DESCRIPCIÓ DE CLASSES

PETITA INTRODUCCIÓ AL DOCUMENT.

En aquest document descrivim les classes que es troben en les diferents capes de la nostra arquitectura.

Per les classes de la capa de presentació i la capa de persistencia no calia una explicació detallada, si més no, l'hem feta per la capa de persistencia ja que hem pensat que facilitaria entendre altres classes i operacions que es troben a la capa de Domini.

Per cada classe s'indiquen els atributs d'aquesta, s'explica breument per a que son utilitzats i es redacta el funcionament general dels mètodes (obviant getters i setters) que queden implementats en la classe.

És possible que, per a l'última entrega, hagin hagut alguns canvis com per exemple que aparegui una classe a la capa de dades encargada de la gestió de recomanacions, per tal de guardar-les i llegir-les. També podrien aparèixer operacions necessàries per tema de front-end o per acabar de retocar els algorismes i funcionalitats del sistema. A més a més, si ens dona temps, intentarem dividir la classe CtrlDomini en diversos controladors de manera que seria molt més fàcil de llegir i entendre quines operacions s'hi troben dins.

CAPA DADES

CtrlItemsFile

És una classe que implementa interfície CtrlItems de la capa de domini. Hem fet una interfície a la capa de Domini per tal de disminuir l'acoblament entre capes. La classe CtrlItemFile s'encarrega de la lectura i escriptura de fitxers d'ítems. Aquesta classe conté 3 operacions:

Operacions:

public List<String> getAll(String filename):

- Aquesta operació es crida des de el CtrlDomini. Rep com a paràmetre un nom de fitxer csv situat al subdirectori DATA sense l'extensió. Aquest fitxer ha de contenir la informació dels ítems que necessitem per al programa. L'operació retorna tot el contingut del fitxer en una llista de Strings. CtrlDomini s'encarrega de com interpretar la informació per carregar en memòria totes les instàncies necessàries (ítems, atributs, etc.).

public void saveItems(Map<String, Pair<Vector<String>, Vector<Vector<String> > > > items):

- Aquesta operació es crida des de CtrlDomini. Es rep com a paràmetre un Map, on la clau és l'id de l'ítem, el valor és un Pair on el primer element es un vector amb els noms de cada columna i el segon element és un vector de vectors de strings que representen, per cada columna, els possibles atributs que té l'ítem identificat per la clau del map. L'operació s'encarrega d'emmagatzemar en el fitxer itemsDB amb extensió JSON situat al subdirectori DATA.

public Map<String, Pair<Vector<String>, Vector<Vector<String> > > > readItems()

- Aquesta operació es crida des de CtrlDomini. Llegeix el fitxer itemsDB.json del subdirectori DATA i retorna les dades del fitxer en un map. No rep res com a paràmetre. Retorna un Map on la clau és l'identificador de cada ítem, el valor és un Pair on el primer element és un vector amb els noms de cada columna i el segon element és un vector de vectors que representen, per a cada columna, els diferents valors de cada atribut que té l'ítem identificat per la clau del map.

CtrlRatingsFile

És una classe que implementa interfície CtrlRating de la capa de domini. Hem fet una interfície a la capa de Domini per tal de disminuir l'acoblament entre capes. La classe CtrlRatingsFile s'encarrega de la lectura i escriptura de fitxers de ratings. Aquesta classe conté 3 operacions:

Operacions:

public List<String> getAll(String filename):

- Aquesta operació es crida des de el CtrlDomini. Rep com a paràmetre un nom de fitxer csv situat al subdirectori DATA sense l'extensió. Aquest fitxer ha de contenir la informació dels ratings (es pot entrar ratings.db, ratings.test.known o ratings.test.unknown) que necessitem per al programa. L'operació retorna tot el contingut del fitxer en una llista de Strings. CtrlDomini s'encarrega de com interpretar

la informació per carregar en memòria totes les instàncies necessàries (Usuaris, Valoracions, etc.).

public void saveRatings(Vector<Integer> usrs, Vector<Integer> nVals, Vector<String> items, Vector<Double> puntuacio):

- Aquesta operació es crida des de CtrlDomini. Es rep com a paràmetre un Vector d'enters que representen els ids dels usuaris, un vector d'enters que representa el número d'ítems que l'usuari iessim del vector usrs ha valorat, un vector de Strings que són identificadors d'ítems i un vector de Doubles que conté la puntuació de cada ítem situat a la mateixa posició. L'operació s'encarrega d'emmagatzemar en el fitxer usuarisRatings amb extensió JSON situat al subdirectori DATA.

  Exemple:
  usrs = {u1, u2}, nVals = {2, 1}, items = {item1, item2, item1}, puntuació = {5, 3.5, 4}
  En aquest cas podem observar que l'usuari u1 ha valorat 2 ítems que són els dos primers ítems del vector ítems (item1 i item2) i els ha puntuat amb un 5 i 3.5 respectivament.

public Map<Integer, Vector< Pair <String, Double> > > readRatings():

- Aquesta operació es crida des de el CtrlDomini. No rep cap variable per paràmetre. La funció s'encarrega de llegir els Ratings del fitxer usuarisRatings amb extensió JSON situat al subdirectori DATA. L'operació s'encarrega d'afegir el contingut del fitxer en un Map on la clau és un enter que identifica un id d'usuari, i el valor és un vector de Pairs on el primer element del vector és un String que és l'identificador d'un ítem i el segon element és un Double que representa la puntuació que l'usuari identificat amb la clau del map ha atorgat a l'ítem. Un cop s'hi ha afegit tota la informació del fitxer usuarisRatings al Map es retorna la informació cap al CtrlDomini que és l'encarregat de crear les diferents instàncies (Usuaris, Valoracions, etc.).

CtrlUsuarisActiusFile

És una classe que implementa interfície CtrlUsuariActiu de la capa de domini. Hem fet una interfície a la capa de Domini per tal de disminuir l'acoblament entre capes. La classe CtrlUsuarisActiusFile s'encarrega de la lectura i escriptura de fitxers dels usuaris registrats al sistema. Aquesta classe conté 5 operacions:

Operacions:
public Pair<Integer, Vector<Pair<String, Double> > >getUsuariActiu(String usrname, String password)

- Aquesta operació es crida des de el CtrlDomini i llegeix del fitxer usuarisActius amb extensió JSON (on guardem les dades dels usuaris registrats al nostre sistema) situat al subdirectori DATA. Es reben com a paràmetre dos Strings; el primer és un nom d'usuari i el segon és la contrasenya de l'usuari. Aquesta funció comprova que si existeix un Usuari amb el Username usrname i que l'usuari amb Username usrname tingui la contrasenya correcta. Si la informació rebuda com a paràmetre és correcte, aleshores es guarda en un Pair la informació de l'usuari. El primer element del pair és un enter que representa l'userId de l'usuari rebut per paràmetre, el segon element és un vector de parelles String i Double que representen les valoracions

fetes per l'usuari, on el String fa referència a un id d'un item i el Double la puntuació que ha fet l'usuari per a aquell ítem.

public void saveUser(int userid, String username, String contrasenya, Vector<Pair<String, Double> >ValoracionsUsuari)

- Aquesta operació es crida des de el CtrlDomini i emmagatzema l'usuari rebut per paràmetre al fitxer usuarisActius amb extensió JSON situat al subdirectori DATA. Es rep com a paràmetre un enter que representa un userId de l'usuari que volem guardar, un String que representa l'username i un altre String que representen el nom d'usuari i la contrasenya de l'usuari que volem guardar. També rebem per paràmetre un vector de parelles String, Double que són les valoracions que ha fet l'usuari. El String representa un id d'ítem i el Double representa la puntuació que l'usuari que volem guardar ha atorgat a l'ítem. La funció s'encarrega de guardar les dades d'aquest usuari al fitxer usuarisActius.json situat al subdirectori DATA.

public Boolean existeixUsuari(String username)

- Aquesta operació retorna cert si existeix un usuari al fitxer usuarisActius.json del subdirectori DATA que tingui Username = username i fals en cas contrari.

public Boolean existeixUsuari(int userId)

- Aquesta operació retorna cert si existeix un usuari al fitxer usuarisActius.json del subdirectori DATA que tingui userId = userId i fals en cas contrari.

public void eliminarUsuari(int userId)

- Aquest mètode elimina de la base de dades d'usuaris Actius (és a dir, del fitxer usuarisActius.json) l'usuari identificat per l'identificador passat per paràmetre, per tant, s'esborra del fitxer tota la informació lligada a aquest usuari.

