《推荐系统实践》学习笔记系列(五)

标签（空格分隔）： 推荐系统

5.1 时间上下文信息

5.1.1 时间效应简介

- 用户的兴趣是变化的
- 物品也是有生命周期的
- 季节效应

5.1.3 系统时间特性的分析

1. 时间特性的体现
    - 数据集每天独立用户数的增长情况
    - 系统的物品变化情况
    - 用户访问情况（用户访问频率）

2. 物品的生存周期和系统的时效性

- 物品平均在线天数
- 相隔T天系统物品流行度向量的平均相似度

5.1.4 推荐系统的实时性

实现推荐系统的实时性除了对用户行为的存取有实时性要求，还要求推荐算法本身具有实时性，而推荐算法本身的实时性意味着：

- 实时推荐系统不能每天都给所有用户离线计算推荐结果，然后在线展示昨天计算出来的结果。所以，要求在每个用户访问推荐系统时，都根据用户这个时间点前的行为实时计算推荐列表。
- 推荐算法需要平衡考虑用户的近期行为和长期行为，即要让推荐列表反应出用户近期行为所体现的兴趣变化，又不能让推荐列表完全受用户近期行为的影响，要保证推荐列表对用户兴趣预测的延续性。

5.1.5 推荐算法的时间多样性

推荐系统每天推荐结果的变化程度被定义为推荐系统的时间多样性。时间多样性高的推荐系统中用户会经常看到不同的推荐结果。

- 要求
  （1）需要保证推荐系统能够在用户有了新的行为后及时调整推荐结果，使推荐结果满足用户最近的兴趣。
  （2）需要保证推荐系统在用户没有新的行为时也能够经常变化一下结果，具有一定的时间多样性。
- 解决方法
  ♦在生成的推荐结果中加入一定的随机性。
  ♦记录用户每天看到的推荐结果，然后在每天给用户进行推荐时，对他前几天看到过很多次的推荐结果进行适当地降权。
  ♦每天给用户使用不同的推荐算法。

5.1.6 时间上下文推荐算法

i. 最近最热门
在没有时间信息的数据集中，我们可以给用户推荐历史上最热门的物品。那么在获得用户行为的时间信息后，最简单的非个性化推荐算法就是给用户推荐最近最热门的物品了。给定时间T，物品i最近的流行度( ) i n T 可以定义为：

$$n_i(T) = \sum_{(u,i,t)\epsilon Train, t<T} \frac{1}{1+\alpha(T-t)}$$

这里，α是时间衰减参数。
ii. 时间上下文相关的ItemCF算法

改善两方面,分别是物品相似度和在线推荐。
在再引入和时间有关的衰减项之前的物品相似度计算公式是：

$$sim(i,j) = \frac{\sum_{u \epsilon N(i) \bigcap N(j)}}{\sqrt{|N(i)||N(j)|}}$$

而在给用户u做推荐时，用户u对物品i的兴趣p(u,i)通过如下公式计算：

$$p(u,i) = \sum_{j \epsilon N(u)} sim(i,j)$$

引入和时间有关的衰减项$f(|t_{ui} - t_{uj}|)$之后:

$$sim(i,j) = \frac{\sum_{u \epsilon N(i) \bigcap N(j)} f(|t_{ui} - t_{uj}|)}{\sqrt{|N(i)||N(j)|}}$$

有很多数学衰减函数，这里使用如下衰减函数：

$$f(|t_{ui} - t_{uj}|) = \frac{1}{1 + \alpha |t_{ui} - t_{uj}|}$$

α是时间衰减参数，它的取值在不同系统中不同。如果一个系统用户兴趣变化很快，就应该取比较大的α，反之需要取比较小的α。
除了考虑时间信息对相关表的影响，我们也应该考虑时间信息对预测公式的影响。一般来说，用户现在的行为应该和用户最近的行为关系更大。因此，我们可以通过如下方式修正预测公式：

$$p(u,i) = \sum_{j \epsilon N(u)} sim(i,j) \frac{1}{1 + \beta |t_0 - t_{uj}|}$$

其中，t_0是当前时间。上面的公式表明，t_{uj}越靠近t_0，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。β是时间衰减参数，需要根据不同的数据集选择合适的值。
iii. 时间上下文相关的UserCF算法
改善两方面，分别是用户相似度和相似兴趣用户的最近行为。

$$w_{uv} = \frac{\sum_{i \epsilon N(u) \bigcap N(v)} \frac{1}{1 + \alpha |t_{ui} - t_{vi}|}}{\sqrt{|N(u)| \bigcup |N(v)|}}$$

