协同过滤

## 协同过滤的概念

协同过滤（Collaborative Filtering：CF）是推荐算法中最经典的一种，是一种基于领域的算法，早在1992年就被提出来了。它的任务是利用用户与项目评分矩阵中的已知元素来预测未知元素的评分值并将预测评分高的项目推荐给用户。其基于系统中其他用户的评分或行为进行预测和推荐，通过集体智慧的力量过滤掉那些用户不感兴趣的物品，筛选出用户感兴趣的内容。

基于协同过滤的推荐算法(Collaborative Filtering Recommendation)技术是推荐系统中应用最早和最为成功的技术之一。它一般采用最近邻技术，利用用户的历史喜好信息计算用户之间的距离，然后利用目标用户的最近邻居用户对商品评价的加权评价值来预测目标用户对特定商品的喜好程度，从而根据这一喜好程度来对目标用户进行推荐。

比如在电商推荐系统中，对用户行为进行分析得到用户喜好后，可以根据用户喜好计算相似用户和物品，然后基于相似用户或者物品进行推荐。

协同过滤的主要思想就是“物以类聚，人以群分”，主要是基于用户和商品历史交互行为信息，构建用户相似关系或者物品相似关系，根据相似关系为用户推荐和用户推荐。在数学上，其实可以理解为，对一个矩阵M（这里就是“用户-物品”相关性矩阵），已知上面部分值（已有的用户喜好程度，如打分等），去预测（填写）相应空白处的值。而填写的方法（例如坐标为(X,Y)的值），就是找到相似的多个X'，根据这些X'对Y的喜好程度推断，或者是根据X对与Y相似的Y'的喜好程度来推断。因此，协同过滤就分为两个大步走，一个是相似，一个是评分预测。

这里用了一个很有意思的思想，就是“转化”，转化说着容易，但是想着却并不容易，协同过滤在这里看来，就是在做一个转化，用户和物品的关联不好比较，那我就将物品转化为用户组合的形式，然后将用户和用户组合进行对比（user-based CF），或者是将用户转化为物品组合，然后将物品和物品组合进行对比（item-based CF），从而将两者构成联系，这是CF的精妙之处，这种方式的精妙，令其使用非常少的信息（没有用到用户性质和物品性质而直接只是两者的交互行为）就能有相对较好的推荐结果。

基于协同过滤的推荐策略的基本思想就是基于大众行为，为每个用户提供个性化的推荐，从而使用户能更快速更准确的发现所需要的信息。协同过滤可以非常直接地构建用户和物品的关系，同时具有很强的解释性，这是很多深度学习方法所没有的。因此，使用协同过滤做召回，通常可以使推荐出来的东西不会太过奇怪。

协同过滤一般是在海量的用户中发掘出一小部分和你品位比较类似的，在协同过滤中，这些用户成为邻居，然后根据他们喜欢的其他东西组织成一个排序的目录作为推荐给你。

## 协同过滤分类

协同过滤方法进一步细分为：基于内存（也称为基于领域的）的协同过滤（Memory-based CF）和基于模型的协同过滤(Model-based CF)。基于内存的协同过滤其实就是基于用户的协同过滤和基于物品的协同过滤的总称。

基于内存的协同过滤方法直接对User-Item矩阵进行研究。通过启发式的方法来进行推荐。核心要素包括相似性度量和推荐策略。选择合适的相似性度量指标来更好的表达两个项目或者用户的相似性是协同过滤的关键；最简单的推荐方法是基于大多数的推荐策略，即推荐那些大多数人产生过行为而目标用户未产生过行为的项目。

* 基于用户的协同过滤（User Based 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（先找到与目标用户有相似兴趣的其他用户，然后将他们感兴趣的内容推荐给此用户）。根据用户与物品之间的行为数据计算用户和用户的相似度，找到与用户A相似的用户B, C, D…，再把用户B, C, D…喜欢的内容推荐给用户A；
* 基于物品的协同过滤（Item Based CF），基于用户对物品的偏好找到相似的物品，然后根据用户的历史偏好，推荐相似的物品给他（先找到与目标用户喜欢的物品相似的物品，再把这些相似的物品推荐给目标用户）。根据用户与物品之间的行为数据计算物品和物品的相似度，找到与物品1相似的物品2, 3…，再把物品2, 3…推荐给看过物品1的用户们；
* 基于模型的协同过滤（Model Based CF），主流的方法可以分为：矩阵分解，关联算法，聚类算法，分类算法，回归算法，神经网络。

