Dominio

PROP Grupo 33
Víctor Muñoz, Alex Moa, Artur Farriols, Mélanie Scaténa
Cuatrimestre otoño
Curso 21-22

Introducción

Este documento incluye una descripción sucinta de las clases indicadas en los diagrama de clases de la capa de dominio con tal de aclarar el propósito y funcionamiento de los principales métodos, funciones y variables. Al tratarse de un trabajo iterativo e incremental, la información aquí proporcionada están sujetos al cambio en función de las modificaciones aconsejadas por el tutor o aquéllas decididas por los integrantes del grupo.

Data Controllers

CtrlDataFactory

Descripción: factoría encargada de devolver la instancia del controlador de la capa de datos.

Funciones y métodos:

- `ICtrlData getICtrlData()`
  Devuelve la instancia del controlador de datos.

ICtrlData

Descripción: Interfaz del controlador de datos, es implementada por CtrlData de la capa de datos.

Controllers

CtrlDomain

Descripción: representa el controlador de la capa del dominio.

Variables:

CtrlItems ctrlItems
CtrlUsers ctrlUsers
CtrlAlgorithms ctrlAlgorithms
CtrlRecommendations ctrlRecommendations

Funciones y métodos:

- `String[] getDatasets()`:
  Devuelve el diccionario de ítems del sistema.

- `void setDataset(String dataset)`:
  Modifica el dataset de cada controlador.

- `boolean loadItems()`:
  Carga los ítems del sistema.

- `boolean loadUsers()`:
  Carga los usuarios del sistema.

- `boolean loadRecommendations()`:
  Carga las recomendaciones del sistema.

- `boolean loadData(String dataset)`:
  Carga todos los datos del sistema (ítems,usuarios y recomendaciones)

- `Map<String, Item> getItems()`:
  Devuelve el diccionario de ítems del sistema.

- `String[] getAlgorithms()`:
  Retorna el conjunto de algoritmos disponible.

- `String[] getPrecisions()`:
  Retorna el conjunto de precisiones disponible.

- `boolean setCurrentUser(String userId)`:
  Devuelve **true** en caso que se pueda modificar el current user por el usuario con dicho identificador y **false** en caso contrario.

- `void loadAlgorithms()`:
  Carga los algoritmos del sistema.

- `String recommend(String algorithm, String precision)`:
  Realiza una recomendación para el usuario actual del sistema.

- `String[][] searchRatingsOfCurrentUser(String itemId)`:
  Retorna la lista de ítems cuyo título se corresponda con el Item Id de la lista de ratings del usuario actual del sistema.

- `void deleteRateOfCurrentUser(String itemId)`:
  Borra el rate del ítem con el item Id dado.

- `void editRateOfCurrentUser(String itemId, Double newRate)`:
  Añade al currentUser el nuevo rate del ítem, incluso en caso de no existir un rate previo.

CtrlAlgorithms

Descripción: Controlador que contiene una instancia de todos los algoritmos y gestiona la interacción entre éstos y el sistema.

Variables:

ContentBasedFiltering contentBasedFiltering.

NewCollaborativeFiltering collaborativeFiltering.

HybridFiltering hybridFiltering.

Funciones y métodos:

`-String[] getAlgorithms()`:
Retorna el conjunto de algoritmos disponible.

`-String[] getPrecisions()`:
Retorna el conjunto de precisiones disponible.

`-void preprocessingData(Map<String, Item> itemMap, Map<String, User> userMap)`:
Realiza por cada algoritmo todos los cálculos previos para poder ejercer una recomendación.

`-Recommendation recommend(String algorithm,String precision, User currentUser, Map<String,Item> unknownItems)`:
Devuelve una recomendación para un algoritmo y usuario concreto.

CtrlItems

Descripción: Controlador encargado de gestionar los ítems del sistema.

Variables:

String dataset
CtrlDataFactory ctrlDataFactory
Map<String,Item> items

-

Funciones y métodos:

- `boolean loadItems()`
  Carga todos los ítems del sistema dado el dataset del controlador.

- `Map<String,Item> getItems()`
  Devuelve el diccionario de ítems del sistema.

- `void addItem(Item item)`
  Añade un ítem al diccionario de ítems.

- `String[] getDatasets()`
  Devuelve el listado de datasets disponibles para el sistema.

CtrlRecommendations

Variables:

String dataset.
CtrlDataFactory ctrlDataFactory.
Map<String,Recommendation> recommendations.

Funciones y métodos:

- `void addRecommendation(Recommendation recommendation)`:
  Añade una nueva recomendación al diccionario de recomendaciones.

- `void setDataset(String dataset)`:
  El dataset de la clase pasa a ser el dataset dado.

- `void initializeRecommendations()`:
  Recommendations pasa a ser un conjunto vacío.

- `String getNewId()`:
  Devuelve el id correspondiente para una nueva recomendación.

- `boolean loadRecommendations()`:

Carga todas las recomendaciones del sistema dado el dataset del controlador.

