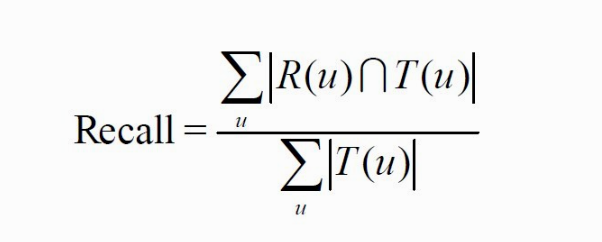
召回率

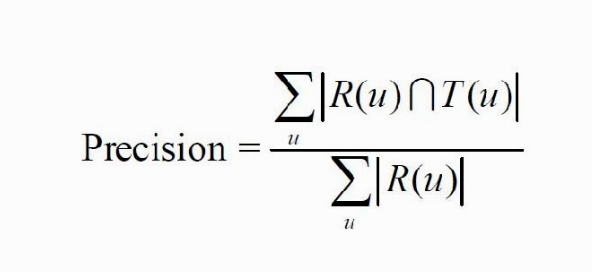
“召回率描述有多少比例的用户—物品评分记录包含在最终的推荐列表中，越大越好”



准确率

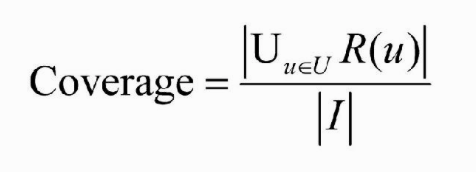
越大越好

“准确率描述最终的推荐列表中有多少比例是发生过的用户—物品评分记录”



覆盖率

“覆盖率表示最终的推荐列表中包含多大比例的物品，越大越好”

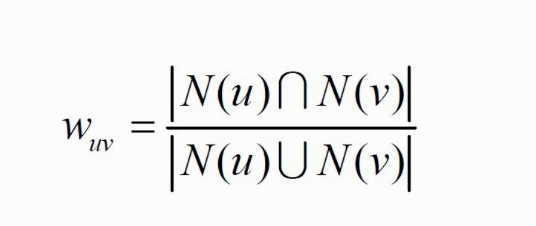


新颖度

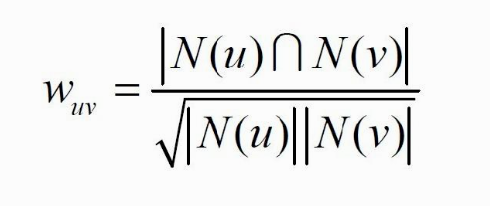
“如果推荐出的物品都很热门，说明推荐的新颖度较低，否则说明推荐结果比较新颖，越小越好”

“两个用户的兴趣相似度”

Jaccard公式:

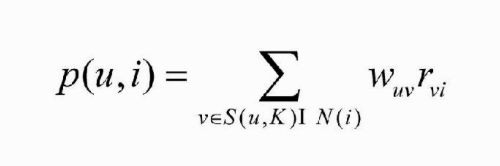


余弦相似度:



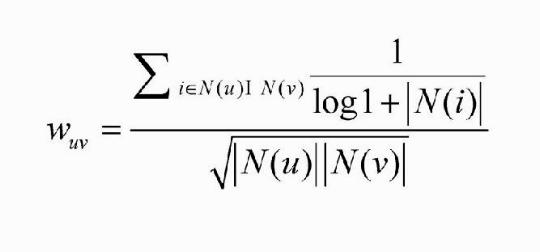
UserCF算法:

“适合用于新闻推荐”



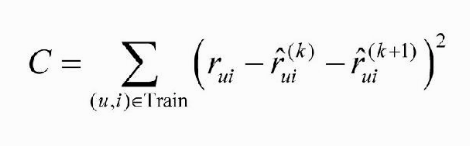
“其中，S (u, K)包含和用户u兴趣最接近的K个用户，N(i)是对物品i有过行为的用户集合，wuv是用户u和用户v的兴趣相似度，rvi代表用户v对物品i的兴趣，因为使用的是单一行为的隐反馈数据，所以所有的rvi=1。”

改进版



模型级联融合

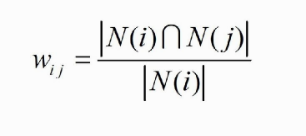
“假设已经有一个预测器，对于每个用户—物品对(u, i)都给出预测值，那么可以在这个预测器的基础上设计下一个预测器来最小化损失函数”



基于物品的协同过滤算法主要分为两步:

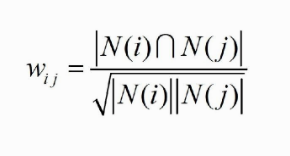
1. 计算物品之间的相似度。
2. 根据物品的相似度和用户的历史行为给用户生成推荐列表。

物品的相似度：



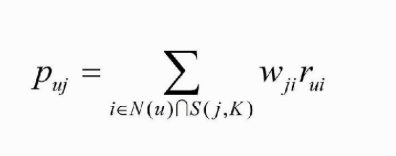
“这里，分母|N (i)|是喜欢物品i的用户数，而分子是同时喜欢物品i和物品j的用户数。”