其中，Item-CF和User-CF是基于协同过滤推荐的两个最基本的算法。

基于用户的协同过滤（User-based CF）假设目标用户会喜欢那些和他有相同喜好的用户喜欢的东西；基于物品的协同过滤（Item-based CF）假设用户会喜欢和他以前喜欢的东西相似的东西。

## 协同过滤算法步骤

实现协同过滤，需要几个步骤：

* 收集用户偏好；
* 找到相似的用户或物品；
* 推断并排序。

# 用户偏好量化

要从用户的行为和偏好中发现规律，并基于此给予推荐，如何收集用户的偏好信息成为系统推荐效果最基础的决定因素。用户有很多方式向系统提供自己的偏好信息，而且不同的应用也可能大不相同。

量化用户偏好的数据主要包括：评分、投票、转发、保存书签、购买、点击、页面停留时间等。这些指标通常被分为两类：

* 显性反馈：用户明确表示对物品喜好的行为。这要方式是评分和喜欢/不喜欢。
* 隐性反馈：不能明确反应用户喜好的行为。（购买日志、阅读日志、浏览日志）

| 用户行为 | 类型 | 特征 | 作用 | |
| --- | --- | --- | --- | --- |
| 评分 | 显式 | 整数量化的偏好，可能的取值是 [0, n]；n 一般取值为 5 或者是 10 | 通过用户对物品的评分，可以精确的得到用户的偏好 | |
| 投票 | 显式 | 布尔量化的偏好，取值是 0 或 1 | | 通过用户对物品的投票，可以较精确的得到用户的偏好 |
| 转发 | 显式 | 布尔量化的偏好，取值是 0 或 1 | 通过用户对物品的投票，可以精确的得到用户的偏好。 如果是站内，同时可以推理得到被转发人的偏好（不精确） | |
| 保存书签 | 显示 | 布尔量化的偏好，取值是 0 或 1 | 通过用户对物品的投票，可以精确的得到用户的偏好。 | |
| 标记标签 (Tag) | 显示 | 一些单词，需要对单词进行分析，得到偏好 | 通过分析用户的标签，可以得到用户对项目的理解，同时可以分析出用户的情感：喜欢还是讨厌 | |
| 评论 | 显示 | 一段文字，需要进行文本分析，得到偏好 | 通过分析用户的评论，可以得到用户的情感：喜欢还是讨厌 | |
| 点击流 ( 查看 ) | 隐式 | 一组用户的点击，用户对物品感兴趣，需要进行分析，得到偏好 | 用户的点击一定程度上反映了用户的注意力，所以它也可以从一定程度上反映用户的喜好。 | |
| 页面停留时间 | 隐式 | 一组时间信息，噪音大，需要进行去噪，分析，得到偏好 | 用户的页面停留时间一定程度上反映了用户的注意力和喜好，但噪音偏大，不好利用。 | |
| 购买 | 隐式 | 布尔量化的偏好，取值是 0 或 1 | 用户的购买是很明确的说明这个项目它感兴趣。 | |

以上列举的用户行为都是比较通用的，推荐引擎设计人员可以根据自己应用的特点添加特殊的用户行为，并用他们表示用户对物品的喜好。

在一般应用中，我们提取的用户行为一般都多于一种，关于如何组合这些不同的用户行为，基本上有以下两种方式：

* 将不同的行为分组：一般可以分为“查看”和“购买”等等，然后基于不同的行为，计算不同的用户/物品相似度。类似于当当网或者 Amazon 给出的“购买了该图书的人还购买了 ...”，“查看了图书的人还查看了 ...”
* 根据不同行为反映用户喜好的程度将它们进行加权，得到用户对于物品的总体喜好。一般来说，显式的用户反馈比隐式的权值大，但比较稀疏，毕竟进行显示反馈的用户是少数；同时相对于“查看”，“购买”行为反映用户喜好的程度更大，但这也因应用而异。

