CF-基于协同过滤的推荐算法

原创 chenhp 代码视界 2019-05-16

点击上方蓝字关注我们

概述

上一篇文章我们介绍了CB推荐算法，本篇文章我们将介绍另外一种推荐算法——基于协同过滤的推荐算法(Collaborative Filtering Recommendations)，下文我们统一简称为CF算法。

协同过滤推荐算法作为推荐算法中最经典的类型，包括在线的协同和离线的过滤两部分。在线协同是指通过在线数据找到用户可能喜欢的物品，离线过滤则是过滤掉一些不值得推荐的数据，比如推荐评分低的，或者推荐评分高但用户已经购买过的数据。

CF算法的数据源是基于用户历史行为和物品的矩阵数据，即UI（User-Item）矩阵数据。CF算法一般可以分为基于用户（User-Based）的协同过滤和基于物品（item-based）的协同过滤。

算法原理

1、User-Based CF

假设：

- 用户喜欢跟他过去喜欢的物品相似的物品
- 历史上相似的物品在未来也相似

方法：

- 给定用户u，找到他过去喜欢的物品的集合R(u)

- 把和R(u)相似的物品推荐给u

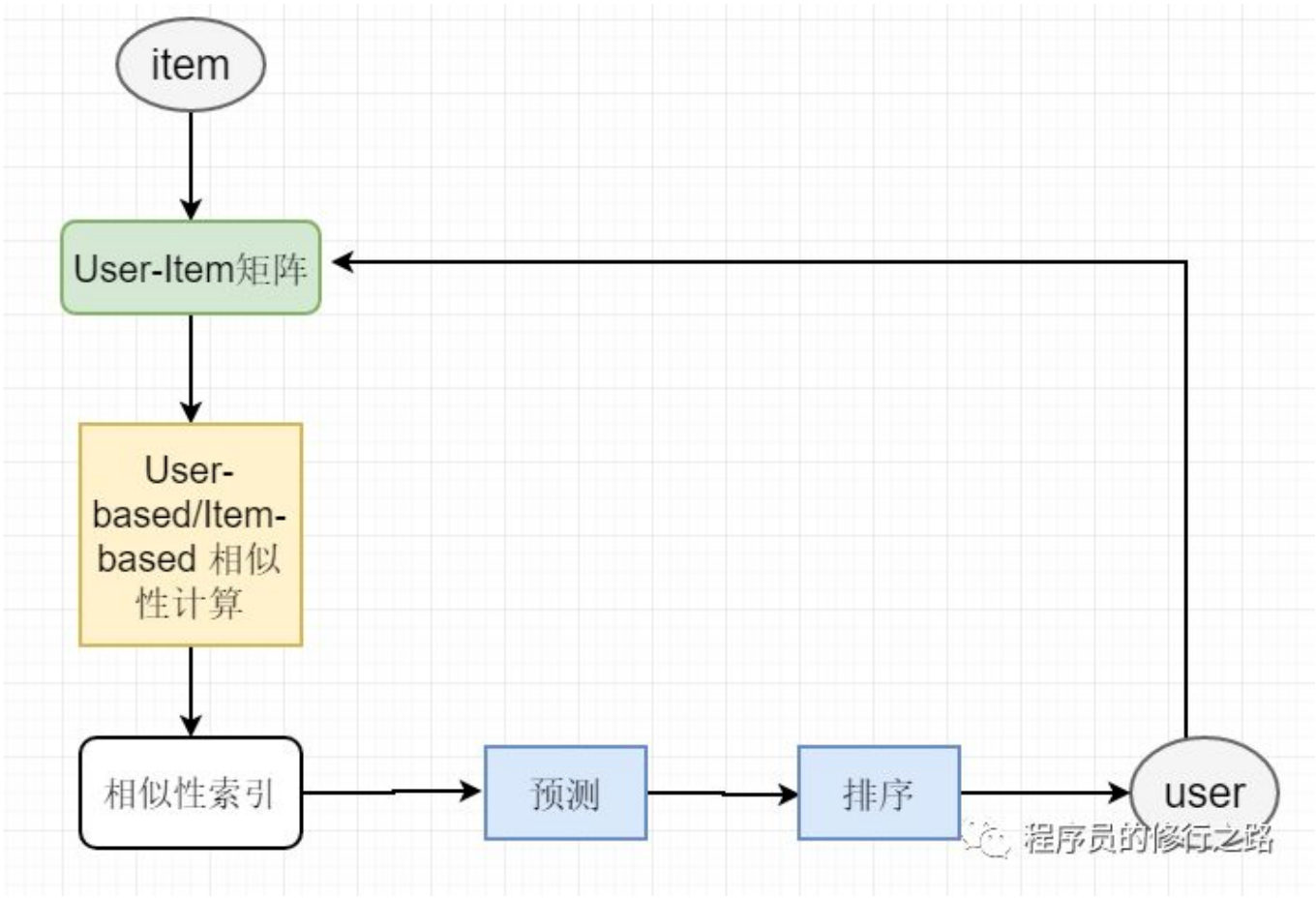
2、Item-Based CF

假设：

- 用户喜欢跟他过去喜欢的物品相似的物品
- 历史上相似的物品在未来也相似

方法：

- 给定用户u，找到他过去喜欢的物品的集合R(u)
- 把和R(u)相似的物品推荐给u



CF算法优缺点

优点：

- 充分利用群体智慧
- 推荐精度高于CB
- 利于挖掘隐含的相关性

缺点：

- 推荐结果解释性较差
- 对时效性强的Item不适用

- 冷启动问题

处理过程

1、数据准备

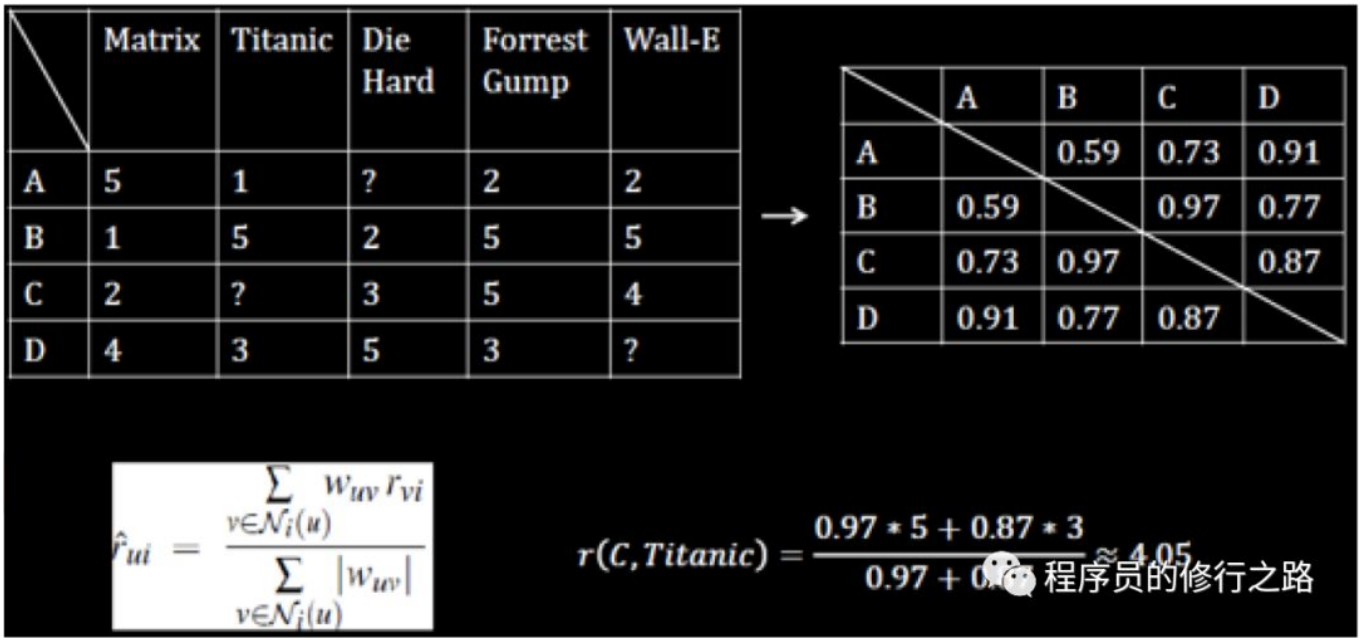
用户user_id,物品item_id，打分score（score可以是用户对某件物品的评分，或是根据用户行为计算出的偏好度得分，比如曝光、点击、收藏的加权得分，具体权重可以参考漏斗模型），数据如下：