为了避免推荐出热门的物品，可以用下面的公式：

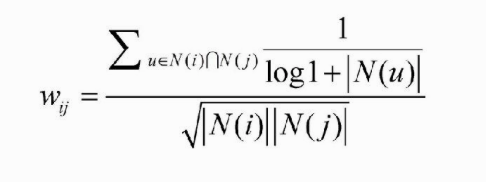


ItemCF算法:

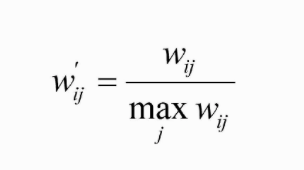
“这里N( u)是用户喜欢的物品的集合，S( i, K)是和物品i最相似的K个物品的集合，wji是物品j和i的相似度，rui是用户u对物品i的兴趣。对于隐反馈数据集，如果用户u对物品i有过行为，即可令rui=1。”



改进版ItemCF算法：

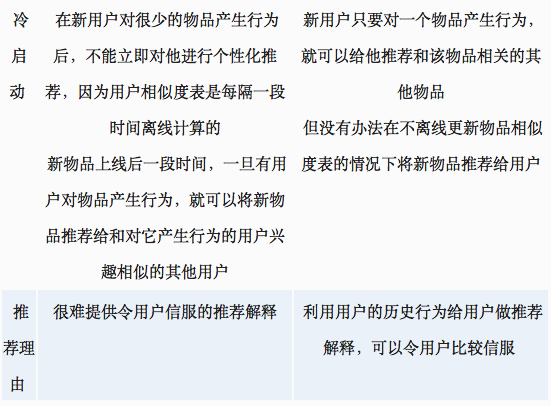


归一化相似度矩阵



UserCF和ItemCF优缺点的对比：





隐语义模型LFM

“通过隐含特征(latent factor)联系用户兴趣和物品。”

1.隐含语义分析技术的分类来自对用户行为的统计，代表了用户对物品分类的看法。

2.隐含语义分析技术允许我们指定最终有多少个分类，这个数字越大，分类的粒度就会越细，反正分类粒度就越粗。

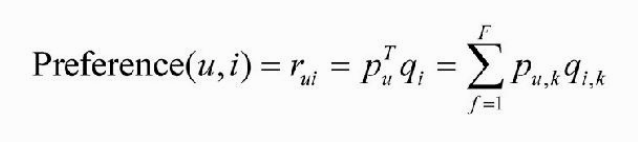
3.“隐含语义分析技术会计算出物品属于每个类的权重，因此每个物品都不是硬性地被分到某一个类中。”

4.“隐含语义分析技术给出的每个分类都不是同一个维度的，它是基于用户的共同兴趣计算出来的，如果用户的共同兴趣是某一个维度，那么LFM给出的类也是相同的维度。”

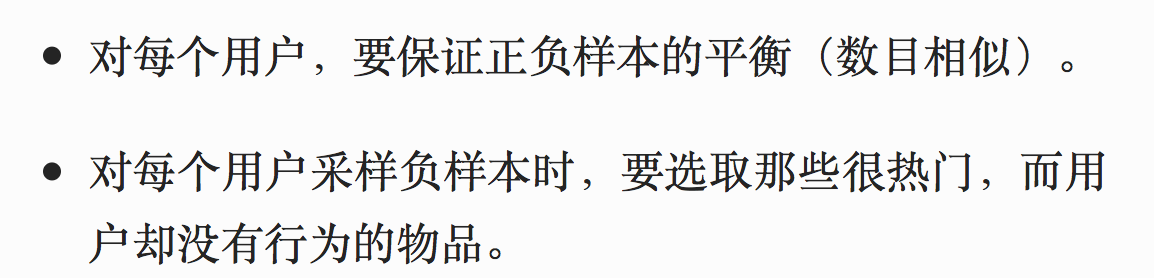
5.“隐含语义分析技术可以通过统计用户行为决定物品在每个类中的权重，如果喜欢某个类的用户都会喜欢某个物品，那么这个物品在这个类中的权重就可能比较高。”

LFM通过如下公式计算用户u对物品i的兴趣：

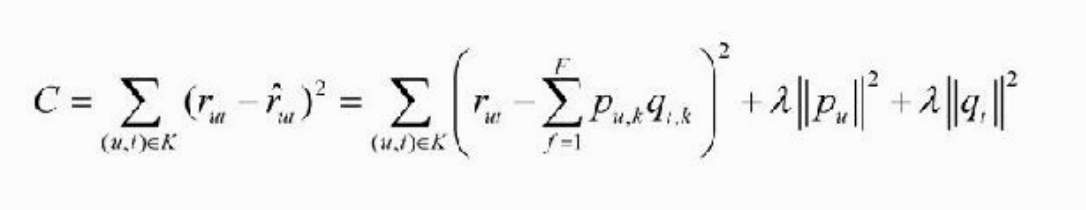
“这个公式中和是模型的参数，其中度量了用户u的兴趣和第k个隐类的关系，而度量了第k个隐类和物品i之间的关系。”

