Practica 3 BMI

Leah Hadeed & Lorenzo Vela

1. Para implementar la clase de Ratings, hemos optado por usar una serie de diccionarios: userDict (diccionario de ratings de los ítems de un usuario dado) e itemsDict (diccionario de ratings de los usuarios de un item dado). De esta manera, podemos simplemente filtrar por claves a la hora de buscar un valor de rating.
2. Para los recomendadores, modificamos las funciones de recommend y score en cada uno.
   1. AverageRecommender: Sumamos todos los ratings para un item dado y dividimos esa suma entre el numero de ratings disponibles.
   2. RandomRecommender: Devolvemos un numero aleatorio y se lo asignamos como rating a un item dado.
   3. MajorityRecommender: Calificamos los ítems por usuario basado en los resultados de la mayoría.
   4. UserKNNRecommender: Calificamos un item dependiendo de una función de similitud dada y comparadolo con los resultados de los k usuarios mas similares al usuario dado.
   5. NormUserKNNRecommender: A diferencia del anterior, aquí normalizamos el resultado del score.
   6. ItemNNRecommender: Sumamos las similitudes de los ítems para poder recomendar los usuarios ítems que son similares a los suyos.
3. Para las funciones de similitud, modificamos la función sim de la clase principal.
   1. CosineUserSimilarity: Como hemos estudiado en clase, para calcular las similitudes entre usuarios, sumamos el producto de sus ratings de cada item y dividimos entre el producto de la raíz cuadrada de sus cuadrados.
   2. CosineItemSimilarity: En vez de comparar ratings de ítems por usuario, como arriba, comparamos ratings de usuarios por item.
   3. PearsonUserSimilarity:
4. Para el ejercicio de implementar student\_test(), tomamos como base las funciones de test\_recommenders y probamos con las nuevas funciones de similitud y recomendación. Los algoritmos implementados son el PearsonUserSimilarity, CosineItemSimilarity, y el ItemNNRecommender.
5. En este apartado, comparamos los dos algoritmos de creación de redes sociales.