CtrlEstatFile

És una classe que implementa interfície CtrlEstat de la capa de domini. Hem fet una interfície a la capa de Domini per tal de disminuir l'acoblament entre capes. La classe CtrlEstatFile s'encarrega de la lectura i escriptura de fitxers dels usuaris registrats al sistema.

Operacions:

public void saveState(int kkmeans, int knearests, String estrategiaDistanica, String estrategiaFiltering, double puntmin, double puntmax, int ultimUserId, int ultimItemId)

- Aquesta operació s'encarregar de guardar l'estat que l'administrador ha configurat en l'última execució del programa. Aquestes dades es guarden al fitxer configRecomanador.json situat al subdirectori DATA. Els paràmetres que es reben fan referència a la k per al kmeans (kkmeans), la k per al knearests (kknearests), el nom de l'estratègia per al càlcul de distàncies utilitzada (estrategiaDistancia), el nom de l'estratègia de filtering que es fa servir per a recomanar ítems (estrategiaFiltering) la puntuació mínima que pot tenir una valoració en el sistema (puntmin), la puntuació màxima que pot tenir una valoració en el sistema (puntmax) i, per últim, els últims ids d'usuari i ítem que s'ha assignat en el sistema (ultimUserId i ultimItemId).

public Pair<Integer, Integer> getKsBD()

- Aquesta operació llegeix del fitxer configRecomanador del subdirectori DATA la informació de la k usada per kmeans i es guarda en el primer element del pair i la k usada per al knearests que es guarda en el segon element del pair. Es retorna la parella de ks.

public Pair<String, String> getEstrategiesBD()

- Aquesta operació llegeix del fitxer configRecomanador del subdirectori Data els noms dels algorismes tant del calcul de distancies com el nom de l'estratègia de càlcul de recomanació guardat al fitxer. L'operació retorna en el primer element del pair el nom de la estratègia de distància i en el segon element el nom de l'estratègia de filtering.

public Pair<Double, Double> getIntervalPuntuacionsBD()

- Aquesta operació llegeix del fitxer configRecomandor del subdirectori DATA els valors mínims i màxims que pot tenir una valoració. Es retorna, en el primer element la puntuació mínima possible que pot tenir una valoració, i en el segon element la puntuació màxima possible que pot tenir una valoració.

public int getUltimID()

- Aquesta operació llegeix del fitxer configRecomandor del subdirectori DATA el valor de l'últim userId que s'ha assignat. Es retorna aquesta informació.

public int getUltimIDItem()

- Aquesta operació llegeix del fitxer configRecomanador del subdirectori DATA el valor de l'últim itemId que s'ha assignat. Es retorna aquesta informació.

CtrlClustersFile

És una classe que implementa interfície CtrlClusters de la capa de domini. Hem fet una interfície a la capa de Domini per tal de disminuir l'acoblament entre capes. La classe CtrlClustersFile s'encarrega de la lectura i escriptura de fitxers i en fitxers dels clústers i centroides calculats al sistema. Permet no calcular-los a cada execució, evitant així un procediment molt costós. D'aquesta manera, es llegeix el fitxer json corresponent als clústers i el fitxer json corresponent als centroides, i s'evita recalcular-los. Per tant, només es recalcularan quan l'administrador seleccioni l'opció.

Operacions:

public void saveClusters(Vector<Vector<Integer>> clusters)

- Aquesta funció guarda els clústers passats per paràmetre al fitxer json corresponent, anomenat clusters.json, fitxer situat al subdirectori DATA.

public void saveCentroids(Hashmap <String, Double> [] centroids)

- Aquesta funció s'encarrega de guardar els centroides passats per paràmetre al fitxer json corresponent, anomenat centroides.json, fitxer situat al subdirectori DATA.

public Vector<Vector<Integer>> readClusters()

- Aquest mètode s'encarrega de llegir el fitxer clusters.json i passar a controlador domini la informació llegida a través d'un Vector<Vector<integer>>, on integer és l'identificador d'un usuari i el vector de vectors representa que hi ha varis clústers on cada clúster conté varis identificadors d'usuaris.

public Hashmap <String, Double> [] readCentroids()

- Aquest mètode s'encarrega de llegir el fitxer centroides.json i passar a controlador domini la informació llegida a través d'un Hashmap<String,Double>, on String és l'identificador d'un ítem i Double representa la puntuació que el centroide ha donat a aquell ítem.

CAPA DOMINI

Aquesta capa està separada en tres packages. El package de Model amb totes les classes que representen el model del sistema, el package de ControladorDomini on està situat el controlador de domini i el package DataInterface que conté una factoria de les interfícies de dades i del controlador de domini i les interfícies utilitzades per la capa de dades per tal de disminuir l'acoblament entre capes.

Package ControladorsDomini

Classe Ctrl Domini

Aquesta classe s'encarrega de realitzar operacions, rebre i respondre crides de la capa de presentació, fer crides a la capa de dades i crear les instàncies necessàries del model del sistema.

Atributs:

1. private CtrlItem ctrlItemFile
   1.1. Aquest atribut és una instància de la interfície CtrlItem que es fa servir per a la lectura i escriptura amb els diferents fitxers d'ítems.
2. private CtrlRating ctrlRatingsFile
   2.1. Aquest atribut és una instància de la interfície CtrlRating que es fa servir per a l'escriptura i lectura amb els diferents fitxers de ratings.
3. private CtrlUsuariActiu ctrlUsuarisActius
   3.1. Aquest atribut és una instància de la interfície CtrlUsuariActiu que es fa servir per a l'escriptura i lectura del fitxer usuarisActius.json
4. private CtrlEstat estatCtrl
   4.1. Aquest atribut és una instància de la interfície CtrlEstat que es fa servir per a l'escriptura i lectura del fitxer configRecomanador.json
5. private CtrlClusters clusterCtrl
   5.1. Aquest atribut és una instància de la interfície CtrlClusters que es fa servir per a l'escriptura i lectura dels fitxers centroides.json i clusters.json
6. private Persona PersonaActual
   6.1. Aquest atribut és una instància de la classe Persona. La PersonaActual fa referència a l'usuari que ha iniciat sessió i està usant el nostre sistema. La PersonaActual pot ser del tipus Admin o del tipus UsuariActiu.
7. private int col_id
   7.1. L'enter representa el número de columna on està situat l'identificador dels ítems.
8. private int n_cols
   8.1. L'enter representa el número de columnes totals que hi ha a les dades d'ítems.
9. private String estrategia
   9.1. Aquest atribut representa el nom de l'estratègia que fem servir per al càlcul de les recomanacions del nostre sistema. El string pot tenir tres possibles valors ("CollaborativeFiltering" o "ContentBasedFiltering" o "HybridAproaches")
10. private String estrategiaDistancia

10.1. L’atribut estrategiaDistancia representa el nom de l'estratègia que fem servir per al càlcul de les distàncies. El string pot tenir tres possibles valors (“DistanciaMitjana”, “DistanciaEuclidiana” o “DistanciaMitjana”).

