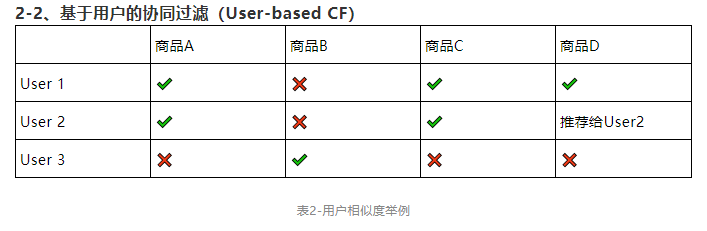
User Based CF

## 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



基于用户的协同过滤，通俗的解释是：和你相似的用户还买了什么？

我们会先找到相似的用户，然后找到此类用户喜欢的且目标用户未接触的物品，将其推荐给目标用户。

上表表2中，我们先找到相似的User 1和User 2（下文简称U1、U2）,他们都购买了商品A、C且未购买商品B。

然后将U1买过的且U2没买过的商品D，推荐给U2。

U3则仅购买商品B，其他的都未购买，推荐系统会认为他与U1、U2没有什么关联，所以我们不会对U3推荐U1和U2购买的商品。

============================================================

User-CF：基于用户对物品的偏好找到相似用户，然后将相似用户喜欢的推荐给当前用户。（假设用户会喜欢那些和他有相同喜好的用户喜欢的东西）

=========================================

举个例子，在学校时，刚进实验室的师弟都会询问学长一些诸如“我应该买什么专业书啊”、“我应该看什么论文啊”的问题。师弟之所以请教师兄，一方面是因为他们有社会关系，互相认识且信任对方，但更主要的原因是师兄和师弟有共同的研究领域和兴趣。

那么，在一个在线个性化推荐系统中，当一个用户A需要个性化推荐时，可以先找到和他有相似兴趣的其他用户，然后把那些用户喜欢的、而用户A没有听说过的物品推荐给A。基于用户的协同过滤算法主要包括以下4个步骤：

（1）构建User-Item矩阵；

（2）根据UI矩阵来计算行（用户维度）的相似度；

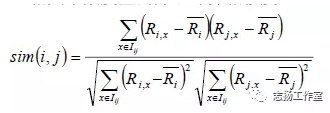
（3）选择特定用户最相似的k个用户；

（4）推荐给特定用户列表中还没有发生过行为而在相似用户列表中产生过行为的高频项目。

UserCF算法有一些缺点：随着网站的用户数目越来越大，计算用户兴趣相似度矩阵将越来越困难，其运算时间复杂度和空间复杂度的增长和用户数的增长近似于平方关系。

===============================

相似度（其实就是余弦相似度）



该公式要计算用户i和用户j之间的相似度, I(ij)是代表用户i和用户j共同评价过的物品， R(i,x)代表用户i对物品x的评分，R(i)头上有一杠的代表用户i所有评分的平均分, 之所以要减去平均分是因为有的用户打分严有的松，归一化用户打分避免相互影响。

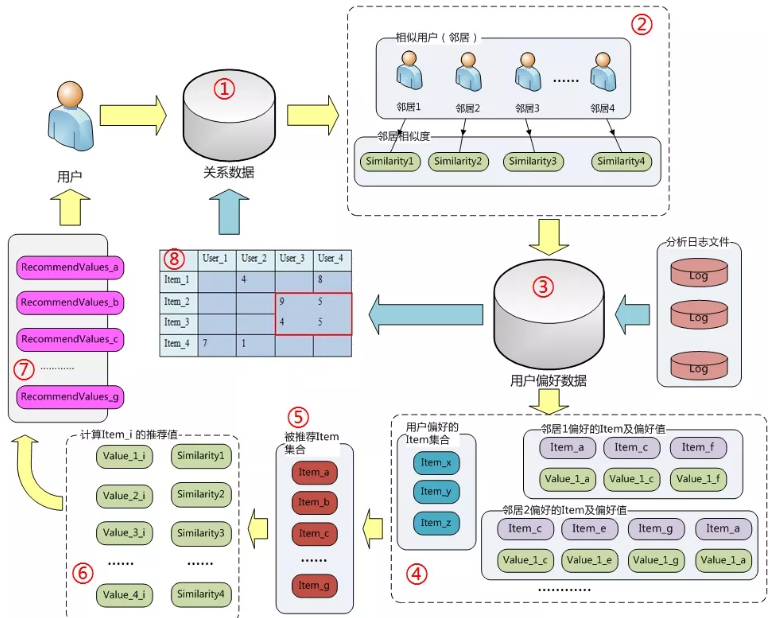
该公式没有考虑到热门商品可能会被很多用户所喜欢, 所以还可以优化加一下权重, 这儿就不演示公式了。