- `Map<String,Recommendation> getRecommendations()`
  Devuelve el diccionario de recomendaciones.

- `boolean saveRecommendations()`:
  Guarda las recomendaciones del sistema.

- `boolean saveNewRecommendation(Recommendation recommendation)`:
  Guarda la nueva recomendación del sistema.

CtrlUsers

Variables:

String dataset
CtrlDataFactory ctrlDataFactory.
Map<String,User> users.
Map<String,User> knownUsers.
Map<String,User> unknownUsers.
User[] currentUser.

Funciones y métodos:

- `Map<String, User> getUsers()`:
  Devuelve el diccionario de usuarios del sistema.

- `Map<String, User> getKnownUsers()`:
  Devuelve el diccionario de usuarios known del sistema.

- `Map<String, User> getUnknownUsers()`:
  Devuelve el diccionario de usuarios unknown del sistema.

- `void setDataset(String dataset)`:
  El dataset de la clase pasa a ser el dataset dado.

- `boolean loadUsers()`:

Carga todos los usuarios del sistema dado el dataset del controlador.

- `boolean saveUsers()`:
  Guarda los usuarios del sistema dado el dataset del controlador.

- `boolean saveKnownUsers()`:
  Guarda los usuarios known del sistema dado el dataset del controlador.

- `boolean saveUnknownUsers()`:
  Guarda los usuarios unknown del sistema dado el dataset del controlador.

- `boolean saveAll()`:
  Guarda todos los usuarios del sistema dado el dataset del controlador.

- `boolean existsUser(String userId)`:
  Retorna **true** en caso que exista el usuario en knownUsers y **false** en caso contrario.

- `boolean setCurrentUser(String userId)`:
  Dado un usuario existente en knownUsers, el current user pasa a contener la información de dicho usuario known y unknwon.

- `User getKnownCurrentUser()`:
  Retorna el known current user.

- `Map<String,Item> getUnknownItemsFromCurrentUser()`:
  Retorna el listado de ítems unknown del current user.

- `String[][] searchRatingsOfCurrentUser(String itemId)`:
  Retorna la lista de ítems cuyo título se corresponda con el Item Id.

- `void deleteRateOfCurrentUser(String itemId)`:
  Borra el rate del ítem con el item Id dado.

- `void editRateOfCurrentUser(String itemId, Double newRate)`:
  Añade al currentUser el nuevo rate del ítem, incluso en caso de no existir un rate previo.

Domain Model

Clase Item

Descripción: esta clase contendrá todas las variables relativas a un ítem así como las distintas funciones que permitan acceder a la información contenida en las susodichas variables.

Variables:

- id.
- tittle.
- type.
- stringAttribute.
- intAttribute.
- doubleAttribute.
- setAttribute.

Funciones y métodos:

- getters()
- setters()

Clase User

Descripción: representa a un individuo registrado en el sistema. El usuario dispondrá de toda la información que permita identificarlo y estará relacionado con aquellos ítems que haya valorado.

Variables:

- id.
- name.
- items.
- ratings.

Funciones y métodos:

- `getters()`
- `setters()`

- `rateItem(itemId)`:
  Permite modificar la valoración asociada a un determinado ítem.

- `quitItem(itemId)`:
  Elimina el ítem identificado por el parámetro del conjunto de ítems valorados por el usuario.

Clase UserGroup

Descripción: esta clase representa un clúster de usuarios.

Variables:
- id.
- users.

Funciones y métodos:
- `getters()`
- `setters()`

- `removeUser(userId)`: elimina el usuario identificado por el parámetro del conjunto.

Clase Recommendation

Descripción: clase que sirve a modo de nexo entre un usuario y los diferentes algoritmos de recomendación. Será, por lo tanto, la clase encargada de proporcionar a los algoritmos todos aquellos datos de los que puedan precisar y los filtros que el usuario desee aplicar. Además, deberá retornar el resultado al usuario.

Variables:
- id.
- user.
- score.
- precisionType.
- recommendedItems.
- algorithm.

Funciones y métodos:

- `getters()`
- `setters()`

- `executeQuery(User user, Map<String, Item> unknownItems, int Q)`: dados un usuario con una serie de valoraciones, un conjunto de ítems a valorar y un entero cuyo propósito es indicar al algoritmo el número de ítems que debe retornar, llevará a cabo una recomendación.

- `queryPrecision(User user, Map<String, Item> unknownItems, int Q, int precision)`:
  Realizará una cantidad distinta de recomendaciones en función del valor que tome el atributo *precisionType*. De este modo, podrán obtenerse resultados cuya precisión se adecue a la esperada por el usuario.

- `limitRecommendedItems(Map<String, Double> recommendedItems, int Q)`:
  Retorna un subconjunto Q de los ítems a los que el algoritmo ha asignado una valoración.

- `normalizedDiscountedCumulativeGain(User unknown)`:
  Una vez finalizada una recomendación, calculará la cercanía de los resultados obtenidos a los teóricos (almacenados en unknownRatings).