11. private int kkmeans;
    11.1. Aquest atribut representa el número de clústers que es fan servir per a l’algorisme CollaborativeFiltering
12. private int kknearest;
    12.1. Aquest atribut representa el número d'ítems més propers que volem fer servir per a l'algorisme ContentBasedFiltering
13. private HashMap<String, Item> Items
    13.1. El HashMap conté totes les instàncies dels ítems que es fan servir per al sistema. Fem servir un map ja que a l’hora d’accedir als ítems podrem accedir a ells a través del seu id.
14. private Vector<Columna> Columnes
    14.1. Aquest Vector de columnes representa cada columna dels fitxers d'ítems. Ordenades de la mateixa manera que les columnes del fitxer d'ítems.
15. private HashMap<Integer, UsuariFitxers> UsuarisRatings
    15.1. Aquest Map conté als usuaris dels fitxers ratings.db o usuarisRatings.json. La clau del map representa l’identificador de cada usuari situat a un dels dos fitxers i el valor del map són les instàncies ja creades dels usuaris amb les seves valoracions ja associades.
16. private HashMap<Integer, UsuariFitxers> UsuarisKnown
    16.1. Aquest Map conté als usuaris del fitxer ratings.test.known. La clau del map representa l’identificador de cada usuari situat al fitxer i el valor del map són les instàncies ja creades dels usuaris amb les seves valoracions ja associades.
17. private HashMap<Integer, UsuariFitxers> UsuarisUnknown
    17.1. Aquest Map conté als usuaris del fitxer ratings.test.unknown. La clau del map representa l’identificador de cada usuari situat al fitxer i el valor del map són les instàncies ja creades dels usuaris.
18. private Vector<UsuariFitxers>[] clusters
    18.1. Clústers és un Vector d’arrays on tenim la informació dels diferents clústers que s’utilitzen per al CollaborativeFiltering
19. private HashMap<String, Double>[] centroides
    19.1. Aquest atribut és un Map on es guarden parelles idItem i puntuacio per a cada centroide, aquest atribut el fem servir per al CollaborativeFiltering per no haver d’anar calculant cada cop aquesta informació.
20. private int ultimID
    20.1. L'últim userId que s’ha assignat a un usuari registrat al sistema, el fem servir per a començar a buscar userIds a partir de l'últim que hem guardat.
21. private Double puntuacioMin
    21.1. La puntuació minima que s’ha trobat en els diferents fitxers de ratings.
22. private Double puntuacioMax
    22.1. La puntuació màxima que s’ha trobat en els diferents fitxers de ratings.

Mètodes:

1. public void inicialitzar()

- Aquest mètode inicialitza tots els atributs de la classe, es crida des de la creadora de CtrlDomini.

2. public void login(String usuari, String contrasenya)
    - Si el string usuari és “Admin” i la contrasenya és Admin PersonaActual és del tipus Admin. Altrament, es crida a la funció getUsuariActiu de la classe CtrlUsuarisActiu per obtenir si existeix un usuari amb els paràmetres rebuts. En cas afirmatiu, la PersonaActual és del tipus UsuariActiu i se li associa l’userId, i les seves valoracions guardades al fitxer json. En el cas que l’usuari no existeixi o la contrasenya sigui correcta salta una excepció.
3. public void logout()
    a. Apart de tancar sessió. Aquest mètode s’encarrega de guardar els canvis que s'hi han produït en el sistema. També assigna que la PersonaActual és null.
4. public boolean sessioIniciada()
    a. Retorna cert si un usuari té una sessió iniciada al sistema (PersonaActual!=null) i retorna fals si no hi ha cap usuari amb la sessió iniciada.
5. public void registrar(String username, String contrasenya)
    - S’intenta donar d’alta a un nou usuari actiu del sistema. La funció crida a un mètode “creaSeguentID()” per tal de generar l’userID de l’usuari que volem enregistrar al nostre sistema. Es comprova abans de crear el nou usuari que no hi ha cap altre usuari amb el mateix Username que el rebut per paràmetre. En el cas que no existeixi cap, es guarden les dades del nou usuari sense assignar-li cap valoració inicialment. En el cas que intentem enregistrar un usuari amb un Username repetit salta l’excepció UsernameJaEnUs.
6. private int creaSeguentID()
    - Aquesta funció retorna el primer enter que es trobi tal que cap dels usuaris del sistema que tingui userId igual a l'enter trobat.
7. public void informaRatings(String nom)
    - Es rep per paràmetre un nom de fitxer de Ratings sense l'extensió. Es crida a la funció llegeixAllRatings i s’encarrega de llegir el fitxer rebut per paràmetre i de guardar i inicialitzar les dades necessàries.
8. public void informaKnown(String nom)
    - Es rep per paràmetre un nom de fitxer de Ratings Known sense l’extensió. Es crida a la funció llegeixAllRatings i s’encarrega de llegir el fitxer rebut per paràmetre i de guardar i inicialitzar les dades necessàries.
9. public void informaUnknown(String nom)
    - Es rep per paràmetre un nom de fitxer de Ratings Unknown sense l’extensió. Es crida a la funcio llegeixAllRatings i s’encarrega de llegir el fitxer rebut per paràmetre i de guardar i inicialitzar les dades necessàries.
10. public void recalculaClusters()
    - Aquesta funció recalcula els clústers. Es fa una crida a l'operació kmeans de la classe CollaborativeFiltering. L'operació del kmeans retorna una parella de Vector<UsuariFitxers>[] i un HashMap<String,Double>[]. Guardarem el primer element de la parella a l'atribut clústers del CtrlDomini i el segon element ho guardem a l'atribut centroides del CtrlDomini.
11. public void canviarValoracioRatings(int userId, String idItem, Double puntuacioNova)
    - Es reben per paràmetre un userId i un idItem que representen un identificador d’usuari de ratings.db i un id d’item. En el cas que no existeixi

l'usuari identificat amb userId o no existeixi l'item identificat amb idItem o l'usuari identificat amb userId no ha valorat l'ítem identificat per idItem o la puntuació rebuda per paràmetre no és vàlida, aleshores, si es dona una de les casuístiques comentades anteriorment, salta una excepció per al cas corresponent. En cas que no salti cap excepció es canvia la valoració de l'usuari userId per l'ítem idItem i s'assigna la nota rebuda per paràmetre.

12. public void afegirValoracioRatings(int userId, String idItem, Double puntuacioNova)
    - Es reben per paràmetre un userId i un idItem que representen un identificador d'usuari de ratings.db i un id d'ítem. En el cas que no existeixi l'usuari identificat amb userId o no existeixi l'ítem identificat amb idItem o l'usuari identificat amb userId ja ha valorat l'ítem identificat per idItem o la puntuació rebuda per paràmetre no és vàlida, aleshores, si es dona una de les casuístiques comentades anteriorment, salta una excepció per al cas corresponent. En cas que no salti cap excepció s'hi afegeix la valoració de l'usuari userId per l'ítem idItem i s'assigna la nota rebuda per paràmetre.
13. public void eliminarValoracioRatings(int userId, String idItem)
    - Es reben per paràmetre un userId i un idItem que representen un identificador d'usuari de ratings.db i un id d'ítem. En el cas que no existeixi l'usuari identificat amb userId o no existeixi l'ítem identificat amb idItem o l'usuari identificat amb userId no ha valorat l'ítem identificat per idItem o la puntuació rebuda per paràmetre no és vàlida, aleshores, si es dona una de les casuístiques comentades anteriorment, salta una excepció per al cas corresponent. En cas que no salti cap excepció s'elimina la valoració de l'usuari userId per l'item idItem.
14. public void afegirUsuariRatings(int userId, Vector<String> idItemsValorats, Vector<Double> puntacions)
    - En el cas que no existeixi cap usuari en el fitxer ratings.db amb el mateix userId aleshores es crea un UsuariFitxers nou identificat amb userId igual al mateix rebut per paràmetres. Es creen i s'hi assignen les valoracions amb les dades dels vectors idItemsValorats i puntuacions (idItemsValorats[i], puntuacions[i]). Finalment, s'hi afegeix al Map UsuarisRatings de CtrlDomini el nou usuari creat per aquesta operació.
15. public void canviarValoracio(String idItem, Double puntuacioNova)
    - La persona és del tipus UsuariActiu i existeix una valoració sobre l'ítem idItem feta per l'usuari que té iniciada una sessió en el sistema actualment. L'operació canvia la nota de la valoració feta per l'usuari sobre l'ítem idItem per la nota = puntuacioNova
16. public void afegirValoracio(String idItem, Double puntuacioNova)
    - La persona és del tipus UsuariActiu. L'operació afegeix una nova valoració a l'usuari amb la sessió iniciada. La valoració es fa amb l'ítem idItem i la nota de la valoració és puntuacióNova.
17. public void eliminarValoracio(String idItem)
    - La persona és del tipus UsuariActiu. El mètode elimina la valoració feta per a l'usuari amb la sessió iniciada al sistema sobre l'ítem identificat amb idItem.
18. public void canviarNomUsuari(String nouNom)
    - L'usuari amb sessió iniciada al sistema es canvia el username per al valor rebut per paràmetre (nouNom).
19. public void canviarContrasenya(String contrasenyaNova)

- L’usuari amb sessió iniciada al sistema es canvia la contrasenya per el valor rebut per paràmetre (contrasenyaNova).