在实际生产环境中, 经常用到另外一个类似的算法Slope One, 该公式是计算评分偏差, 即将共同评价过的物品, 将各自的打分相减再求平均。

======================================

跟你喜好相似的人喜欢的东西，你也很有可能喜欢 --基于用户的协同过滤推荐（User-based CF）

===============================================



①.同样也是查询，只是查询的对象不一样了，查询的是与该用户相似的用户，所以一来直接查了关系数据源。以及相似用户与该用户的相似度。

②.与刚才类似，也是对数据集的一个优化，不过作用可能没那么大。

③.查询关系数据源，得到相似用户即邻居偏好过的物品；如步骤④；图中由于空间小，没有把所有邻居的偏好关系都列出来，用……表示。其次还要得到该用户偏好过的物品集合。

④.被推荐的Item集合是由该用户的所有邻居的偏好过的物品的并集，同时再去掉该用户自己偏好过的物品。作用就是得到你的相似用户喜欢的物品，而你还没喜欢过的。

⑤.集合优化同基于物品的协同过滤算法的步骤②。

⑥.也是对应类似的，依次计算被推荐集合中Item\_i 的推荐值，计算的方式略有不同，Value\_1\_i表示邻居1对，Item\_i的偏好值，乘以该用户与邻居1的相似度 Similarity1；若某个邻居对Item\_i偏好过，就重复上述运算，然后取平均值；得到Item\_i的推荐值。

⑦、⑧. 与上一个算法的最后两部完全类似，判断两个用户相似的法子和判断两个物品相似的法子一样。

================================

a.假设用户喜欢那些跟他有相似爱好的用户喜欢的东西

b.假设具有相似兴趣的用户在未来也具有相似兴趣

c.给定用户u，找到一个用户的集合N(u)，他们和u具有相似的兴趣，将N(u)喜欢的物品推荐给用户

===================================

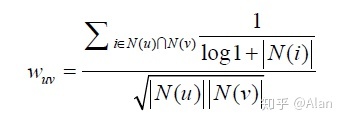
2.1UserCF算法原理

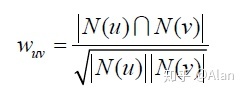
UserCF算法核心主要分两步：

* 找到与目标用户A兴趣相似的用户；
* 得到用户A相似用户评分物品中用户A没有进行评分或是浏览购买过的Item，并推荐给用户A；

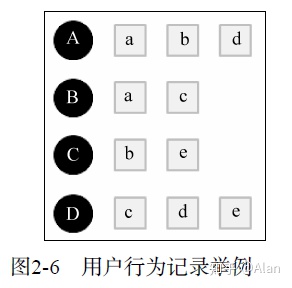
第一步：计算两个用户的兴趣相似度。这里，协同过滤算法主要利用行为的相似度计算兴趣的相似度。给定用户u和用户v，令N(u)表示用户u曾经有过正反馈的物品集合，令N(v)为用户v曾经有过正反馈的物品集合。那么，我们可以通过如下的Jaccard公式简单地计算u和v的兴趣相似度或者通过余弦公式：

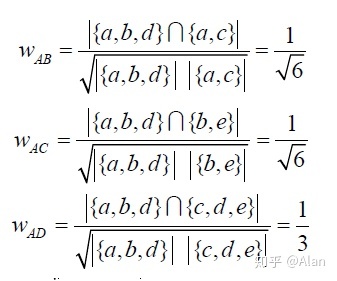
余项公式：





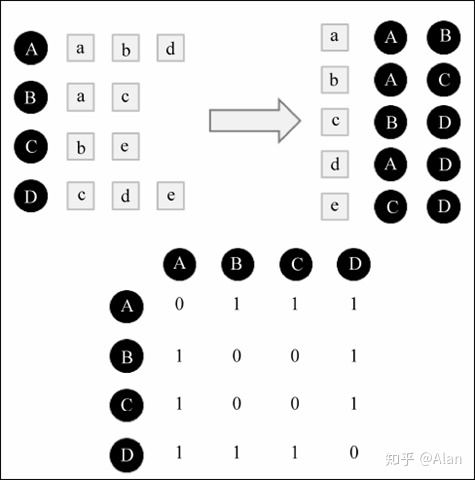
这个一个行为记录 我们可以根据余弦公式计算如下





上述算法很简单但是计算量较大，因为需要所有用户之前的复杂度 n（n-1）/2。 下面这种计算用户相似度算法通过空间换时间。

首先建立物品到用户的倒排表，然后统计每两个用户的公共物品数量（如下图所示）。



第二步：计算u对物品i的感兴趣程序：

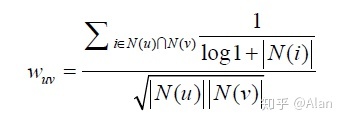
https://pic3.zhimg.com/80/v2-841822de0d889812635125e497287016_720w.jpg