收集了用户行为数据，我们还需要对数据进行一定的预处理，其中最核心的工作就是：减噪和归一化。

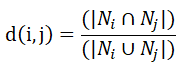
* 减噪：用户行为数据是用户在使用应用过程中产生的，它可能存在大量的噪音和用户的误操作，我们可以通过经典的数据挖掘算法过滤掉行为数据中的噪音，这样可以使我们的分析更加精确。
* 归一化：如前面讲到的，在计算用户对物品的喜好程度时，可能需要对不同的行为数据进行加权。但可以想象，不同行为的数据取值可能相差很大，比如，用户的查看数据必然比购买数据大的多，如何将各个行为的数据统一在一个相同的取值范围中，从而使得加权求和得到的总体喜好更加精确，就需要我们进行归一化处理。最简单的归一化处理，就是将各类数据除以此类中的最大值，以保证归一化后的数据取值在 [0,1] 范围中。

进行的预处理后，根据不同应用的行为分析方法，可以选择分组或者加权处理，之后我们可以得到一个用户偏好的二维矩阵，一维是用户列表，另一维是物品列表，值是用户对物品的偏好，一般是[0,1]或者[-1, 1]的浮点数值。

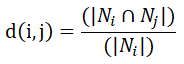
# 相似性度量

相似性度量是一个非常常见的问题，衡量两者的相似性，其实就是衡量两者的差距有多大，即“距离”，这里其实就是计算距离的问题，不同点只是在实际中用什么数据来衡量距离了。那么在推荐系统，实质上就是从喜好程度来衡量距离，换言之，两个人共同喜欢的东西越多，那么两个人就越相似（user-based），或者是两个物品共同喜欢的人越多，这两个物品就越相似（item-based），如果是一个“非黑即白”的二分类问题，那很简单，就是一个共现问题了，然而如果是一个打分问题，例如0-5分，那就是一个非常灵活的距离问题了。

对于共现问题，通过集合的形式去计算就很清楚。



i和j的距离，就是两者共同喜欢占两者之一喜欢的比例，之所以先讲这个，因为这个是满足“距离”所要求的自反性，即i与j的距离要和j与i的距离相等，但是实际上，在推荐系统中，其实衡量i和j的关系，只需要考虑i本身即可，并不要求了解j喜欢而i不喜欢的东西，因此又可以改进为：



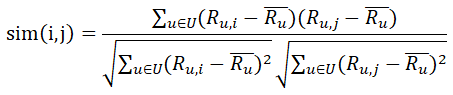
而在更为灵活的距离问题中，则要考虑连续型的关系了，因此在这里抛开例如只有0,1,2,3,4分之类的判断，而直接考虑连续型的问题，在推荐系统中，一般采用余弦相似度、欧几里德距离、皮尔逊相关系数、杰卡德系数、曼哈顿距离、切比雪夫距离、马氏距离。

关于各种度量距离的方法，请参考，这里就不一一列举了。

## 调整后余弦相似度

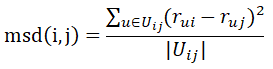
余弦相似度更多的是从方向上区分差异，而对绝对的数值不敏感。因此没法衡量每个维数值的差异，会导致这样一个情况：比如用户对内容评分，5分制，X和Y两个用户对两个内容的评分分别为(1,2)和(4,5)，使用余弦相似度得出的结果是0.98，两者极为相似，但从评分上看X似乎不喜欢这2个内容，而Y比较喜欢，余弦相似度对数值的不敏感导致了结果的误差，需要修正这种不合理性，就出现了调整余弦相似度，即所有维度上的数值都减去一个均值，比如X和Y的评分均值都是3，那么调整后为(-2,-1)和(1,2)，再用余弦相似度计算，得到-0.8，相似度为负值并且差异不小，但显然更加符合现实。

调整后的余弦相似度通过从每个共同评分对中减去各自用户的平均评分来抵消该缺点，定义如下：



## MSD均方差相似度

仍然考虑物品i和j的相似度，MSD考虑的角度为同时喜欢物品i和j的用户对于这两个物品的评分差距程度：



上式表示均方差，值越小，物品i和j相似度越大。为了与之前的相似度表示一致（值越大，物品相似度越大），定义相似度为：



## 相似度量选择

在选择相似性度量时，可根据以下几点进行选择：

• 当数据受用户偏好/用户的不同评分尺度影响时，请使用皮尔逊相似度

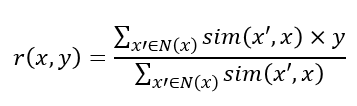
• 如果数据稀疏，则使用余弦（许多额定值未定义）

• 如果数据不稀疏并且属性值的大小很重要，使用欧几里得距离

• 建议使用调整后的余弦（Adjusted Cosine Similarity）进行基于商品的方法来调整用户偏好。

# 评分预估

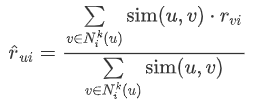
评分预估在这里实际上可以理解为一个“数学期望”，其实就是一个加权平均，权重就是相似人和目标用户的相似度，平均值是喜好程度的平均值，根据多个相似人的共同喜好程度预估目标用户的喜好程度，根据排序得到最终的推荐结果。