20. public void eliminarUsuariActiu()
    - L’usuari amb sessió iniciada al sistema s’elimina del sistema i de la base de dades.
21. public void eliminarItem(String idItem)
    - Aquest mètode rep per paràmetre un identificador d'ítem i, si aquest existeix, elimina del sistema l'ítem identificat per al string rebut per paràmetre i conseqüentment s'elimina les valoracions d'usuaris cap aquest ítem.
22. public void afegirItem (Vector<String> nomColumnes, Vector<Vector<String>> ValorsPerColumna)
    - Afegeix al sistema l'ítem amb els atributs amb nom nomColumnes i els valors ValorsPerColumna. El sistema atorga a l'ítem un id generat automàticament. El nom de les columnes han de coincidir amb els que hi ha ja al sistema.
23. private String crearSeguentIdItem()
    - Crea automàticament un identificador d'ítem tenint en compte l'últim identificador atorgat en el sistema i els identificadors del HashMap Items.
24. public void canviarValorsAtribut(String id, Vector<String> valorsNous, String nomColumna)
    - Canvia el HashSet de valors d'atributs de l'atribut de la columna 'nomColumna' i de l'ítem identificat amb 'id' per el conjunt de valorsNous.
25. public void afegirValorAtribut(String id, String nomColumna, String valor)
    - Afegeix 'valor' al HashSet de valors de l'atribut en la columna 'nomColumna' de l'ítem identificat amb 'id'.
26. public void eliminarValorAtribut (String id, String nomColumna, String valor)
    - Elimina 'valor' de l'atribut en la columna 'nomColumna' de l'ítem identificar per 'id'.
27. public Vector<String> demanaRecomanacio1(int idUsuari, int nSortida)
    - L’operació rep per paràmetre un idUsuari d’un usuari existent al sistema i el nombre d'ítems que volem que es recomanin. L'operació crida al mètode getRecomanació amb l’algorisme recomanador que l’admin ha definit. Es retorna un conjunt de nSortida elements que representen ids d'ítems ordenats de millor a pitjor.
28. public Vector<String> demanaRecomanacio2(int idUsuari, Vector<String> items, int nSortida)
    - L’operació rep per paràmetre un idUsuari existent al sistema, un conjunt d'ítems (els ids dels ítems) per als quals l’usuari idUsuari no ha valorat cap dels ítems i volem saber l’ordre del conjunt de millor a pitjor. També rebem per paràmetre el nSortida que representa el nombre d'ítems que volem al conjunt de retorn. Es retorna el subconjunt de nSortida elements del vector ítems ordenat de millor a pitjor.
29. public Pair <Vector<Vector<String>>,Double> getQualitatRecomanacio(Vector<Integer> UsuarisAValorar, Vector<Vector<String>> ItemsAValorar, Vector<Integer> nItemsSortida)
    - Es rep per paràmetre un vector amb els identificadors dels usuaris a valorar, usuaris extrets del fitxer de known, un Vector<Vector<String>> que conté per cada usuari del vector UsuarisAValorar, un vector d’identificadors d'ítems per al qual es vol una recomanació i finalment un vector<Integer> amb el nombre

d'ítems recomanat que vol cada usuari com a sortida. De manera que l'usuari UsuarisAValorar[i] vol una recomanació per ItemsAValorar[i] i el nombre d'ítems que vol a la sortida és nItemsSortida[i].
    - El sistema demana una recomanació per cada usuari passant per paràmetre la informació que pertoca (i que s'extreu dels vectors esmentats anteriorment) i guarda el resultat temporalment en un vector<String>, on s'hi guarda els string dels identificadors d'ítem recomanats.
    - Per cada recomanació retornada es valora la qualitat d'aquesta i doncs es crida a la funció getDCGRecomanació, que retorna el DCG de la recomanació. Aquest DCG s'acumula en una variable anomenada DCG total.
    - Finalment, es calcula el DCG promig.
30. public Double getDCGRecomanacio(UsuariFitxers u, Vector<String> recomanats) throws ExcepcionsRecomanador
    - Es rep per paràmetre un usuari sobre el qual s'ha fet la recomanació, i el resultat d'aquesta (que és un conjunt de Strings que representen els identificadors d'un ítem). Aquest mètode retorna un double que representa el DCG, mesura per quantificar el ben ordenat que està un conjunt, en aquest cas la recomanació. En el fitxer de descripció d'algorismes i estructures de dades es pot trobar una explicació de com funciona aquesta operació.
31. public void llegeixAllRatings(String filename, HashMap<Integer, UsuariFitxers> cjtUsuaris)
    - Es rep per paràmetre un nom de fitxer que conté la informació dels ratings amb extensió .csv dels conjunts de dades (ratings.db, ratings.test.known, ratings.test.unknown) i un HashMap inicialment buit anomenat cjtUsuaris. Aquesta operació s'encarrega de llegir el fitxer amb nom filename situat al subdirectori DATA i crea les instàncies d'UsuarisFitxers, amb les seves valoracions creades. Cada usuari creat s'associa al map corresponent (UsuarisRatings, UsuarisKnown o UsuarisUnknown depenent del fitxer on llegim). També es van actualitzant els atributs de puntMax i puntMin.
32. public void llegeixAllItems(String filename)
    - Es rep per paràmetre un nom de fitxer amb extensió .csv que contingui la informació dels ítems que volem usar en el nostre sistema recomanador. Aquesta operació fa una crida a la capa de dades i li proporciona tota la informació del fitxer. L'operació interpreta la informació rebuda per la capa de dades i inicialitza les instàncies d'ítems, columnes, i els diferents atributs de CtrlDomini relacionats amb els ítems. També per cada ítem creat els afegeix al map Items de CtrlDomini.
33. public void guardarRatings()
    - Aquest mètode fa la crida a la capa de dades per guardar el contingut dels usuaris situats a l'atribut de CtrlDomini anomenat UsuarisRatings (map) i totes les valoracions dels usuaris.
34. public void llegirRatingsBD()
    - Aquesta operació demana la informació dels ratings a la capa de dades. Una vegada la capa de dades ha retornat la informació de ratings, es creen els usuaris i s'associen amb el map UsuarisRatings i es creen les valoracions corresponents per a cada usuari. A més, s'actualitzen els atributs puntMin i puntMax.
35. public void llegirItemsBD()

- Aquesta operació demana la informació dels ítems a la capa de dades. Una vegada la capa de dades ha retornat la informació dels ítems, es creen els ítems i s'associen al map Items i es creen les instàncies de les classes atributs i columnes.

36. public void guardarClustersCentroides()
    - Aquesta operació s'encarrega de fer una crida per a guardar les dades dels clústers i centroides. Per això fa dues crides a CtrlClusters, una per guardar els clústers i l'altre per guardar els centroides
37. public void llegirClustersCentroides()
    - Aquesta operació és l'encarregada de fer les crides a CtrlClusters necessàries per tenir la informació dels clústers i centroides. Una vegada rep la informació la guarda en els atributs de la classe CtrlDomini de "clústers" i "centroides" respectivament.
38. public void guardarCanvisUsuaris()
    - Aquest mètode guarda la informació de l'usuari que té la sessió iniciada al sistema. S'envia la informació relacionada amb l'usuari (Username, contrasenya, userId, valoracions, etc.) a la capa de dades i aquesta s'encarrega de guardar-la en el fitxer corresponent.
39. public void guardarItemsBD()
    - Aquesta operació crida al controlador de dades d'ítems per tal de guardar en un fitxer json la informació sobre els diferents ítems situats en l'atribut map anomenat Ítems.
40. public ArrayList<ArratList<String>> getValoracionsUser()
    - Aquest mètode retorna una array list d'array list de strings que representen les valoracions fetes per l'usuari que té la sessió iniciada al sistema. Aquest mètode es fa servir per enviar la informació cap a capa de presentació.
41. public ArrayList<ArrayList<String>> getNomColumnes()
    - Aquest mètode retorna una array list d'array lists de strings que representen els noms de les columnes que poden tenir els ítems. Aquesta operació es fa servir per enviar la informació cap a la capa de presentació.
42. public ArrayList<ArrayList<String>> getItemsNoValorats()
    - Aquest mètode retorna una array list d'array lists de strings que representen els atributs dels items per al qual l'usuari amb sessió iniciada al sistema no ha fet cap valoració. Aquesta funció es fa servir per enviar la informació cap a la capa de presentació.