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。
在得到用户相似度后，UserCF通过如下公式预测用户对物品的兴趣：

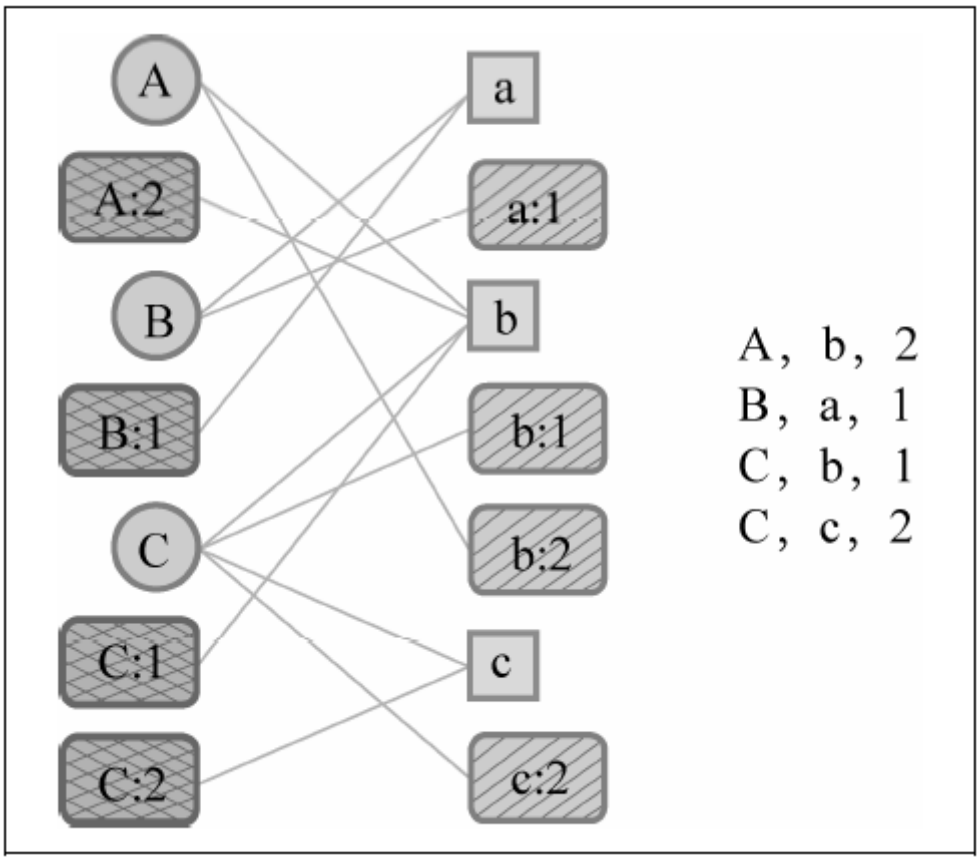
$$p(u,i) = \sum_{w \epsilon S(u,K)} w_{uv} r_{vi}$$

其中，$S(u,K)$包含了和用户u兴趣最接近的K个用户。如果用户v对物品i产生过行为，那么$r_{vi} = 1$，否则$r_{vi} = 0$。
如果考虑和用户u兴趣相似用户的最近兴趣，我们可以设计如下公式：

$$p(u,i) = \sum_{w \epsilon S(u,K)} w_{uv} r_{vi} \frac{1}{1 + \alpha |t_0 - t_{vi}|}$$

5.1.7 时间段图模型

时间段图模型$G(U, S_U, I, S_I, E, w, \sigma)$是一个二分图。
U是用户节点集合，S_U是用户时间段节点集合。一个用户时间段节点$v_{ut} \epsilon S_U$会和用户u在时刻t喜欢的物品相连。
I是物品集合，S_I是物品时间段节点集合。一个物品时间段节点$v_{it} \epsilon S_I$会和所有在时刻t喜欢物品i的用户通过边相连。
E是边集合，它包含了3种边：（1）如果用户u对物品i有行为，那么存在边$e_{v_u, v_i} \epsilon E$;(2)如果用户u在t时刻对物品i有行为，那么存在两条边$e(v_{ut}, v_i), e(v_u, v_{it}) \epsilon E$。$w(e)$定义了边的权重,$\sigma(e)$定义了顶点的权重。