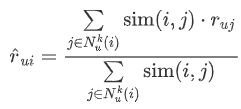
值得注意的是，在上面计算的是距离，距离越大两则越不相近，而在这里sim是指相似度，相似度越大两者越相近，这两者的转化一般用倒数、相反数等方式可以转化。

如此一来，就得到了用户和物品的关系，排序后即可完成推荐。

## 基于相似用户的加权平均

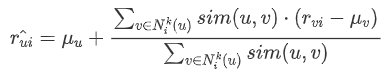


OR

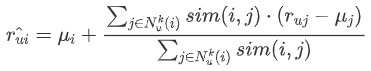


式中表示对当前商品i有评分行为的其他用户中与当前用户u最相似的k个用户。

## 基于相似用户并考虑用户评分均值

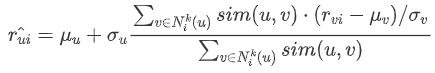


OR

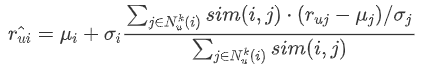


式中表示用户u评分的平均值。

## 基于相似用户并考虑用户评分均值、方差

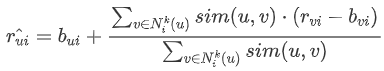


OR

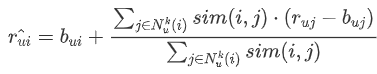


式中表示用户u评分的标准差。

## 基于相似用户并考虑用户评分偏置



OR



其中是基线模型中用户u给物品i打分的预估值。为所有用户评分的均值。为user偏差（如果某用户比较苛刻，打分都相对偏低，则会为负值。相反，如果某用户经常对很多商品都打正分，则bu为正值）。为item偏差，反映商品受欢迎程度。

# 协同过滤优缺点

## 优点

1. 推荐精度高于基于内容的推荐
2. 能够过滤难以进行机器自动内容分析的信息
3. 充分利用群体智慧，挖掘隐含的相关性，避免了内容分析的不完全和不精确，并且能够基于一些复杂的，难以表述的概念（如信息质量、个人品位）进行过滤
4. 有推荐新信息的能力，可以发现内容上完全不相似的信息。这也是基于协同过滤的推荐算法和基于内容的推荐一个较大的差别，基于内容的推荐很多都是用户本来就熟悉的内容，而基于协同过滤的推荐可以发现用户潜在的但自己尚未发现的兴趣偏好
5. 能够有效地使用其他相似用户的反馈信息，减少用户的反馈量，加快个性化学习的速度
6. 能起到意想不到的推荐效果，经常能推荐出来一些惊喜结果
7. 只依赖用户行为，不需要对内容进行深入了解，使用范围广
8. 对推荐对象没有特殊的要求，能处理非结构化的复杂对象，如音乐、电影

## 缺点

1. 对时效性强的Item不适用
2. 需要大量冷启动数据
3. 很难给出合理的推荐解释
4. 用户对商品的评价非常稀疏，这样基于用户的评价所得到的用户间的相似性可能不准确（即稀疏性问题）
5. 随着用户和商品的增多，系统的性能会越来越低
6. 如果从来没有用户对某一商品加以评价，则这个商品就不可能被推荐（即最初评价问题）
7. 计算资源有限导致的扩展性问题

# Item-CF和User-CF如何选择

## user和item数量分布以及变化频率

如果user数量远远大于item数量，采用Item-CF效果会更好，因为同一个item对应的打分会比较多，而且计算量会相对较少。例如：购物网站、技术博客、文章等不常更新、数据稳定的网站或APP；（小说APP）

如果item数量远远大于user数量，则采用User-CF效果会更好，原因同上。例如：新闻类、短视频、热点多、社交性质重、需常更新数据的网站或APP；（抖音、微博）

