per a cada usuari quin és el seu centroide més proper. També emprem la classe *ItemValoracióEstimada*, que ens permet ordenar les recomanacions segons la seva valoració estimada.

D'algorismes hem usat els algorismes descrits a la documentació per organitzar els usuaris en clústers d'usuaris semblants (*k-Means*) i per posteriorment obtenir valoracions estimades (*Slope-1*)

ContentBasedFiltering
Per a aquesta classe hem emprat l'estructura de dades *ArrayList* per a emmagatzemar un conjunt d'*ItemValoracioEstimada* que ens permet ordenar un conjunt d'ítems segons la seva valoració estimada.

Per a la part d'algorisme, hem implementat l'algorisme proposat a la documentació de k-NearestNeighbours. L'hem implementat amb aquestes dues consideracions que hem vist adients:

La similitud entre dos ítems (aquesta funció es troba a conjunt d'ítems, però hem vist adient descriure-la aquí) l'hem calculat sumant la similitud dels seus atributs multiplicada per la seva ponderació. La similitud dels atributs es calcula així:

- Si són valors numèrics, la diferència dels valors dividida entre la diferència del valor màxim i mínim de tots els ítems.
- Si són dates es tracta igual que amb números però passant les dates a dies.
- Si són valors booleans 1 si són iguals, 0 altrament.
- Si són strings la distància d'editat (calculada amb programació dinàmica) dividida entre la mida màxima de les dues strings.
- Si es un conjunt de strings (és a dir, tags) és la mida de la intersecció dividit entre la mida de la unió.

Per decidir com ordenar les valoracions i obtenir-ne les Q millors, la documentació proposa ordenar-les per nombre d'aparicions, tot i que ho deixa obert a discussió. Nosaltres hem decidit que ens semblava una millor alternativa multiplicar la similitud per la valoració de l'ítem semblant i agafar aquest valor més alt com a valoració estimada de l'ítem. Això valora més els ítems semblants a ítems molt ben valorats, i ens retorna una valoració estimada el qual facilitarà la implementació d'estratègies híbrides. Si durant el transcurs del treball veiem que un altra alternativa és millor la implementarem.

Finalment, l'algorisme demana definir un "threshhold" a partir dels quals es consideren les valoracions de l'usuari, per a només agafar les positives. Nosaltres hem considerat adequat agafar el valor 3.5, però hem plantejat altres alternatives com trobar la valoració en la posició ⌊mida*0.75⌋, que es podria calcular eficientment usant l'algorisme de selecció.

Controlador Persistencia
Aquesta classe usa com a estructura de dades la classe *ArrayList*. Aquesta és usada per a passar i rebre les dades cap a capes superiors, normalment en forma de Matriu bidimensional de Strings, tot i que en algun cas s'opera amb matrius unidimensionals i tridimensionals. Per a la implementació d'algunes funcions, es considera la primera fila de l'array bidimensional la capçalera de la taula.
L'atribut de la classe *estat* emmagatzema 6 atributs que usen classes del domini, i manté la informació fins a l'hora de guardar-la usant una *ArrayList*.

També usa una altra estructura proporcionada per java: la classe *File*. Aquesta permet obtenir informació sobre els arxius i directoris de l'ordinador, crear-los i eliminar-los.
Respecte als algorismes usats, aquesta classe no n'usa cap llevat de recorreguts sencers de taules.

Controlador Load
Aquesta classe usa com a estructura de dades la classe *ArrayList*, la classe *File* i la classe *FileReader*. *ArrayList* és usat per a mantenir en una taula bidimensional les dades que es van llegint en forma de strings. La classe *File* és usada com a input de la funció per saber de quin arxiu s'ha de llegir les dades. La classe *FileReader* instància un nou objecte d'aquest tipus a partir de la *File*, i ens permet fer operacions de lectura de l'arxiu obert.
Aquesta classe no usa cap algorisme llevat del recorregut de l'arxiu que es llegeix. Se suposa un arxiu en format .csv i es fa un recorregut de taula.

Controlador Save
Aquesta classe usa com a estructura de dades la classe *ArrayList*, la classe *File* i la classe *FileWriter*. *ArrayList* és usat per a mantenir en una taula bidimensional les dades que es passen per paràmetre a la funció principal de guardat. La classe *File* és usada com a input de la funció per saber a quin arxiu s'ha d'escriure les dades. La classe *FileWriter* instància un nou objecte d'aquest tipus a partir de la *File*, i ens permet fer operacions d'escriptura de l'arxiu obert.
Aquesta classe no usa cap algorisme llevat del recorregut de la taula. Es guarden els valors de la taula en format .csv.

Usuari
Usuari empra l'estructura de dades *ConjuntRecomanacions* que es tracta d'una extensió d'*ArrayList*. No empra cap algorisme.

Conjunt Usuaris
Aquesta classe es tracta d'una extensió de l'estructura de dades *ArrayList*. Addicionalment, donat que el conjunt d'usuaris es troba ordenat per ID hem implementat l'algorisme *BinarySearch*, que ens permet cercar un usuari en cost asimptòtic logarítmic en comptes del cost lineal que té la cerca per defecte de *l'ArrayList*.

Recomanació
La classe recomanació posseeix un Usuari, un Item i un float opcional Valoració. No empra cap altra estructura de dades ni algorisme.

Conjunt Recomanacions

Aquesta classe es tracta d'una extensió de l'estructura de dades *ArrayList*, ordenat primer per ID ítem i després per ID d'usuari. Ho hem decidit així, ja que ja podem accedir a les valoracions d'un usuari fent usuari.getValoracions().

Addicionalment, donat que el conjunt d'usuaris es troba ordenat, hem implementat l'algorisme *BinarySearch*, que ens permet cercar una recomanació en cost asimptòtic logarítmic en comptes del cost lineal que té la cerca per defecte de l'ArrayList.

Ítem

La classe Ítem disposa de 3 *ArrayList* estàtiques, que representen les capçaleres dels atributs, els tipus dels atributs i els pesos dels atributs. També disposa d'una *ArrayList* privada amb els valors dels atributs. Addicionalment fa servir l'estructura de dades *Enum* que representa els tipus que pot tenir un atribut.

Conjunt d'Ítems

La classe *ConjuntItems* es tracta d'una extensió de la classe *ArrayList.* Donat que el conjunt d'ítems es troba ordenat per ID hem implementat l'algorisme *BinarySearch*, que ens permet cercar un ítem en cost asimptòtic logarítmic en comptes del cost lineal que té la cerca per defecte de *l'ArrayList*.

Aquesta classe també disposa d'un algorisme per calcular la similitud entre dos Ítems, que invoca una funció per calcular la similitud entre dos atributs. Aquests algorismes estan descrits a *ContentBasedFiltering*.