定义完图的结构后，最简单的想法是可以利用前面提到的PersonalRank算法给用户进行个性化推荐。但是因为这个算法需要在全图上进行迭代计算，所以时间复杂度比较高。因此我们提出了一种称为路径融合算法的方法，通过该算法来度量图上两个顶点的相关性。
一般来说，图上两个相关性比较高的顶点一般具有如下特征：
两个顶点之间有很多路径相连；
两个顶点之间的路径比较短；
两个顶点之间的路径不经过出度比较大的顶点
从这3条原则出发，路径融合算法首先提取出两个顶点之间长度小于一个阈值的所有路径
然后根据每条路径经过的顶点给每条路径赋予一定的权重，最后将两个顶点之间所有路径的权重之和作为两个顶点的相关度。

假设 $P=\{v_1,v_2,\cdots,v_n\}$ 是连接顶点 v_1 和 v_n 的一条路径，这条路径的权重 $\Gamma(P)$ 取决于这条路径经过的所有顶点和边：

$$\Gamma(P)=\sigma(v_n)\prod_{i=1}^{n-1}\frac{\sigma(v_i)\cdot w(v_i,v_{i+1})}{|\text{out}(v_i)|^{\rho}}$$

这里out(v)是顶点v指向的顶点集合，|out(v)|是顶点v的出度，$\sigma(v_i)\in(0,1]$ 定义了顶点的权重，$w(v_i,v_{i+1})\in(0,1]$ 定义了边 $e(v_i,v_{i+1})$ 的权重。上面的定义符合上面3条原则的后两条。首先，因为 $\frac{\sigma(v_i)\cdot w(v_i,v_{i+1})}{|\text{out}(v_i)|^{\rho}}\in(0,1)$，所以路径越长$n$越大，$\Gamma(P)$ 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 $\Gamma(P)$ 也会比较小。

在定义了一条路径的权重后，就可以定义顶点之间的相关度。对于顶点v和v′，令 $p(v,v',K)$ 为这两个顶点间距离小于K的所有路径，那么这两个顶点之间的相关度可以定义为：

$$d(v,v')=\sum_{P\in P(v,v',K)}\Gamma(P)$$

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 $\sigma(v)$ 定义如下：

$$\sigma(v)=\begin{cases}1-\alpha\,(v\in U)\\ \alpha\,(v\in S_U)\\ 1-\beta\,(v\in I)\\ \beta\,(v\in S_I)\end{cases}$$

这里，$\alpha,\beta\in[0,1]$ 是两个参数，控制了不同顶点的权重。

5.2 地点上下文信息

基于位置的推荐算法

1. 数据集有三种不同的形式

- （用户，用户位置，物品，评分），每一条记录代表了某一个地点的用户对物品的评分。它们使用的是MovieLens数据集。该数据集给出了用户的邮编，从而可以知道用户的大致地址。
- （用户，物品，物品位置，评分），每一条记录代表了用户对某个地方的物品的评分。LARS使用了FourSquare的数据集，该数据集包含用户对不同地方的餐馆、景点、商店的评分。
- （用户，用户位置，物品，物品位置，评分），每一条记录代表了某个位置的用户对某个位置的物品的评分。

2. 发现了用户兴趣和地点相关的两种特征：

- 兴趣本地化：不同地方的用户兴趣存在着很大的差别。
- 活动本地化：一个用户往往在附近的地区活动。

3. 针对三种数据集的方法

i. 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。不过这样做的缺点是，每个叶子节点上的用户数量可能很少，因此他们的行为数据可能过于稀疏，从而无法训练出一个好的推荐算法。为此，我们可以从根节点出发，在到叶子节点的过程中，利用每个中间节点上的数据训练出一个推荐模型，然后给用户生成推荐列表。而最终的推荐结果是这一系列推荐列表的加权。
ii. 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

$$RecScore(u,i) = P(u,i) - TravelPenalty(u,i)$$

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$TravelPenalty(u,i)$
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）。
iii. 从第三种数据集的定义可以看到，它相对于第二种数据集增加了用户当前位置这一信息。而在给定了这一信息后，我们应该保证推荐的物品应该距离用户当前位置比较近，在此基础上再通过用户的历史行为给用户推荐离他近且他会感兴趣的物品。

作者[钱昊达]
2018年8月26日