Package DataInterface

Interfície CtrlEstat

Aquesta interfície situada en el package DataInterface es fa servir per disminuir l'acoblament entre capes. Aquesta interfície conté les operacions necessàries de control de dades per al fitxer que conté la configuració establerta per l'administrador. Aquesta interfície la implementa la classe CtrlEstatFile.java situada a la capa de dades.

Atributs:

Aquesta classe no té cap atribut, ja que és una interfície.

Mètodes:
Els mètodes només tenen la capçalera, ja que, com hem esmentat abans és una interfície i, per tant, els mètodes s'implementen a les classes que implementen les interfícies, en el cas d'aquesta interfície els mètodes els implementa la classe CtrlEstatFile.java. Anteriorment en el document ja hem esmentat que fan les operacions a la classe que s'implementa a la interfície.

1. public void saveState(int kkmeans, int knearests, String estrategiaDistanica, String estrategiaFiltering, double puntmin, double puntmax, int ultimUserId)
2. public Pair<Integer, Integer> getKsBD()
3. public Pair<String, String> getEstrategiesBD()
4. public Pair<Double, Double> getIntervalPuntuacionsBD()
5. public int getUltimID()

Interfície CtrlItem

Aquesta interfície situada en el package DataInterface es fa servir per disminuir l'acoblament entre capes. Aquesta interfície conté les operacions necessàries del control de dades per als diferents fitxers que contén informació dels ítems. Aquesta interfície la implementa la classe CtrlItemFile.java situada a la capa de dades.

Atributs:
Aquesta classe no té cap atribut, ja que és una interfície.

Mètodes:
Els mètodes només tenen la capçalera, ja que, com hem esmentat abans és una interfície i, per tant, els mètodes s'implementen a les classes que implementen les interfícies, en el cas d'aquesta interfície els mètodes els implementa la classe CtrlItemFile.java. Anteriorment en el document ja hem esmentat que fan les operacions a la classe que s'implementa a la interfície.

1. public List<String> getAll(String filename)
2. public void saveItems(Map<String, Pair<Vector<String>, Vector<Vector<String>>>>items)
3. public Map<String, Pair<Vector<String>, Vector<Vector<String> > > > readItems()

Interfície CtrlRating

Aquesta interfície situada en el package DataInterface es fa servir per disminuir l'acoblament entre capes. Aquesta interfície conté les operacions necessàries de control de dades per als diferents fitxers que contenen informació dels ratings. Aquesta interfície la implementa la classe CtrlRatingsFile.java situada a la capa de dades.

Atributs:
Aquesta classe no té cap atribut ja que és una interfície.

Mètodes:

Els mètodes només tenen la capçalera, ja que, com hem esmentat abans és una interfície i, per tant, els mètodes s'implementen a les classes que implementen les interfícies, en el cas d'aquesta interfície els mètodes els implementa la classe CtrlRatingsFile.java. Anteriorment en el document ja hem esmentat que fan les operacions a la classe que s'implementa a la interfície.

1. public List<String> getAll(String filename)
2. public void saveRatings(Vector<Integer> usrs, Vector<Integer> nVals, Vector<String> ítems, Vector<Double> puntuacio)
3. public Map<Integer, Vector< Pair <String, Double> > > readRatings()

Interfície CtrlUsuariActiu

Aquesta interfície situada en el package DataInterface es fa servir per disminuir l'acoblament entre capes. Aquesta interfície conté les operacions necessàries de control de dades per al fitxer que conté la informació dels usuaris registrats al sistema. Aquesta interfície la implementa la classe CtrlUsuarisActiusFile.java situada a la capa de dades.

Atributs:
Aquesta classe no té cap atribut, ja que és una interfície.

Mètodes:
Els mètodes només tenen la capçalera, ja que, com hem esmentat abans és una interfície i, per tant, els mètodes s'implementen a les classes que implementen les interfícies, en el cas d'aquesta interfície els mètodes els implementa la classe CtrelUsuarisActiusFile.java. Anteriorment en el document ja hem esmentat que fan les operacions a la classe que s'implementa a la interfície.

1. public Pair<Integer, Vector<Pair<String, Double> >> getUsuariActiu(String usrname, String password)
2. public void saveUser(int userid, String username, String contrasenya, Vector<Pair<String, Double> >ValoracionsUsuari);
3. public Boolean existeixUsuari(String username);
4. public Boolean existeixUsuari(int userId);
5. public void eliminarUsuari(int userId);

Interfície CtrlClusters

Aquesta interfície situada en el package DataInterface es fa servir per disminuir l'acoblament entre capes. Aquesta interfície conté les operacions necessàries per a guardar i llegir els clústers i centroides calculats per al sistema. Aquesta interfície la implementa la classe CtrlClustersFile.java situada a la capa de dades.

Atributs:
Aquesta classe no té cap atribut, ja que és una interfície.

Mètodes:
Els mètodes només tenen la capçalera, ja que, com hem esmentat abans és una interfície i, per tant, els mètodes s'implementen a les classes que implementen les interfícies.

Anteriorment en el document ja hem esmentat que fan les operacions a la classe que s'implementa a la interfície.

1. public void saveClusters(Vector<Vector<Integer>> clusters)
2. public void saveCentroids(Hashmap <String, Double> [] centroids)
3. public Vector<Vector<Integer>> readClusters()
4. public Hashmap <String, Double> [] readCentroids()

Classe FactoriaCtrl

Aquesta classe es tracta d'una factoria, ja que només necessitem una instància de la classe per tal de crear les instàncies de les altres classes. Aquesta factoria s'encarrega de crear i proporcionar les instàncies de les classes i interfícies CtrlDomini, CtrlRating, CtrlUsuariActiu, CtrlEstat i CtrlItem.

Atributs:

1. private static FactoriaCtrl factoriaCtrl
    - Aquest atribut és estàtic, ja que per fer que la classe de FactoriaCtrl sigui singleton hem de garantir que aquest atribut no sigui null.
2. private CtrlItem ItemCtrl
    - Aquest atribut té guardat una instància de la interfície CtrlItem
3. private CtrlRating RatingsCtrl
    - Aquest atribut té guardada una instància de la interfície CtrlRating
4. private CtrlUsuariActiu UsuariActiuCtrl
    - Aquest atribut té guardat una instància de la interfície CtrlUsuariActiu
5. private CtrlEstat EstatCtrl
    - Aquest atribut té guardat una instància de la interfície CtrlEstat
6. private CtrlDomini
    - Aquest atribut té guardat una instància de la classe CtrlDomini

Mètodes:

1. public static FactoriaCtrl getInstance()
    - Aquest mètode retorna la instància de factoriaCtrl, si factoriaCtrl és null aleshores es crea la instància de factoriaCtrl.
2. public CtrlDomini getCtrlDomini()
    - Aquest mètode retorna la instància de DominiCtrl, si DominiCtrl és null, aleshores es crea la instància de CtrlDomini i se li assigna a l'atribut ItemCtrl.
3. public CtrlItem getCtrlItem()
    - Aquest mètode retorna la instància d'ItemCtrl, si ItemCtrl és null aleshores es crea la instància de CtrlItem i se li assigna a l'atribut ItemCtrl.
4. public CtrlRating getCtrlRating()
    - Aquest mètode retorna la instància de RatingsCtrl, si RatingsCtrl és null, aleshores es crea la instància de CtrlRating i se li assigna a l'atribut RatingsCtrl.
5. public CtrlUsuariActiu getCtrlUsuariActiu()
    - Aquest mètode retorna la instància de UsuariActiuCtrl, si UsuariActiuCtrl és null, aleshores es crea la instància de CtrlUsuariActiu i se li assigna a l'atribut UsuariActiuCtrl.
6. public CtrlEstat getCtrlEstat()

- Aquest mètode retorna la instància de EstatCtrl, si EstatCtrl és null, aleshores es crea la instància de CtrlEstat i se li assigna a l'atribut EstatCtrl.