得到用户之间的兴趣相似度后，UserCF算法会给用户推荐和他兴趣最相似的K个用户喜欢的物品。上面右边公式度量了UserCF算法中用户u对物品i的感兴趣程度：其中，S(u, K)包含和用户u兴趣最接近的K个用户，N(i)是对物品i有过行为的用户集合，Wuv是用户u和用户v的兴趣相似度，Rvi代表用户v对物品i的兴趣，因为使用的是单一行为的隐反馈数据，所以所有的Rvi=1。

上述推荐算法缺陷：

如果两个用户都曾经买过《新华字典》，这丝毫不能说明他们兴趣相似，因为绝大多数中国人小时候都买过《新华字典》。但如果两个用户都买过《数据挖掘导论》，那可以认为他们的兴趣比较相似，因为只有研究数据挖掘的人才会买这本书。换句话说，两个用户对冷门物品采取过同样的行为更能说明他们兴趣的相似度。因此，John S. Breese在论文中提出了如下公式，根据用户行为计算用户的兴趣相似度：

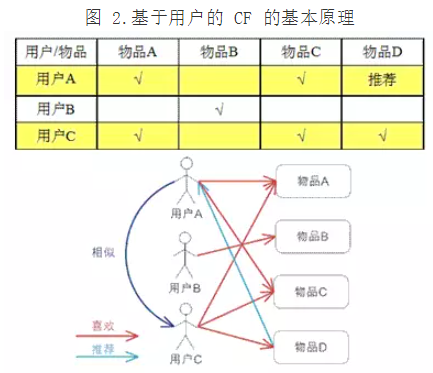
计算用户相似度的改进算法：



分子中的倒数惩罚了用户u和用户v共同兴趣列表中热门物品对他们相似度的影响。N(i)是对物品i有过行为的用户集合，越热门，N(i)越大。

===========================

基于用户的 CF 的基本思想相当简单，基于用户对物品的偏好找到相邻邻居用户，然后将邻居用户喜欢的推荐给当前用户。计算上，就是将一个用户对所有物品的偏好作为一个向量来计算用户之间的相似度，找到 K 邻居后，根据邻居的相似度权重以及他们对物品的偏好，预测当前用户没有偏好的未涉及物品，计算得到一个排序的物品列表作为推荐。图 2 给出了一个例子，对于用户 A，根据用户的历史偏好，这里只计算得到一个邻居 - 用户 C，然后将用户 C 喜欢的物品 D 推荐给用户 A。

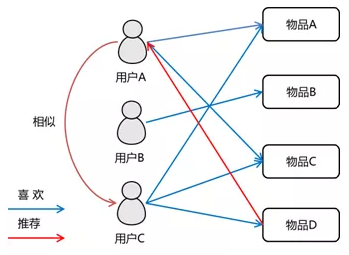


==================================

1) 第一种，基于用户 (User-based CF)

也就是说，系统通过分析一个用户和哪些用户的特征比较像，然后看看这些用户喜欢买哪类的商品，再从这些商品里挑出一些推荐给该用户。或许在一些银行或者理财产品售卖的机构会有这样的一种列表，利用这种列表同样也能够去观察哪些用户之间更相似，然后找到相似的用户，再把这些用户比较喜好的产品推荐给他。

称为User-based CF（User-based Collaborative Filtering），或者叫基于用户的协同过滤；



实现：将一个用户对所有物品的偏好作为一个向量（Vector）来计算用户之间的相似度，找到K-邻居后，根据邻居的相似度权重以及他们对物品的喜好，为目标用户生成一个排序的物品列表作为推荐，列表里面都是目标用户为涉及的物品。

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

a.假设用户喜欢那些跟他有相似爱好的用户喜欢的东西

b.假设具有相似兴趣的用户在未来也具有相似兴趣

c.给定用户u，找到一个用户的集合N(u)，他们和u具有相似的兴趣，将N(u)喜欢的物品推荐给用户

=====================================

推荐算法首先要介绍的一定是协同过滤算法了(collaborative filtering，CF)，CF算法的汇总的是所有的<user,item>行为对，有点像朋友推荐，比如用户A和用户B都喜欢差不多的东西(item相似)，用户B喜欢某样东西，但是用户A还没有喜欢，那么此时就将用户B喜欢的item推荐给用户A。

(User-Based CF)，还有一种协同推荐，即对比数据(item)，发现itemA和itemB类似(即被差不多的users喜欢)，就把某user的所有喜欢的item的类似item过滤出来作为候选推荐给该user。