在实际生产环境中，有可能因为用户无登陆，而cookie信息又极不稳定，导致只能使用item-cf

如果用户行为变化频率很慢(比如小说)，用User-CF结果会比较稳定

如果用户行为变化频率很快(比如新闻, 音乐, 电影等)，用Item-CF结果会比较稳定

## 用户特点

UserCF算法适用于用户邻居多情况，例如：微信、微博、支付宝等

ItemCF算法适用于用户喜欢他以前购买过物品类型相似的物品，喜欢同类型物品的自相似度要大的情况；如果用户个性化需求强烈，则ItemCF合适，且ItemCF的可解释性较强；要求对新用户友好，ItemCF比较合适。

## 相关和惊喜的权衡

item-based出的更偏相关结果，出的可能都是看起来比较类似的结果

user-based出的更有可能有惊喜，因为看的是人与人的相似性，推出来的结果可能更有惊喜

## 数据更新频率和时效性要求

当物品数量、特征相对固定的情况下，更多采用基于物品的协同过滤（item-based CF）。因为相对稳定，一定程度上意味着不需要实时计算，对服务器的压力就很小了。

当物品和用户量都非常大，这样会造成购买的物品重叠性较低，我们很难才能找到相似的用户，Item-Based CF就不适用了。

而在如内容类媒体，微博、新闻网站等，物品（内容）的数量、特征都在不断的变化，去计算物品的相似度性能消耗反而更大了。而新闻媒体更倾向于群体的喜好，这个时候使用User-Based CF也更加的合适；对于像影视，音乐之类的还是可以采用item-cf的。

## 推荐系统多样性

* 单用户多样性：ItemCF算法小于UserCF算法多样性丰富，ItemCF算法覆盖面小、丰富度低
* 多用户多样性：ItemCF算法大于UserCF算法多样性丰富，因为UserCF算法注重社会热点物品

ItemCF算法推荐具有新颖性，容易发现推荐长尾里的物品，ItemCF算法计算精度小于UserCF算法。如果考虑多样性，那么ItemCF算法计算精度大于UserCF算法，UserCF算法推荐长尾物品能力不足。

|  |
| --- |
| 推荐的多样性，有两种度量方法：  　　第一种度量方法是从单个用户的角度度量，就是说给定一个用户，查看系统给出的推荐列表是否多样，也就是要比较推荐列表中的物品之间两两的相似度，不难想到，对这种度量方法，Item CF的多样性显然不如User CF的好，因为Item CF的推荐就是和以前看的东西最相似的。  　　第二种度量方法是考虑系统的多样性，也被称为覆盖率(Coverage)，它是指一个推荐系统是否能够提供给所有用户丰富的选择。在这种指标下，Item CF的多样性要远远好于 User CF，因为User CF总是倾向于推荐热门的，从另一个侧面看，也就是说，Item CF的推荐有很好的新颖性，很擅长推荐长尾里的物品。所以，尽管大多数情况，Item CF的精度略小于User CF，但如果考虑多样性，Item CF却比 User CF好很多。 |

## 适用场景

　　在非社交网络的网站中，内容内在的联系是很重要的推荐原则，它比基于相似用户的推荐原则更加有效。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。可以看到，在这种情况下，Item CF 的推荐成为了引导用户浏览的重要手段。同时 Item CF 便于为推荐做出解释，在一个非社交网络的网站中，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。

　　相反的，在现今很流行的社交网络站点中，User CF是一个更不错的选择，User CF加上社会网络信息，可以增加用户对推荐解释的信服程度。

## 用户对推荐算法的适应度

　　前面我们大部分都是从推荐引擎的角度考虑哪个算法更优，但其实我们更多的应该考虑作为推荐引擎的最终使用者 -- 应用用户对推荐算法的适应度。

　　对于User CF，推荐的原则是假设用户会喜欢那些和他有相同喜好的用户喜欢的东西，但如果一个用户没有相同喜好的朋友，那User CF的算法的效果就会很差，所以一个用户对的CF算法的适应度是和他有多少共同喜好用户成正比的。

Item CF算法也有一个基本假设，就是用户会喜欢和他以前喜欢的东西相似的东西，那么我们可以计算用户喜欢的物品的自相似度。一个用户喜欢物品的自相似度大，就说明他喜欢的东西都是比较相似的，也就是说他比较符合Item CF方法的基本假设，那么他对Item CF 的适应度自然比较好；反之，如果自相似度小，就说明这个用户的喜好习惯并不满足Item CF 方法的基本假设，那么对于这种用户，用Item CF方法做出好的推荐的可能性非常低。