Package Model

Classe Persona

Aquesta classe fa referència a les persones que usaran l'app. Aquestes persones poden ser o bé Admin (administrador), usuaris actius del sistema, que s'han registrat i fan servir el sistema, o bé usuarisBD, que son aquells usuaris que no estan registrats però en fem servir la informació (extreta de ratings) per tal de poder usar el recomanador.

Atributs:

1. private int id
   Aquest atribut serveix per identificar la persona i poder diferenciar-la d'entre les altres i referenciar-la.

Mètodes:

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

Classe Admin.java:

Aquesta classe fa referència a l'administrador que configura el sistema i fa els canvis adients per a millorar-ne el funcionament. És subclasse de Persona, i, per tant, hereta els seus atributs i mètodes.

Atributs:

1. private String username
   Aquest atribut representa el nom d'usuari de l'administrador. Per defecte és "Admin" i no es pot canviar. L'username s'utilitza a l'hora de fer login per fer més senzill accedir al sistema (en comptes de recordar un nombre identificador que pot ser de moltes xifres, has de recordar un nom que tu mateix has triat).
2. private String contrasenya
   Aquest atribut representa la contrasenya del compte de l'administrador i s'utilitza a l'hora de fer el log in. Es dona una predeterminada, però, es pot, i és convenient, canviar-la.

Mètodes:

Les operacions de la subclasse Admin són getters i setters dels seus propis atributs. Cal recordar que al ser subclasse de Persona, també hereta les seves operacions.

Classe Usuari

Aquesta classe engloba els dos tipus d'usuari que hi ha al sistema, usuari actiu (persona que ha fet login i no és administrador) i usuariFitxers (usuari del qual usem les dades per a fer recomanacions però que no es pot registrar). Alhora aquesta classe és subclasse de Persona i doncs hereta els seus atributs i operacions.

Atributs:

1. private Map <String, Valoracio> ValoracionsUsuari

Mètodes:

1. public void afegirValoracio(Valoracio v)
    - Afegeix al map ValoracionsUsuaris la valoració v
2. public void canviarPuntuacio(String idItem, Double puntuacioNova) throws ExcepcionsRecomanador
    - Canvia la puntuació de la valoració sobre l'ítem idItem feta per l'usuari. Es comprova prèviament que evidentment existeix una valoració fet per l'usuari per a l'ítem identificat amb idItem (i, per tant, comprova que idItem està contingut en el map ValoracionsUsuari).
3. public void eliminarValoracio(String idItem) throws ItemNoValorat
    - S'elimina la valoració sobre l'ítem idItem feta per l'usuari. Es comprova previament que evidentment existeix una valoració feta per l'usuari per l'ítem identificat amb idItem (i, per tant, comprova que idItem està contingut en el map ValoracionsUsuari).
4. public HashMap<String,Double> getCoordenades()
    - S'obté un map amb clau String i valor Double que representa les valoracions fetes per l'usuari. String és l'identificador de l'ítem i Double la puntuació donada a aquest ítem.

Classe UsuariActiu

Aquesta classe fa referència a l'usuari, no administrador, que té fet log in (és a dir, que està usant el sistema). És subclasse de Persona i, per tant, hereta els seus atributs i mètodes.

Atributs:

1. private String username
   Aquest atribut representa el nom d'usuari del usuari actiu. L'username s'utilitza a l'hora de fer login per fer més senzill accedir al sistema (en comptes de recordar un nombre identificador que pot ser de moltes xifres, has de recordar un nom que tu mateix has triat).
2. private String contrasenya
   Aquest atribut representa la contrasenya del compte de l'usuari i s'utilitza a l'hora de fer el log in. Es comprova a l'hora d'asignar-lo que és únic per tot el sistema.

Mètodes:

Les operacions de la subclasse Usuari Actiu són getters i setters dels seus propis atributs. Cal recordar que al ser subclasse d'Usuari que és alhora subclasse de Persona, també hereta les seves operacions.

Classe UsuariFitxers

Representa els usuaris extrets de Ratings o Known. Aquests usuaris no estan registrats en el sistema, però, s'usen com a dades de training d'aquest. Hereten tot de la seva super-classe Usuari i no tenen cap atribut o mètode extra.

Classe Valoració

Aquesta classe serveix per enregistrar les valoracions que els usuaris fan sobre determinats ítems. També serveix per poder emmagatzemar temporalment en memòria les valoracions predites pel sistema (que s'usen per a retornar en ordre els ítems recomanats).

Atributs:

1. private double puntuació;
   Representa la puntuació que l'usuari que ha fet la valoració li dona a l'ítem valorat o la puntuació predita pel recomanador.
2. private Usuari u;
   Representa l'usuari que ha fet la valoració o per al qual s'ha predit una valoració.
3. private Item item;
   Representa l'ítem valorat.
4. private boolean esPredictiva;
   Indica si la Valoració ha estat predictiva o no. Si té valor null o fals és predictiva, en cas contrari és una valoració real.

Mètodes:

La classe només té getters i setters dels seus atributs.

Classe Item

Els objectes d'aquesta classe representen els ítems que els usuaris poden valorar i rebre a les recomanacions.

Atributs:

1. private String id
   És l'identificador de l'ítem, serveix per distingir un ítem de tota la resta.
2. private Vector<Atribut> atributs
   Aquest Vector representa tots els atributs de l'ítem

Mètodes:

1. public double ComparaItem(Item comparat) throws AtributsDiferents
   - Serveix per comprovar la distància/grau de similitud entre dos ítems, self i l'item comparat.

2. public double ComparaItemEucl(Item comparat) throws AtributsDiferents
   - Serveix obtenir la distància euclidiana entre dos ítems, self i l'ítem comparat.

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 ítem donat i les columnes a les diferents categories d'atributs del que està compost l'ítem. Aquesta classe serveix per saber quins valors d'atribut estan lligats a quina columna i doncs són del mateix atribut.

Atributs:

1. private int numCol;

Indica quin número de columna és. Aquest nombre s'otorga ordenadament quan es llegeix el fitxer d'ítems.
2. private String nomCol;
   Aquest atribut guarda el nom de la columna (i, per tant, el nom del Atribut de l'ítem).
3. private Double maxCol;
   Indica, en cas que els atributs compresos en ella siguin AtributNumeric, el valor màxim de la columna.
4. private Double minCol;
   Indica, en cas que els atributs compresos en ella siguin AtributNumeric, el valor mínim de la columna.
5. private Boolean esClau;
   Serveix per saber si la columna és la que permet identificar l'ítem.
6. private HashSet<Atribut> atributs;
   Aquest HashSet d'atributs representa els diferents valors de la columna, és a dir, els diferents valors d'atribut que té la columna en la matriu d'ítems.

Mètodes:
A part de getters i setters, Columna té la següent operació (que és quelcom semblant a un setter)

1. public void afegirAtribut(Atribut a)
   Aquest mètode afegeix l'Atribut passat per paràmetre al HashSet d'atribut. Si l'atribut a afegir és del tipus AtributNumeric, actualitza els valors de minCol i maxCol.

Classe Atribut

La classe Atribut representa els diferents valors d'atributs que pot tenir un ítem. 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 numèriques i un booleà que representa el compliment de diferents condicions o propietats. És per això que hi ha les tres subclasses d'atribut; Atribut Categòric, Atribut Numèric i Atribut Booleà.

Atributs:

1. private Item item
   Representa l'ítem que té aquest valor per l'atribut de la columna col.
2. private Columna col
   Representa la columna de la qual és valor d'atribut

Mètodes:
A part de getters trobem:

1. protected void ComprobaExcepcioAtributsDiferents(Atribut a) throws AtributsDiferents
   - Aquest mètode comprova si l'atribut self i el passat per paràmetre són del mateix tipus d'atribut.
2. public double ComparaAtributs(Atribut a) throws AtributsDiferents
   - Retorna el grau de similitud entre l'Atribut sobre el qual s'invoca la funció i l'Atribut que es passa com a paràmetre. Té una implementació diferent en cada subclasse, segons el seu tipus.