user_id	item_id	score
id1	item1	3
id1	item2	2
id2	item1	4
id2	item2	3

2、计算相似性矩阵

CF算法的关键在于计算获得user或item的相似度矩阵，即UU矩阵和II矩阵。

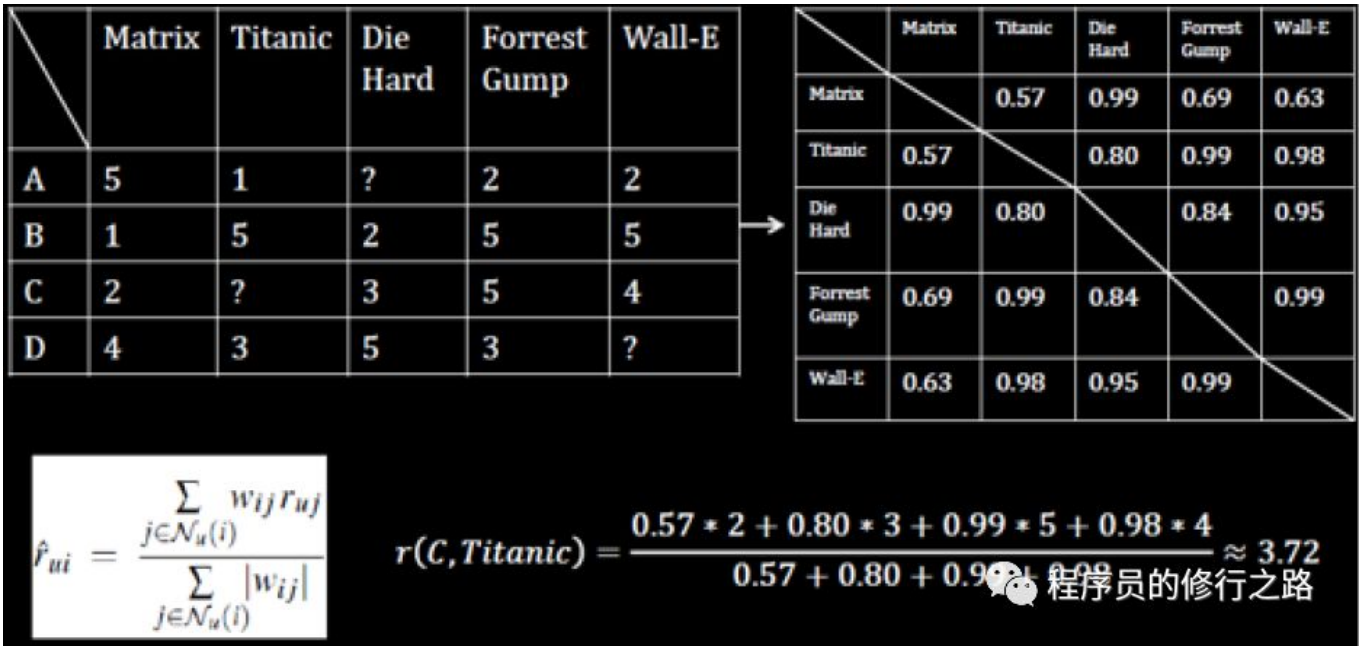
User-Based：



用户之间的相似度计算，是基于对相同的物品打过分，可以将各个分值联合起来作为一个向量，然后计算余弦相似度：

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Item-Based:



计算各个Item之间的相似度矩阵，即对两个Item都打过分的id的打分情况作为向量，同理得到item的相似度矩阵。

3、推荐

根据相似度矩阵，选择与目标用户相似度最高的几位用户，在第一张表中选取各自打分较高的物品，形成一个推荐候选集合，准备推荐给目标用户。

区别

通过两种方法，我们发现两种的分数不一样，那么该用哪个呢，哪个真实，其实这个不重要，生活中我们一般是基于用户给用户推荐Top问题，而不是打分情况，即只要排好序就可以，工作这个分数其实还是有用的，一般我们有这么个准则，哪个维度小用哪个，电商网站物品的矩阵远大于用户矩阵，规模太大有时候造成一些慢，相反一样。

那么我们来看一下这两个对比不同

	User-Based	Item-Based
性能	适用用户较少场合，如果用户多，计算用户相似矩阵代价太大	适用于物品数明显小于用户数的场合，如果物品很多，计算物品相似度矩阵代价很大
领域	时效性强，用户个性化兴趣不太明显的领域	长尾物品丰富，用户个性化需求强烈的领域
实时性	用户有新行为，不一定造成推荐结果立即变化	用户有新行为，一定会导致推荐结果的实时变化
冷启动	在新用户对很少的物品产生行为后，不能立即对他进行个性化推荐，因为用户相似度表是每隔一段时间离线计算的 新物品上线后一段时间，一旦有用户对物品产生行为，就可以将新物品推荐给对它产生行为的用户兴趣相似的其他用户	新用户只要对一个物品产生行为，就可以给他推荐和该物品相关的其他物品 但没有办法在不离线更新物品相似度表的情况下将新物品推荐给用户
推荐理由	很难提供令用户信服的推荐解释	利用用户的历史行为给用户做推荐解释，可以令用户比较信服

-- End--