# User Based CF与Item Based CF的区别

基于用户的协同过滤（user cf），是先找相似的人，再找相似的人喜欢的物品。UserCF算法注重用户所在的兴趣小组，给用推荐兴趣小组的热门物品，更注重社会化；

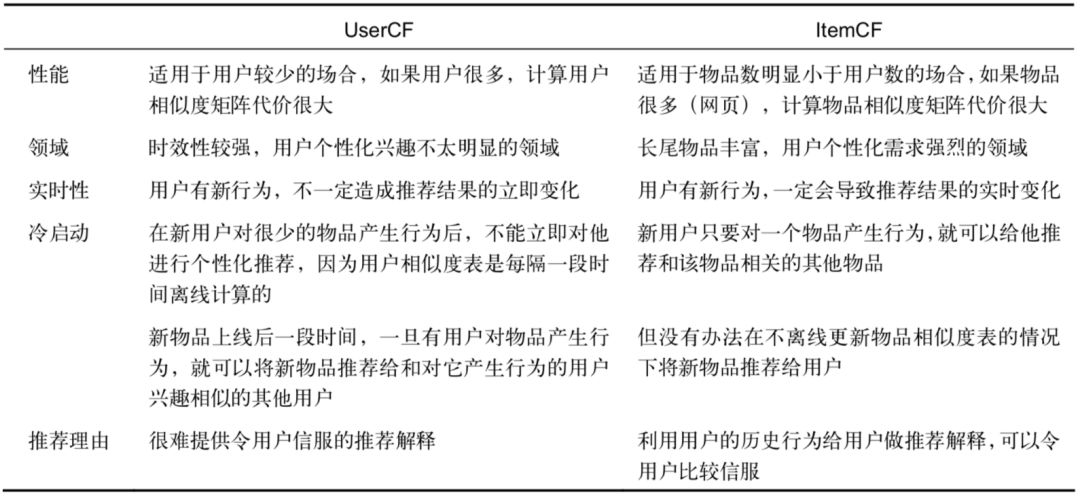
基于物品的协同过滤（item cf），则是找到和某个物品相关的物品。Item算法注重用户的过往行为的历史物品，用户本省兴趣变化和继承，对用户影响更大，因此更注重个性化；

UserCF 给用户推荐那些和他有共同兴趣爱好的用户喜欢的物品，而 ItemCF 给用户推荐那些和他之前喜欢的物品类似的物品。

UserCF 的推荐结果着重于反映和用户兴趣相似的小群体的热点，而 ItemCF 的推荐结果着重于维系用户的历史兴趣。

可以看到，UserCF和ItemCF各有优点，下面结合实例说明。比如在新闻网站中，用户的兴趣不是特别细化，绝大多数用户都喜欢看热门的新闻。即使是个性化，也是比较粗粒度的，比如有些用户喜欢体育新闻，有些喜欢社会新闻，UserCF可以给用户推荐和他有相似爱好的一群其他用户今天都在看的新闻，这样在抓住热点和时效性的同时，保证了一定程度的个性化。UserCF适合用于新闻推荐的另一个原因是从技术角度考量的。因为作为一种物品，新闻的更新非常快，每时每刻都有新内容出现，而ItemCF需要维护一张物品相关度的表，如果物品更新很快，那么这张表也需要很快更新，这在技术上很难实现。绝大多数物品相关度表都只能做到一天一次更新，这在新闻领域是不可以接受的。而UserCF只需要用户相似性表，虽然UserCF对于新用户也需要更新相似度表，但在新闻网站中，物品的更新速度远远快于新用户的加入速度，而且对于新用户，完全可以给他推荐最热门的新闻，因此UserCF显然是利大于弊。但是，在图书、电子商务和电影网站，比如亚马逊、豆瓣、Netflix中，ItemCF则能极大地发挥优势。首先，在这些网站中，用户的兴趣是比较固定和持久的。这些系统中的用户大都不太需要流行度来辅助他们判断一个物品的好坏，而是可以通过自己熟悉领域的知识自己判断物品的质量。因此，这些网站中个性化推荐的任务是帮助用户发现和他研究领域相关的物品。此外，这些网站的物品更新速度不会特别快，一天一次更新物品相似度矩阵对它们来说不会造成太大的损失，是可以接受的。