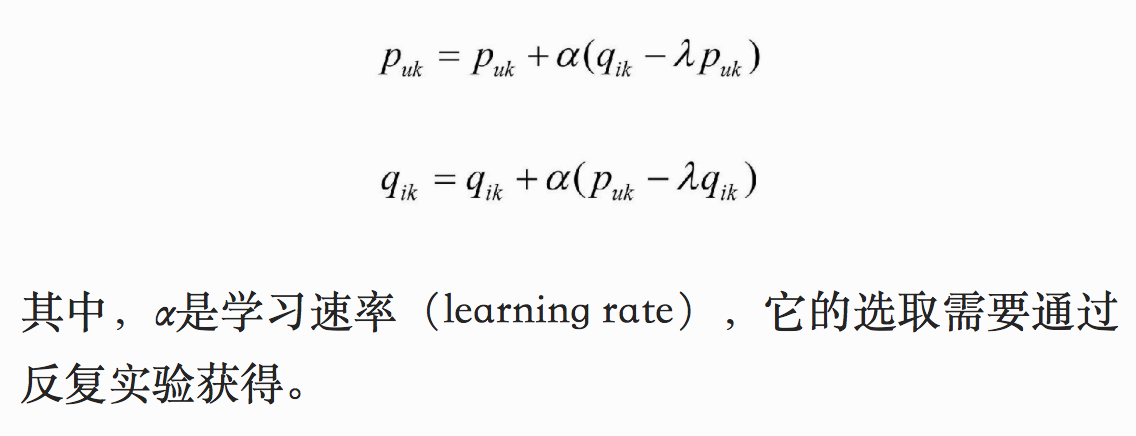


对负样本采样时应该遵循以下原则：

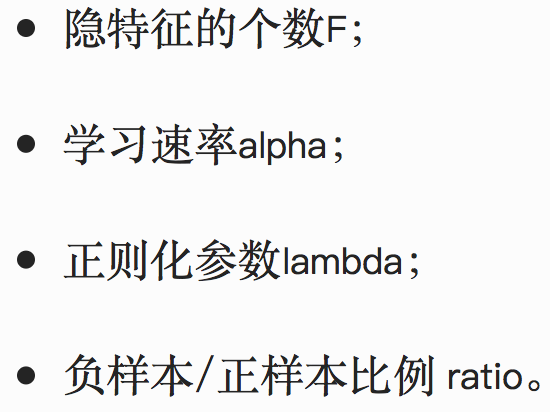


其中如果(u, i)是正样本，则有，否则有。然后，需要优化如下的损失函数来找到最合适的参数p和 q：



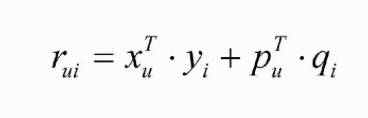


在LFM中，重要的参数有4个：



预测用户u是否会单击链接i：

其中，yi是根据物品的内容属性直接生成的，xuk是用户u对内容特征k的兴趣程度，用户向量xu可以根据历史行为记录获得，而且每天只需要计算一次。而pu、 qi是根据实时拿到的用户最近几小时的行为训练LFM获得的。



假设有M个用户和N个物品，K条用户对物品的行为记录

LFM离线计算的空间复杂度：

需要的存储空间是O(F\*(M+N))

假设是用户相关表，则需要O(M\*M)的空间，而对于物品相关表，则需要O(N\*N)的空间。

LFM离线计算的时间复杂度：

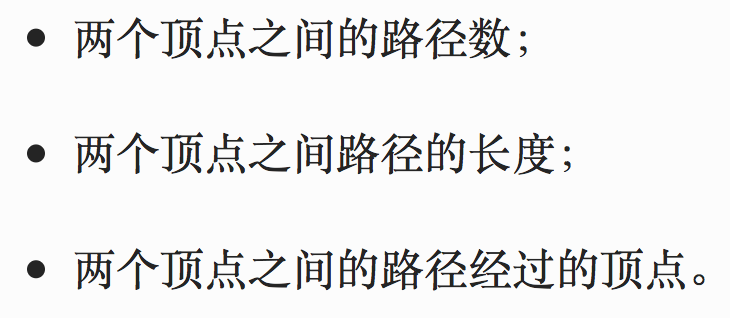
计算复杂度是O(K\*F\*S)

UserCF计算用户相关表的时间复杂度是O(N\*(K/N)^2)，而ItemCF计算物品相关表的时间复杂度是O(M\*( K/M)^2)

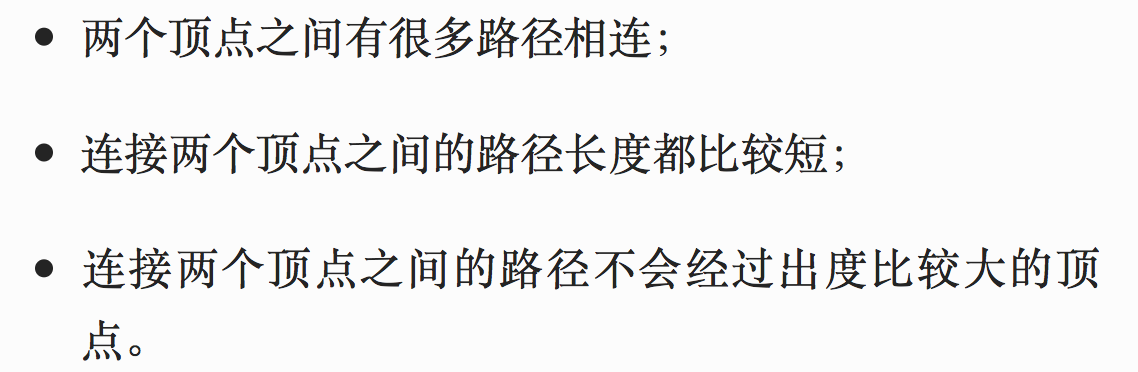
在一般情况下，LFM的时间复杂度要稍微高于UserCF和ItemCF

基于图的推荐算法

“一般来说图中顶点的相关性主要取决于下面3个因素：”

