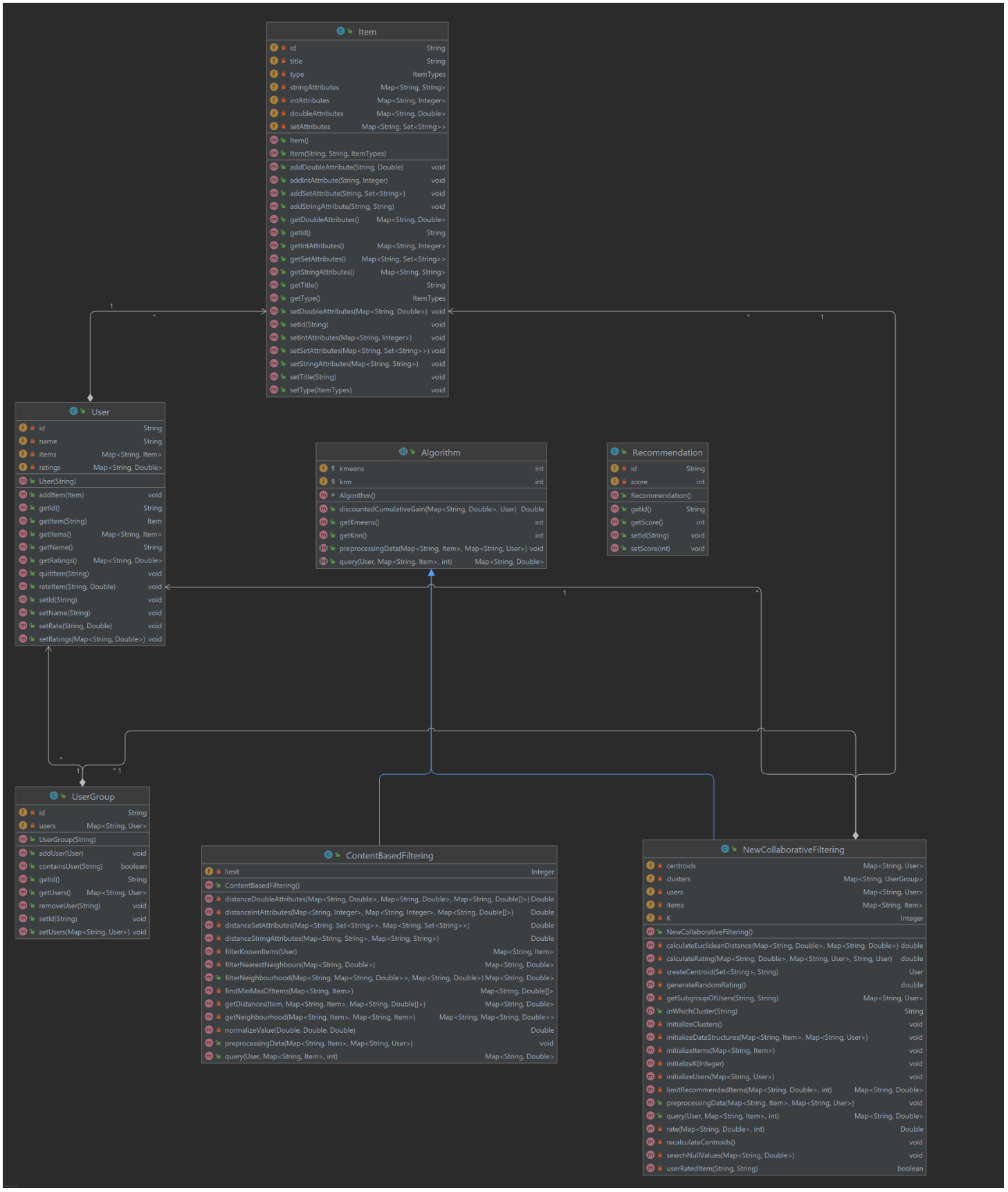
Diagrama de Clases

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 el diagrama de clases con tal de aclarar el propósito y funcionamiento de los principales métodos, funciones y variables. Al tratarse de un trabajo iterativo e incremental, el diagrama de clases así como la información aquí proporcionada están sujetos al cambio en función de las modificaciones aconsejadas por el tutor o aquellas decididas por los integrantes del grupo.



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
- score

Funciones y métodos:

- getters()

- setters()

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 Content-basedFiltering

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): 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 euclideana.
- 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.