从技术上考虑，UserCF需要维护一个用户相似度的矩阵，而ItemCF需要维护一个物品相似度矩阵。从存储的角度说，如果用户很多，那么维护用户兴趣相似度矩阵需要很大的空间，同理，如果物品很多，那么维护物品相似度矩阵代价较大。在图书、电子商务网站中，物品的数目则是比较少的。新闻网站是个例外，物品的相似度变化很快，而且物品数目庞大。



# 问题

## 问题1：冷启动及对应方案

### 用户冷启动

• 提供热门排行榜，等用户数据收集到一定程度再切换到个性化推荐

• 利用用户注册时提供的年龄、性别、IP、登录时间等数据做粗粒度的个性化

• 利用用户社交网络账号，导入用户在社交网站上的好友信息，然后给用户推荐其好友喜欢的物品

• 在用户新登录时要求其对一些物品进行反馈，收集这些兴趣信息，然后给用户推荐相似的物品

### 物品冷启动

• 将新物品推荐给可能对它感兴趣的用户，利用内容信息，将他们推荐给喜欢过和它们相似的物品的用户

• 物品必须能够在第一时间展现给用户，否则经过一段事件后，物品的价值就大大降低了

• UserCF和ItemCF都行不通，只能利用Content based解决该问题，频繁更新相关性数据

### 系统冷启动

• 引入专家知识，通过一定高效方式迅速建立起物品的相关性矩阵

## 问题2：Item\_Based和Content-Based的区别

区别在于的CB中相似度是根据item的属性向量计算得到，而CF中是根据所有用户对item的评分向量计算得到。

## 问题3：ALS 交替最小二乘（Alternating Least Squares）

• 算是ml中对CF算法的一种优化

• 对于一个users-items-score的评分数据集，ALS会建立一个users\*items的m\*n的矩阵。其中，m为users的数量，n为items的数量。

• 这个数据集中，并不是每个用户都对每个产品进行过评分，所以这个矩阵往往是稀疏的，用户i对产品j的评分往往是空的。

• ALS所做的事情就是将这个稀疏矩阵通过一定的规律填满，这样就可以从矩阵中得到任意一个user对任意一个item的评分，ALS填充的评分项也称为用户i对物品j的预测得分。

## 问题4：用户兴趣变化（基于时序的协同过滤推荐算法）

由于传统的协同过滤算法忽略了随着时间变化而用户的兴趣也在不断发生变化这一问题，即存在用户兴趣漂移现象。传统的协同过滤推荐算法只是单一地通过评分来分析用户的兴趣爱好，统一地将评分用1~5分代表用户的喜爱程度，其时效性不足。

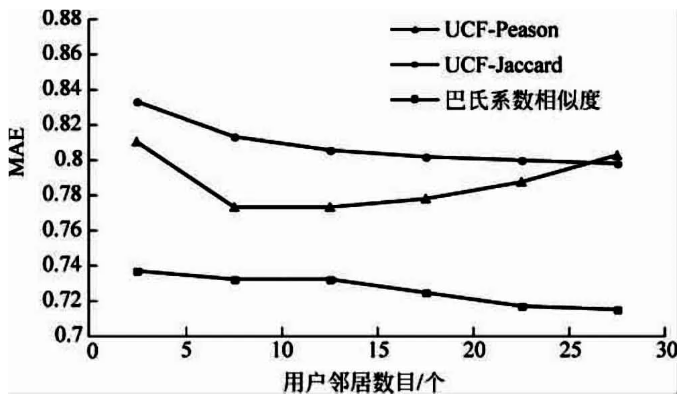
随着时间推移，用户的关注点在不断变化，如何捕获这一动态的时间效应是一个难题。

提出了基于巴式系数改进相似度的协同过滤推荐算法。

针对传统协同过滤推荐算法中评分数据稀疏性及所造成的推荐质量不高问题，提出一种巴氏系数（Bhattacharyya Coefficient）改进相似度的协同过滤推荐算法。在基于近邻协同过滤推荐算法基础上，利用巴氏系数改进相似度计算方法，在计算相似度时不仅依靠两个用户间的共同评分而是所有评分，首先利用Jaccard相似度来计算用户间的全局相似度；其次使用巴氏系数获得评分分布的整体规律，并结合Pearson相关系数来计算其局部相似度；最后融合全局相似度和局部相似度得到最终的相似度矩阵。