A més a més, també té operacions abstractes que les seves subclasses implementaran. Aquestes són getters o setters de les subclasses;

- public abstract HashSet<String> getSet();
- public abstract String getTipus();
- public abstract void afegirValor(String s);
- public abstract void canviaValors(Vector<String> vals);
- public abstract void eliminaValor(String s);

Classe AtributCategoric

AtributCategoric és una subclasse d'Atribut, per tant, hereta els seus mètodes i atributs. Serveix per poder classificar aquells valors d'atributs que fan referència a categories, noms, etc.

Atributs:

1. private HashSet<String> Categories
   Aquest hashset recull tots els valors de categoria que té l'atribut per a una columna donada i un item donat. És un hashset, ja que hi ha ítems al document items.csv on en una columna té més d'un valor.

Mètodes:

1. Tots els seus mètodes són getters o setters del HashSet Categories, que serveixen per, o bé inicialitzar el set, o afegir un valor al set, eliminar un valor del set, obtenir tot el set o obtenir un valor concret del set.

Classe AtributNumeric

AtributNumeric és una subclasse d'Atribut, per tant, hereta els seus mètodes i atributs. Serveix per poder classificar aquells valors d'atributs que fan referència a números, xifres, etc.

Atributs:

1. private HashSet<Double> Valors
   Aquest set recull totes aquelles xifres que té l'atribut per a una columna donada i un ítem donat, és a dir, els valors d'una cel·la de la matriu d'items.csv.

Mètodes:

1. public double ComparaAtributs(Atribut a) throws AtributsDiferents
   - Retorna el grau de similitud entre l'Atribut sobre el qual s'invoca la funció i l'Atribut que es passa com a paràmetre. S'estableix que la similitud pot oscil·lar entre 0 i 5 i es comparen un per un els valors dins dels HashSets Valors dels atributs.
2. private double comparaValors(double v1, double v2, double max, double min)
   - Retorna la similitud entre dos valors, aplicant la següent fórmula : 1-(|valor1-valor2|/|valorMaximColumna-ValorMinimColumna|)

La resta dels seus mètodes són getters o setters del HashSet Valors, que serveixen per, o bé inicialitzar el set, o afegir o eliminar un valor al set, obtenir tot el set o obtenir un valor concret del set.

Classe AtributPropietat

AtributPropietat és una subclasse d'Atribut, per tant, hereta els seus mètodes i atributs. Serveix per poder classificar aquells valors d'atributs que fan referència a propietats que prenen el valor cert o fals.

Atributs:

1. private HashSet<boolean> Propietats

Aquest set recull tots els valors que té l'atribut per a una columna donada i un ítem donat. És a dir, representa el contingut d'una cel·la de la matriu items.csv

Mètodes:

1. public double ComparaAtributs(Atribut a) throws AtributsDiferents
    - Retorna el grau de similitud entre l'Atribut sobre el qual s'invoca la funció i l'Atribut que es passa com a paràmetre. S'estableix que la similitud pot oscil·lar entre 0 i 5 i el que fa és extreure aquesta similitud segons el nombre de cert i fals en comú que tenen els hashsets "Propietats"

Tota la resta dels seus mètodes són getters o setters del HashSet Propietats, que serveixen per, o bé inicialitzar el set, o afegir o eliminar un valor al set, obtenir tot el set o obtenir un valor concret del set.

Classe Recomanació

Una recomanació representa el resultat d'executar els algorismes de filtering. La recomanació retorna unes Valoracions predictives sobre uns ítems concrets per a un usuari concret (qui demana la recomanació o bé l'usuari de Ratings que l'administrador està usant per a comprovar la qualitat del sistema).

Atributs:

1. private Strategy str;
   Instància que s'inicialitza a la constructora segons l'estratègia de filtering escollida. Serveix per poder cridar a les funcions de filtering necessàries per obtenir una recomanació.
2. private Usuari usuari;
   Representa l'usuari pel qual es fa la recomanació.
3. private TreeSet<Valoracio> resultat;
   Emmagatzema les Valoracions de tipus predictives que ha retornat l'estratègia de filtering.

Mètodes:

A part de getters i de la constructora, on li passen per paràmetre l'estratègia de filtering a usar i l'estratègia de distància a usar (en cas del CollaborativeFiltering) i el valor de k, trobem la següent funció:

1. public Vector<String> getRecomanacio(Usuari u, Map<String,Item> items, int n)
    - Aquest mètode crida sobre la instància de Strategy, inicialitzada a la constructora, la funció de filtering. A filtering per paràmetre se li passa l'usuari per al qual predir la recomanació, els ítems a estudiar (conjunt d'items sobre el qual extreure els ítems recomanats). Filtering ens retorna un TreeSet amb un conjunt de valoracions ordenades decreixentment. Per tal de fer el retorn, s'itera sobre les valoracions retornades per filtering n vegades, agafant doncs els identificadors dels n ítems primers i doncs els n ítems del conjunt 'items' que més haurien d'agradar a l'usuari (n es el nombre d'ítems que l'usuari vol com a retorn). Finalment es retorna un Vector amb n Strings, corresponents als n identificadors dels items recomanats com els n millors.

Classes Strategy

Interfícies que s'usen per poder aplicar el patró de disseny Estratègia. Aquest patró permet canviar en temps d'execució l'estratègia que es fa servir, és a dir, permet canviar en temps d'execució de quina manera es fa certa funció. Aquestes interfícies tenen unes operacions que obligatòriament cada classe que les usi ha d'implementar.

Concretament, la classe Strategy és la que s'usa per a l'estratègia de filtering i té l'operació: Filtering (Usuari u, Map<String,Item> items) ::TreeSet<Valoració>.

La classe StrategyDistancia s'usa per a l'estratègia del càlcul de distàncies entre usuaris i té l'operació: distancia(HashMap<String,Double> coordenades1, HashMap<String,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.

Atributs:

1. private String nom
   Aquest String guarda el nom de l'estratègia usada, en aquest cas "CollaborativeFiltering"
2. private StrategyDistancia StDist
   Guarda una instància de l'estratègia de distància (de tipus com la que s'hagi configurat), per tal de cridar sobre aquesta la funció distancia quan es necessita calcular la distància entre un usuari i un centroide.
3. private int k
   Nombre de clústers que es crearan en el k-means.

Mètodes:

1. calculaParticio(Usuari u, HashMap<String,Double>[] centroides) :: int
    - Serveix per determinar a quin clúster ha d'anar un usuari u. Retorna un enter entre 0 i k-1 corresponent al número de clúster.

2. Filtering (Usuari ua, HashMap<String,Item> items) ::TreeSet<Valoració>
    - Correspon a la implementació del mètode definit a la interfície Strategy. Per entrada rep un usuari, pel qual s'ha de fer la recomanació i el conjunt d'ítems no valorats per l'usuari i doncs pels quals predirà una nota. Demana la informació necessària a ctrlDomini per tal de poder cridar, per cada ítem del Hashmap de l'entrada, l'algorisme Slope One, que retorna un double (la nota predita per aquell ítem). Un cop rep el double, crea una Valoració amb predictiva true i l'emmagatzema al TreeSet que es retornarà com a resultat.
      Retorna, doncs, el conjunt de valoracions predites pels algorismes de filtering, ordenades de major a menor usant com a comparació les puntuacions estimades per slope one d'aquestes.
3. SlopeOne(Usuari ua, item i, Vector<UsuariFitxers> uparticio)::Double
    - Prediu, per un ítem pel qual l'usuari ua no ha fet una valoració, quina nota li donaria aquest usuari, tenint en compte les valoracions que sí que ha fet i les valoracions que altres usuaris de la seva partició han fet sobre l'ítem a valorar i els ítems valorats per l'usuari ua.
      Retorna un double, que correspon a la nota predita.

4. kmeans(HashMap<Integer, UsuariFitxers> allUsersFitxers, Map<String, Item> allItems)
:: Pair< Vector<UsuariFitxers>[] , HashMap<String,Double>[] >
    - Implementa l'algorisme k-means per separar tots els usuaris en k clústers segons la proximitat de les seves valoracions. Retorna una 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.

Atributs:

1. private String nom
   Aquest String guarda el nom de l'estratègia usada, en aquest cas "CollaborativeFiltering"
2. private int k
   Nombre de clústers que es crearan en el k-means.