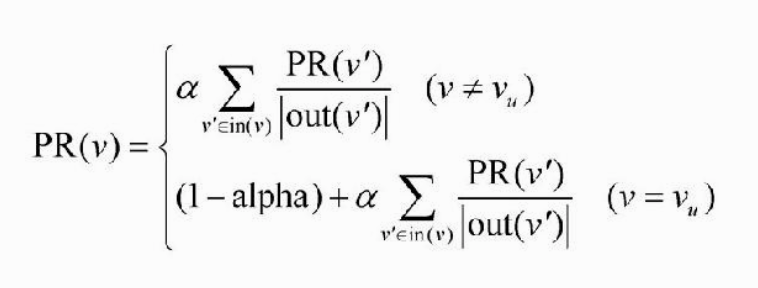


“而相关性高的一对顶点一般具有如下特征：”



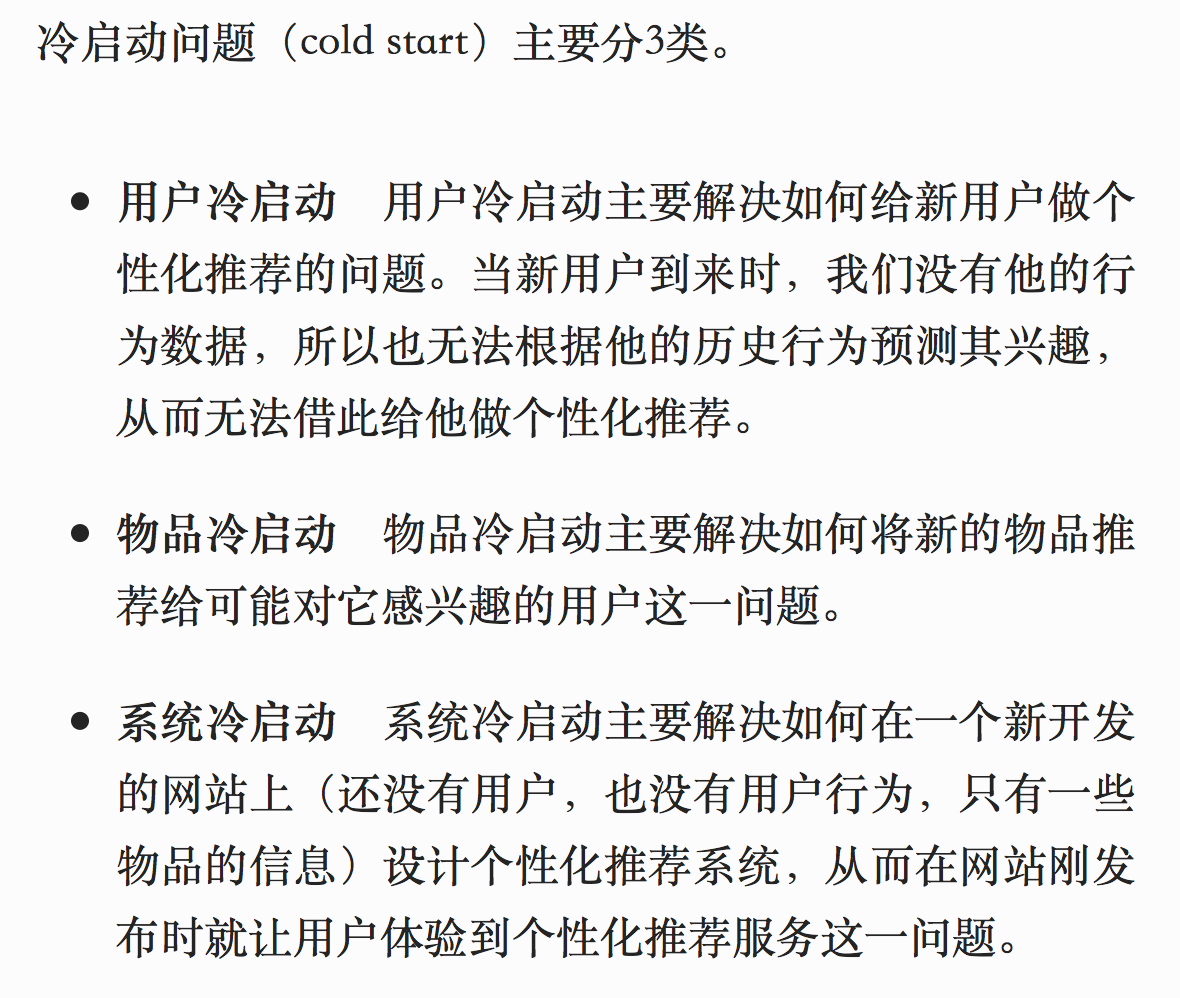
基于随机游走的PersonalRank算法

假设要给用户u进行个性化推荐，可以从用户u对应的节点vu开始在用户物品二分图上进行随机游走。游走到任何一个节点时，首先按照概率α决定是继续游走，还是停止这次游走并从vu节点开始重新游走。如果决定继续游走，那么就从当前节点指向的节点中按照均匀分布随机选择一个节点作为游走下次经过的节点。这样，经过很多次随机游走后，每个物品节点被访问到的概率会收敛到一个数。最终的推荐列表中物品的权重就是物品节点的访问概率。