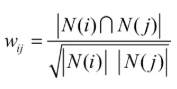
巴氏系数（Bhattacharyya Coefficient, BC）是对两个统计样本的重叠的近似计算，可用来对两组样本的相关性进行测量，已广泛应用于信号处理、模式识别研究领域。在统计学中巴氏系数用于测量两种离散概率分布的可离性，衡量两个概率分布之间的相似度。

随着用户邻居数目k的增大，BCCF算法其MAE值总体趋于减少，当邻居数达到30时MAE值基本稳定；相对于传统的Jaccard和Pearson相关系数计算方法，BCCF算法的MAE值最低且相对稳定。

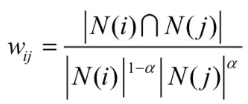


## 问题5：热门物品问题

热门物品问题是指，在物品过热的情况下，绝大部分的人都会与热门物品有交互行为，此时ItemCF就会认为这个热门物品与其他的物品都很相似，此时会出现一些问题，首先看看在0-1的情况下评判相似度的公式：

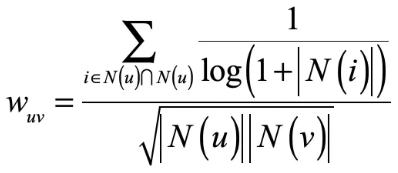


其中，N(j)表示喜欢j产品的用户集合，|N(j)|表示喜欢j产品的用户个数，如果物品j非常流行，分子会无限接近|N(j)|，尽管分母做了控制，但是可以发现物品j和大量物品都很接近，此时很可能就一直推这个热门物品，其他商品也被挤下去，再者，有些用户其实并不喜欢。因此，最简单的方式就是使用惩罚，给热门商品进行降权，从而得到解决。



可以看到，alpha=0.5时就是上面的公式，通过增加alpha就能够达到惩罚的效果，在分母，alpha的变大能够使|N(j)|更被看重，从而另其被除的更多。

在ItemCF会出现的问题在UserCF中同样会出现，比如很多人都会买《新华字典》但是不一定会买《推荐系统实践》，此时前者会影响两人的相似度，因此可以进行削弱，此处的可进行降权，降权的方式如下：



其中，N(u)和N(v)表示用户u和v喜欢的产品集合。要避免值之间的差距太大，最简单的方式就是用log进行平滑，而避免数据异常，此处加一。

## 问题6：基于内容的推荐与协同过滤的区别

基于内容的推荐只考虑了对象的本身性质，将对象按标签形成集合，如果你消费集合中的一个则向你推荐集合中的其他对象；基于协同过滤的推荐算法，充分利用集体智慧，即在大量的人群的行为和数据中收集答案，以帮助我们对整个人群得到统计意义上的结论，推荐的个性化程度高。

# 协同过滤的召回作用

协同过滤能够很好地通过行为识别用户的喜好程度，借助相似度来按计算用户对未知商品喜好程度，以达到推荐的效果，然而可以看到，里面使用的信息非常有限，只有简单的用户喜好信息，而忽略了各种诸如地点、时间、用户性格等的多重信息，而产品的信息也被忽略，产品品牌、规则等，此时推荐内容非常受限，因此只是一种“粗排”，在比较大且成熟的推荐系统项目中，后续还需要经过精排和各种策略来实现精确排序，因此此处可以说协同过滤可以起到的是一个召回作用。

协同过滤的召回作用可以体现在下面几点：

* 协同过滤得到的内容更加有可解释性
* 协同过滤的得不到的内容，大概率是用户不喜欢的内容，可缩小精排范围
* 由于是粗排，则并不一定就要很高的准确性和实时性

因此协同过滤大部分情况下是一个召回作用。当然的，在一些初步形成的系统中，协同过滤也能有较好的效果，不容忽略。

# 小结

协同过滤根据用户的行为推断一个对其他物品的兴趣，完成推荐的过程，其主要目标是衡量“用户”和“物品”之间的匹配程度。主要有user cf和item cf经常使用，其共性都在上面进行了详细的阐述，其细节会在后面单独做一次阐述。

# 参考

<https://mp.weixin.qq.com/s/cNw8Ao1Z23Q12YGyZacJ1Q>

<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>