基于用户的协同过滤推荐算法

基于用户的协同过滤算法的思想是有相似兴趣的用户(user)可能会喜欢相同的物品(item)。所以我们需要计算不同用户之间的“相似度”，当我们想要为一个用户 A 推荐物品时，我们希望根据与用户 A 相似的若干个用户所喜爱的物品来为 A 推荐。

变量含义：
N：记录用户看过的电影数量，如： N[“1”] = 10 表示用户 ID 为 “1” 的用户看过 10 部电影；
W：相似矩阵，存储两个用户的相似度，如：W[“1”][“2”] = 0.66 表示用户 ID 为 “1” 的用户和用户 ID 为 “2” 的用户相似度为 0.66 ；
train：用户记录数据集中的数据， 格式为： train= { user : [[item1, rating1], [item2, rating2], …], …… }
item_users：将数据集中的数据转换为 物品_用户 的倒排表，这样做的原因是在计算用户相似度的时候，可以只计算看过相同电影的用户之间的相似度（没看过相同电影的用户相似度默认为 0 ），倒排表的形式为： item_users = { item : [user1, user2, …], ……}
k：使用最相似的 k 个用户作推荐
n：为用户推荐 n 部电影

实现步骤：

加载数据
从数据集中读出数据，将数据以 { user : [[item1, rating1], [item2, rating2], …], ……} 的形式存入 train 中，并整理成形如 { item : [user1, user2, …], ……} 的倒排表，存入 item_users 中。

计算相似度矩阵
相似度公式：

$$W_{uv} = \frac{N(u,v)}{\sqrt{N(u)*N(v)}}$$

N(u,v)表示用户 u 和 v 看过相同电影的个数，N(u)表示用户 u 看过的电影个数

遍历 item_users ，统计每个用户看过的电影 item 的次数，同时遍历 item_users 中的 users ，先计算用户 u 和用户 v 看过相同电影的个数存在 W 中， 然后遍历 W ，使用上述公式计算用户 u 和用户 v 的相似度。

推荐
最后为用户 u 推荐，首先从 item_users 中获得用户 u 已经看过的电影，然后将用户按相似度排序，并取前 k 个用户，对这 k 个用户看过但用户 u 没有看过的电影计算加权评分和，即用户相似度*相似用户对电影评分进行加权求和。

基于物品的协同过滤推荐算法

基于物品的协同过滤算法的思想是给用户推荐和用户喜欢的商品相似的商品，这就涉及到如何计算两个商品的相似度了。我们并不是利用物品本身的内容属性来计算相似度（这是基于内容的推荐），而是通过分析用户行为记录（评分、购买、点击、浏览等行为）来计算两个物品的相似度，同时喜欢物品 A 和物品 B 的用户数越多，就认为物品 A 和物品 B 越相似。

变量含义：

N：记录电影被多少用户看过，如： N["1"] = 10 表示有 10 个用户看过 ID 为 "1"的电影；
W：相似矩阵，存储两个电影的相似度，如：W["1"]["2"] = 0.66 表示 ID 为 "1" 的电影和 ID 为 "2" 的电影相似度为 0.66 ；
train：用户记录数据集中的数据， 格式为： train= { user : [[item1, rating1], [item2, rating2], …], …… }
k：使用最相似的 k 个电影
n：为用户推荐 n 部电影

实现步骤：

加载数据

从数据集中读出数据，将数据以 { user : [[item1, rating1], [item2, rating2], …], ……} 的形式存入 train 中。

计算相似度矩阵

相似度公式：

$$W_{ij} = \frac{N(i,j)}{\sqrt{N(i)*N(j)}}$$

N(i,j)表示同时看过电影 i 和电影 j 的用户数，N(i)表示看过电影 i 的用户数

首先遍历 train 变量 { user : [[item1, rating1], [item2, rating2], …], ……} ，取出用户看过的电影列表，使用变量 i 遍历该用户看过的电影 ，对看过该电影的用户数（即变量 N(i) ）加一，同时使用变量 j 遍历该用户看过的电影，如果 i 和 j 不同，则 W[i][j] 加一（此时 W[i][j] 记录的是同时看过电影 i 和电影 j 的用户数）。然后遍历 W 矩阵，对 W 中的每一个元素根据上述公式进行计算

推荐

首先获得用户已经看过的电影列表，遍历该电影列表，对于每一个用户看过的电影 i ，找出与电影 i 最相似的前 k 个电影（对 W[i] 按照相似度排序），计算这 k 个电影各自的加权评分(rank):
对于可能被推荐的电影j，即电影相似度sim(i,j)*该用户对看过电影评分rating(i)的在不同i上的加权和。（注意我们只取与每个i相似的前k个电影，如果j不在i的top-k中，不参与加权）。

对 rank 按照评分倒序排序，取前 n 个没看过的电影推荐给用户即可。