Clase Algorithm

Descripción: superclase que contiene las variables necesarias para la ejecución de los algoritmos. Ésta tendrá, por el momento, tres subclases que representarán las distintas estrategias a implementar (Content-based, Collaborative filtering e Hybrid Approach)

Variables:
- kmeans.
- knn.

Funciones y métodos:

- `getters()`
- `setters()`

- `preprocessingData()`:

Método responsable de cargar los datos almacenados en los controladores y realizar todas aquellas operaciones que permitan al algoritmo efectuar una recomendación.

- `query(user, unknownItems, Q)`:
  Dados un usuario con una serie de valoraciones, un conjunto de ítems a valorar y un entero cuyo propósito es indicar al algoritmo el número de ítems que debe retornar, llevará a cabo una recomendación.

- `discountCumulativeGain(ratedItems, unknownRatings)`:
  Una vez finalizada una recomendación, calculará la cercanía de los resultados obtenidos a los teóricos (almacenados en unknownRatings).

Clase ContentBasedFiltering

Descripción: subclase que, a partir de los atributos pertenecientes a los ítems valorados, tratará de medir la distancia entre éstos y un conjunto de ítems aún por valorar.

Variables:

- limit.

Funciones y métodos:

- `preprocessingData()`
- `query(user, unknownItems, Q)`

- `distanceStringAttributes(Map<String,String> knownItemAttributes, Map<String,String> unknownItemAttributes)`:
  Calcula la distancia entre los atributos de tipo String.

- `distanceIntAttributes(Map<String,Integer> knownItemAttributes, Map<String,Integer> unknownItemAttributes, Map<String,Double[]> attributesMinMax)`:
  Calcula la distancia entre los atributos de tipo entero.

- `distanceDoubleAttributes(Map<String,Double> knownItemAttributes, Map<String,Double> unknownItemAttributes, Map<String,Double[]> attributesMinMax):`
  Calcula la distancia entre los atributos de tipo double.
- `distanceSetAttributes(Map<String, Set<String>> knownItemAttributes, Map<String,Set<String>> unknownItemAttributes):`
  Calcula la distancia entre los conjuntos de atributos.

- `filterNeighbourhood(Map<String,Map<String,Double>> neighbourhood, Map<String,Double> knownItemsRating):`
  Dados una serie de ítems, retorna aquellos que recibirían una mejor valoración.

Clase CollaborativeFiltering

Descripción: subclase que, en base a las valoraciones de individuos similares al indicado, retornará un subconjunto de los ítems especificados
Variables:
- centroids.
- clusters.
- users.
- items .
- K.

Funciones y métodos:
- `preprocessingData()`
- `query(user, unknownItems, Q)`

- `initializeDataStructures(Map<String, Item> itemMap, Map<String, User> userMap):`
  Inicializa las estructuras de datos de las que precisa el algoritmo.

- `initializeK(Integer number):`
  Le proporciona un valor adecuado a K.

- `createCentroid(Set<String> itemsId, String id):`
  Se crea un centroide con unas valoraciones aleatorias cuyo identificador se corresponde con el pasado por parámetro.

- `initializeClusters()`:
  Una vez creados los centroides, se generan los grupos de usuarios, asociados cada uno a su respectivo centroide.

- `initializeUsers(Map<String, User> userMap)`: m
  Método encargado de inicializar la estructura de datos que contendrá los usuarios.

- `initializeItems(Map<String, Item> itemMap)`:
  Método encargado de inicializar la estructura de datos que contendrá los ítems.

- `calculateEuclideanDistance(Map<String, Double> userRatings, Map<String, Double> centroidRatings)`:
  Función que, en base a las valoraciones de un usuario y un centroide dados, calcula la distancia euclidiana.

- `inWhichCluster(String userId)`:
  Comprueba en qué clúster se encuentra el usuario identificado por el parámetro.

- `recalculateCentroids()`:

- `getSubgroupOfUsers(String itemId, String centroidId)`:
  Retorna el subconjunto de usuarios pertenecientes al clúster especificado que han valorado el ítem indicado por el parámetro.

- `userRatedItem(String userId, String ItemId)`:
  Comprueba si el usuario especificado por el parámetro ha valorado el ítem indicado.

- `calculateRating(Map<String, Double> knownRatings, Map<String, User> subgroupOfUsers, String unknownItem, User unknown)`:
  Función que calcula el rating que un usuario asignaría a un ítem empleando, para ello, la información proporcionada por los parámetros.

- `limitRecommendedItems(Map<String, Double> recommendedItems, int Q)`:
  Retorna un subconjunto Q de los ítems a los que el algoritmo ha asignado una valoración.

- `searchNullValues(Map<String, Double> recommendedItems)`: Dado un conjunto de ítems y sus respectivas valoraciones, busca y sustituye todos aquellos ítems cuyo rating sea NaN.

Clase Hybrid Approach

Descripción: subclase que, combinando los dos algoritmos mostrados previamente, retornará un subconjunto de los ítems especificados
Variables:
Funciones y métodos:

- `preprocessingData()`
- `query(user, unknownItems, Q)`