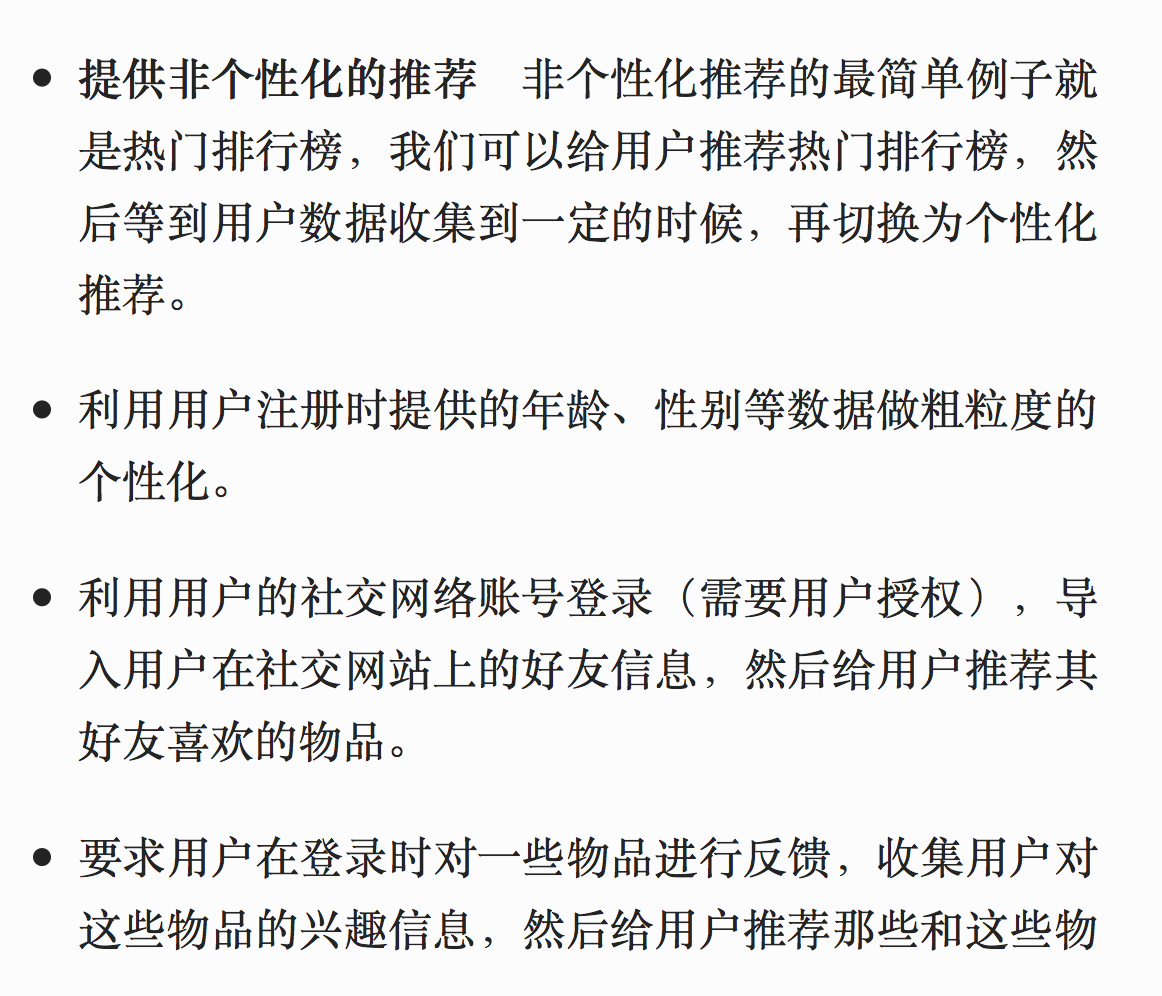


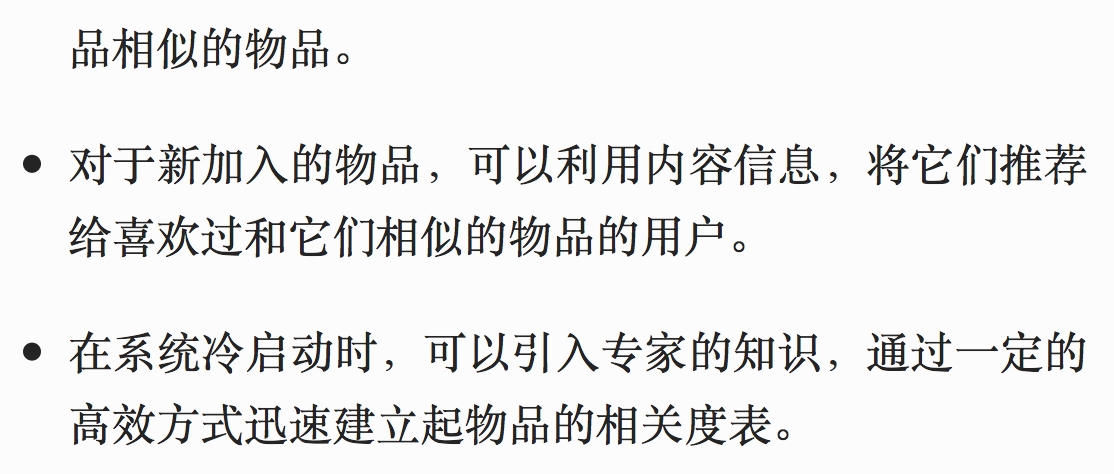
冷启动问题：

很多在开始阶段就希望有个性化推荐应用的网站来说，如何在没有大量用户数据的情况下设计个性化推荐系统并且让用户对推荐结果满意从而愿意使用推荐系统，就是冷启动的问题。



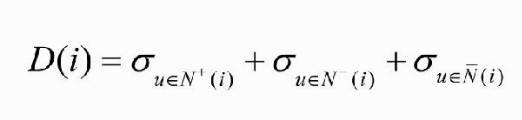
一般来说，可以参考如下解决方案。





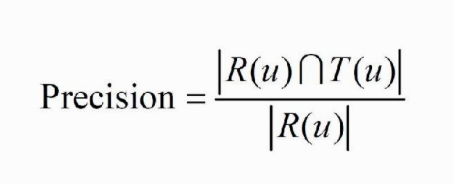
度量一个物品的区分度D (i):