Mètodes:

1. Filtering (Usuari 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 ítem.

2. 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 ítems dels com a molt k ítems més semblants a l'ítem donat.

3. 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 ítem i segons el seu grau de similitud de l'ítem que s'intenta predir amb la d l'ítem que la puntuació és coneguda.

4. ponderarItem(Pair<Integer, Double> p) :: Double
    - Serveix per, tenint en compte el nombre de vegades que s'ha repetit un ítem (és a dir, el nombre de repeticions és el nombre de vegades que ha sortit que aquest ítem és similar a ítems que a l'usuari li agraden, i per tant, el nombre d'ítems agradats semblants a aquest) i la suma de les notes predites en funció de la nota a l'ítem semblant i la similitud a aquest, calcular la nota que es preveu que l'usuari donarà a aquest ítem, 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 implementa la classe Strategy, i doncs la funció filtering, de manera que es combinen els dos algorismes de filtering: Collaborative filtering i Content Based Filtering.
Concretament, del Collaborative Filtering s'usa Slope One i de Content Based s'usa l'algorisme de Knearest. Pots trobar una explicació del funcionament d'aquesta classe al document on s'expliquen les estructures de dades i els algorismes utilitzats.

Atributs:

1. private String nom
   És el nom de l'estratègia.
2. private StrategyDistancia StDist
   Representa la instància d'estratègia de distància que s'usarà. Aquesta s'inicialitza a la constructora tenint en compte un paràmetre que li arriba.
3. static private int kn
   Representa el valor de k que s'usarà per al K-nearest

Mètodes:

1. calculaParticio(Usuari u, HashMap<String,Double>[] centroides) :: int
    - Serveix per determinar a quin clúster ha d'anar un usuari u. Retorna un enter entre 0 i k-1 corresponent al número de clúster.

2. Filtering (Usuari ua, Map<String,Item> items) ::TreeSet<Valoració>
    - Correspon a la implementació del mètode definit a la interfície Strategy. Conté la implementació de l'algorisme Slop One. Retorna el conjunt de

valoracions predites pels algorismes de filtering, ordenades de major a menor usant com a comparació les puntuacions estimades per slope one d'aquestes. Dins d'aquest es crida a Hybrid en el cas que no s'hagi pogut predir puntuació per un ítem. Hybrid que implementa i fa servir la part del content based filtering.

3. private double hybrid(Item it, HashMap<String, Item> allItems, Usuari u, Vector<UsuariFitxers> uparticio, double puntmin, double puntmax)
    - Aquesta funció calcula, cridant a l'algorisme de Knearest) els k ítems semblants per a l'item passat per paràmetre i , per cada un d'aquests k semblants es calcula una nota predictiva que l'usuari u li podria donar, a través de SlopeOne. Amb cadascuna de les notes predites pels k semblants, es fa la mitjana i el resultat d'aquesta mitjana és el valor de retorn. El retorn representa la nota que se li donarà a l'ítem pel qual no havíem pogut predir nota a través de slope one sol.
4. private PriorityQueue< Pair<Double,Item> > KNearests(Item donat, Map<String,Item> tots)
    - Aquesta funció retorna una PriorityQueue amb els k ítems més semblants a l'ítem passat per paràmetre. Per tal de trobar els ítems més semblants compara parelles d'ítems, on un és l'ítem pel qual volem els knearest i l'altre és un ítem del conjunt 'tots' i atorga una similitud a l'ítem del conjunt 'tots'. Les similituds són a través de les quals es decideix quins són els k més semblants.

Classe DistanciaEuclidiana

Atributs: -
Mètodes:

1. public Double distancia(HashMap<String,Double> coordenades1, HashMap<String,Double> coordenades2)
   Calcula la distància aplicant la fórmula de distància euclidiana sobre les valoracions a ítems en comú entre l'usuari que te coordenades1 i l'usuari o centroide amb coordenades 2. Sempre es recorre coordenades1, ja que el centroide, en cas que la distancia es calculi entre un usuari i un centroide, es col·locarà sempre en coordenades2 i doncs es complirà sempre en aquest cas que coordenades1.size()<=coordenades2.size()<=nombre d'ítems total, disminuint així el cost de l'algorisme de distància.

Classe DistanciaMitjana

Aquesta classe serveix per calcular la distància entre dos usuaris, dos centroides o un usuari i un centroide a partir de les valoracions fetes. S'aplica la fòrmula de distància mitjana.

Atributs: -
Mètodes:

1. public Double distancia(HashMap<String,Double> coordenades1, HashMap<String,Double> coordenades2)

Calcula la distància aplicant la fòrmula de distància mitjana sobre les valoracions a ítems en comú entre l'usuari que te coordenades1 i l'usuari o centroide amb coordenades 2. Sempre es recorre coordenades1, ja que el centroide, en cas que la distància es calculi entre un usuari i un centroide, es col·locarà sempre en coordenades2 i doncs es complirà sempre en aquest cas que coordenades1.size()<=coordenades2.size()<=nombre d'ítems total, disminuint així el cost de l'algorisme de distància.

Classe DistanciaPonderada

Aquesta classe serveix per calcular la distància entre dos usuaris, dos centroides o un usuari i un centroide a partir de les valoracions fetes anomenades coordenades. S'aplica la fòrmula de disancia euclidiana.

Atributs: -
Mètodes:

1. public Double distancia(HashMap<String,Double> coordenades1, HashMap<String,Double> coordenades2)
   Calcula la distància aplicant la fórmula de distància euclidiana sobre les valoracions a ítems en comú entre l'usuari que te coordenades1 i l'usuari o centroide amb coordenades 2. Al resultat que dona el càlcul de la distància euclidiana sobre els ítems valorats en comú, s'aplica una penalització pels ítems no comuns.

   Sempre es recorre coordenades1, ja que el centroide, en cas que la distància es calculi entre un usuari i un centroide, es col·locarà sempre en coordenades2 i doncs es complirà sempre en aquest cas que coordenades1.size()<=coordenades2.size()<=nombre d'ítems total, disminuint així el cost de l'algorisme de distància.

Classes comparator

Les classes comparator, que no estan en el diagrama de classes pero sí estan en codi, les usem per a poder definir com ha de funcionar la comparació d'elements dins de certes estructures. Per exemple, tenim un comparator per al TreeSet, que serveix per donar una manera de comparar les valoracions i ordenar-les en el TreeSet com volem (per puntuació descendent).

CAPA PRESENTACIÓ

La capa de presentació està formada per vistes, un controlador per cada una d'aquestes (sense tenir en compte les auxiliars ja donades per la llibreria javafx) que s'encarrega de gestionar aquestes vistes i un controlador general de presentació que s'encarrega de fer operacions comunes a més d'un controlador, els canvis de vistes i la inicialització del programa preparant les vistes i fent crides a controlador de domini per tenir-ho tot a punt.

Controladors

Controlador Presentació (el general) és un singleton i té, com a principal atribut, una referència a la instància del controlador de domini. Les operacions més destacables són, la de canvi de vistes, les d'omplir les taules de les vistes, i operacions "pont" que l'únic que fan és invocar operacions del controlador de domini (entre elles, getters i setters) per tal de reduir l'acoblament entre els diferents controladors de les vistes (capa de presentació) i el controlador de domini (capa de domini) i les operacions que creen una vista auxiliar per mostrar a l'usuari informació addicional o errors (ja incloses en la llibreria de javafx).

La resta de controladors, aquells que gestionen cadascun la seva vista associada, tenen operacions de captura d'esdeveniments (botons principalment) i el seu tractament corresponent (crides al controlador de domini i actualitzacions de la vista), i operacions de cerca per tal d'aplicar filtres dins d'una taula de la vista segons una columna en concret.

Vistes

Les vistes d'aquest programa consisteixen en fitxers ".fxml", on es disposen els diferents elements d'una vista (botons, camps de text i taules, principalment) usant la llibreria de javafx i amb crides associades als elements de la vista cap al controlador de la vista; on bàsicament es crida al controlador quan es clica un botó o quan s'escriu alguna cosa en un camp de text per tal de verificar l'entrada (que només s'entrin números, per exemple).