# 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 |

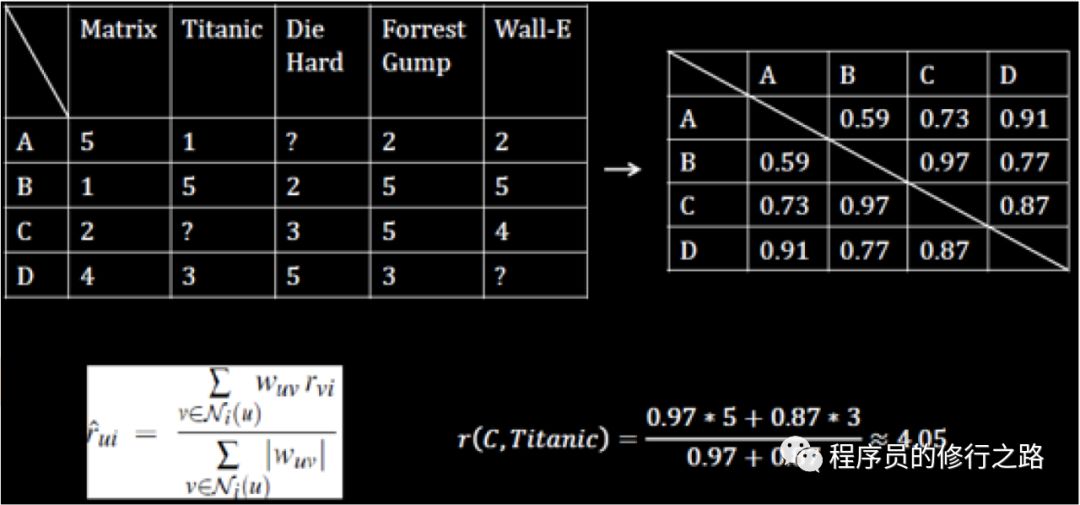
| id1 | item2 | 2 |

| id2 | item1 | 4 |

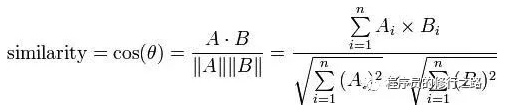
| id2 | item2 | 3 |

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。



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：



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。

# 优缺点

## 算法的优势

经常能推荐出一些意想不到的结果

进行有效的长尾item

只依赖用户行为，无需对内容进行深入了解，适用范围广。

## 算法的劣势

一开始需要大量的<user,item>行为数据，即需要大量冷启动数据

很难给出合理的推荐解释

# 参考

<https://mp.weixin.qq.com/s/RkizBXpK2CvBlwLUwaKhHg>

<https://mp.weixin.qq.com/s/rvzGQShfEU-waaFLPpRvGg>

<https://mp.weixin.qq.com/s/7SS8fBFkPB7lLnyELMf2jw>

<https://mp.weixin.qq.com/s/S3N9KFu1PXOQ67i2ge8GkQ>

<https://mp.weixin.qq.com/s/lfkTryraR22INdPzR0sidg>

<https://mp.weixin.qq.com/s/kDL7D_5NwzqprUphByKi_w>

<https://mp.weixin.qq.com/s/zvJX9k3u3f_WdtII-QF-aw>

<https://mp.weixin.qq.com/s/lWCR9CizwiqGqSrH2ISxYg>

<https://mp.weixin.qq.com/s/FKkE_wm4O4bYD3MycBFaDQ>

<https://zhuanlan.zhihu.com/p/94024379>

<https://mp.weixin.qq.com/s?__biz=MzAxNDkzNjkzMA==&mid=2247485468&idx=1&sn=dded415b69aa0ef548a87ef582566213&chksm=9b8af6>

<https://mp.weixin.qq.com/s/lNGRz0uzmKj0J3oX6egQ3A>

<https://mp.weixin.qq.com/s/N21N7EYh4xXYSlc1KwEspw>

<https://mp.weixin.qq.com/s/2nXYNT1FQSgxLQ1HM2DJhg>

<https://mp.weixin.qq.com/s/rRR-JaGaF2iNiyq62iq1bg>

=================================

对于电商场景通常会觉得 Item-CF 从性能和复杂度上比 User-CF 更优可能是基于以下几个方面：

1. 对于电商网站，用户的数量远超过物品的数量，同时物品的数据相对稳定，因此计算物品的相似度不但计算量较小，同时也不必频繁更新。
2. Item-CF的推荐成为了引导用户浏览的重要手段。（比如在购书网站上，当你看一本书的时候，推荐引擎会给你推荐相关的书籍，这个推荐的重要性远远超过了网站首页对该用户的综合推荐。)
3. 同时Item-CF便于为推荐做出解释。（在电商的网站中，给某个用户推荐一本书，同时给出的解释是某某和你有相似兴趣的人也看了这本书，这很难让用户信服，因为用户可能根本不认识那个人；但如果解释说是因为这本书和你以前看的某本书相似，用户可能就觉得合理而采纳了此推荐。）

所以单从复杂度的角度，这两个算法在不同的系统中各有优势，推荐引擎的设计者需要根据自己应用的特点选择更加合适的算法。