其中，是喜欢物品i的用户集合，是不喜欢物品i的用户集合，是没有对物品i评分的用户集合。是喜欢物品i的用户对其他物品评分的方差，是不喜欢物品i的用户对其他物品评分的方差，是没有对物品i评分的用户对其他物品评分的方差。

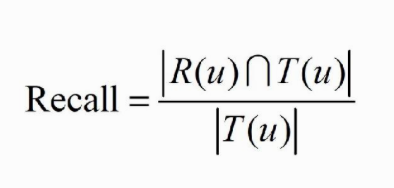


标签：

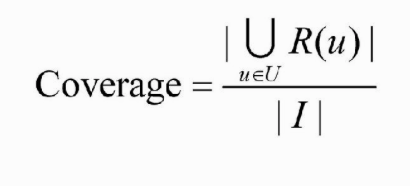
准确率：



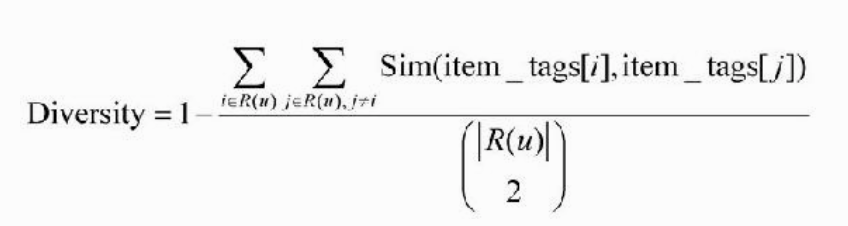
召回率：



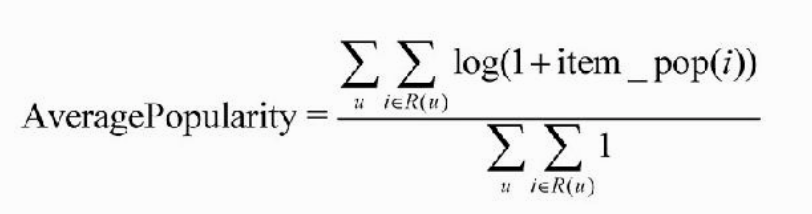
覆盖率：



多样性：

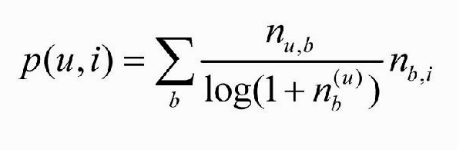


平均热门度:



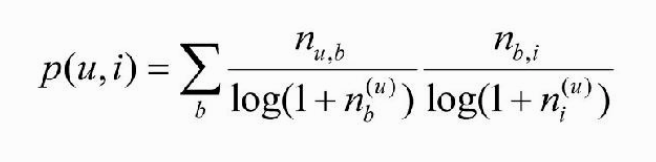
TagBasedTFIDF算法：

这里，记录了标签b被多少个不同的用户使用过。



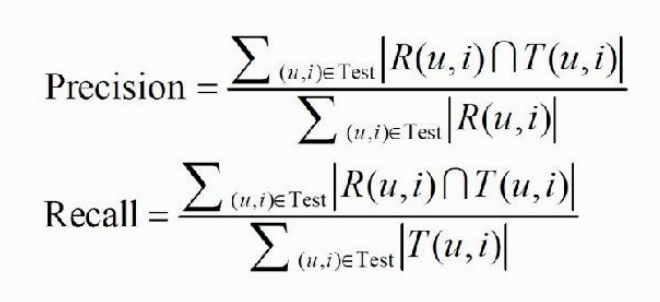
TagBasedTFIDF++算法：

其中，记录了物品i被多少个不同的用户打过标签。



对某一件物品打标签

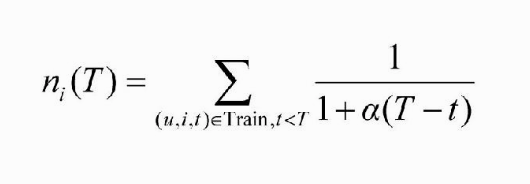
对于测试集中的每一个用户物品对(u, i)，我们都会推荐N个标签给用户u作参考。令R( u, i)为我们给用户u推荐的应该在物品i上打的标签集合，令T(u, i)为用户u实际给物品i打的标签的集合



时间上下文推荐算法

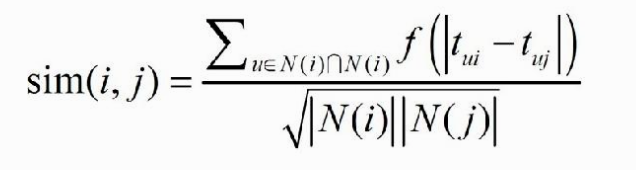
给定时间T，物品i最近的流行度可以定义为：

这里，α是时间衰减参数。

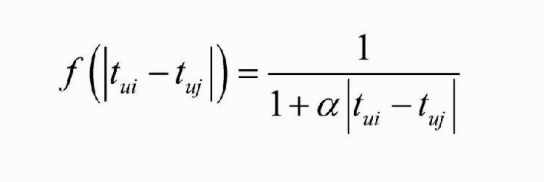


时间上下文相关的ItemCF算法

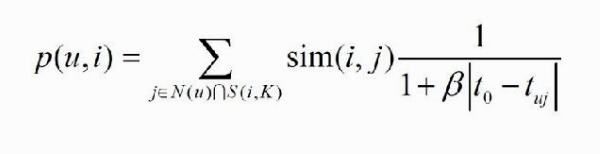
上面的公式在分子中引入了和时间有关的衰减项，其中是用户u对物品i产生行为的时间。f函数的含义是，用户对物品i和物品j产生行为的时间越远，则越小。



α是时间衰减参数，它的取值在不同系统中不同。



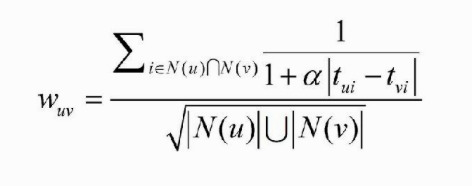
其中，是当前时间。上面的公式表明，越靠近，和物品j相似的物品就会在用户u的推荐列表中获得越高的排名。是时间衰减参数，需要根据不同的数据集选择合适的值。



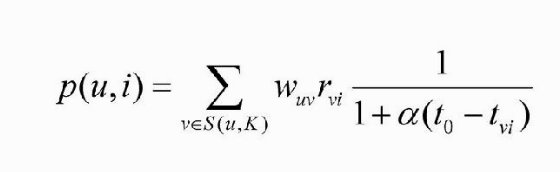
时间上下文相关的UserCF算法

计算用户u和用户v的兴趣相似度：

上面公式的分子对于用户u和用户v共同喜欢的物品i增加了一个时间衰减因子。用户u和用户v对物品i产生行为的时间越远，那么这两个用户的兴趣相似度就会越小。



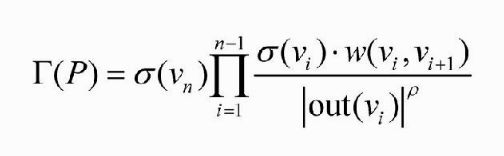
UserCF



路径融合算法

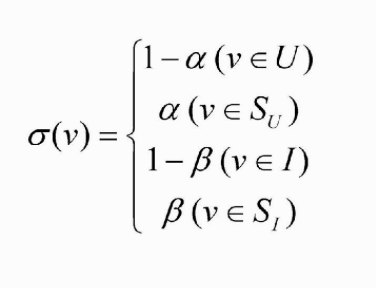
假设是连接顶点和的一条路径，这条路径的权重取决于这条路径经过的所有顶点和边：

这里out(v)是顶点v指向的顶点集合，|out(v)|是顶点v的出度，定义了顶点的权重，定义了边的权重。上面的定义符合上面3条原则的后两条。首先，因为，所以路径越长n越大，就越小。同时，如果路径经过了出度大的顶点v '，那么因为|out(v')|比较大，所以也会比较小。



对于时间段图模型，所有边的权重都定义为1，而顶点的权重定义如下：

这里，是两个参数，控制了不同顶点的权重。



基于位置的推荐算法

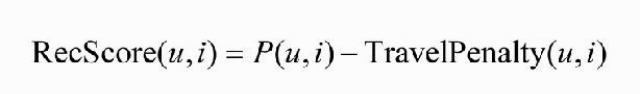
1. （用户， 用户位置， 物品， 评分）

对于第一种数据集，LARS的基本思想是将数据集根据用户的位置划分成很多子集。因为位置信息是一个树状结构，比如国家、省、市、县的结构。因此，数据集也会划分成一个树状结构。然后，给定每一个用户的位置，我们可以将他分配到某一个叶子节点中，而该叶子节点包含了所有和他同一个位置的用户的行为数据集。然后，LARS就利用这个叶子节点上的用户行为数据，通过ItemCF给用户进行推荐。

1. （用户， 物品， 物品位置， 评分）

“对于第二种数据集，每条用户行为表示为四元组（用户、物品、物品位置、评分），表示了用户对某个位置的物品给了某种评分。对于这种数据集，LARS会首先忽略物品的位置信息，利用ItemCF算法计算用户u对物品i的兴趣P( u, i)，但最终物品i在用户u的推荐列表中的权重定义为：

在该公式中，TravelPenalty(u,i)表示了物品i的位置对用户u的代价。计算Travel- Penalty(u,i)的基本思想是对于物品i与用户u之前评分的所有物品的位置计算距离的平均值（或者最小值）。

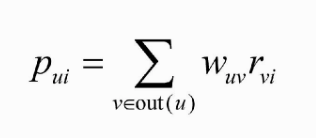


1. （用户， 用户位置， 物品， 物品位置， 评分）

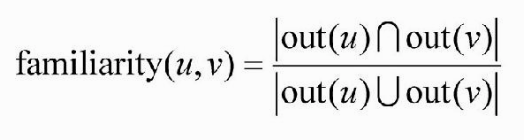
基于邻域的社会化推荐算法

用户u对物品i的兴趣pui

这里，由两部分相似度构成，一部分是用户u和用户v的熟悉程度，另一部分是用户u和用户v的兴趣相似度

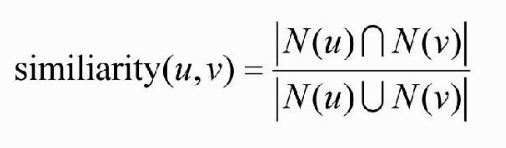


熟悉度可以用用户之间的共同好友比例来度量



兴趣相似度（similarity）

其中N( u)是用户u喜欢的物品集合。



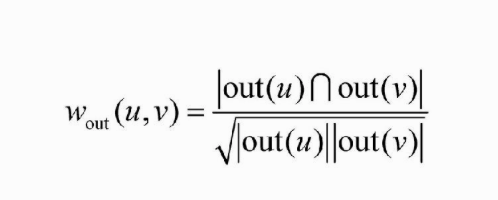
基于图的社会化推荐算法

好友推荐算法

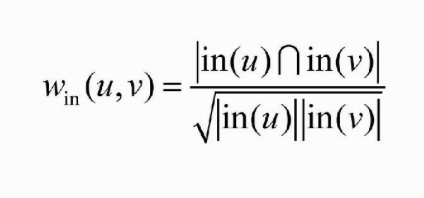
最简单的好友推荐算法是给用户推荐好友的好友。

对于用户u和用户v，我们可以用共同好友比例计算他们的相似度：

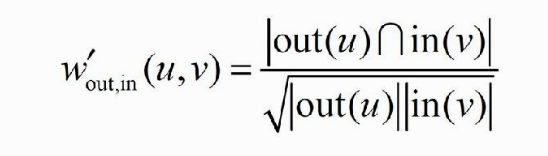
第一种：



第二种：

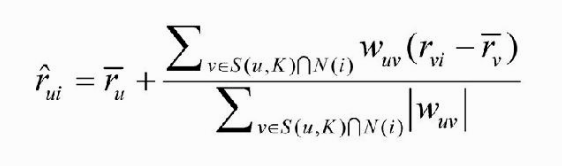


第三种：

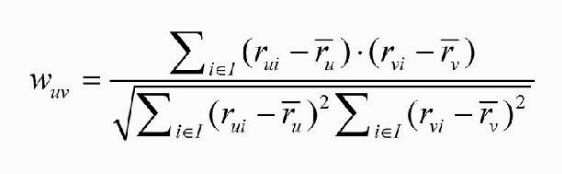


基于邻域的方法

预测一个用户对一个物品的评分

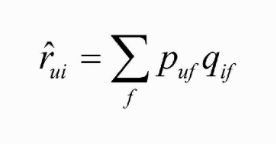


这里，S( u, K)是和用户u兴趣最相似的K个用户的集合，N( i)是对物品i评过分的用户集合，是用户v对物品i的评分，是用户v对他评过分的所有物品评分的平均值。

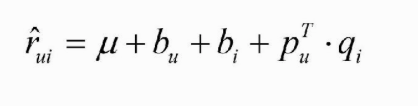


LFM算法的逐步优化

1. 无优化版



1. 偏置优化版



1. 考虑邻域的偏置优化版

