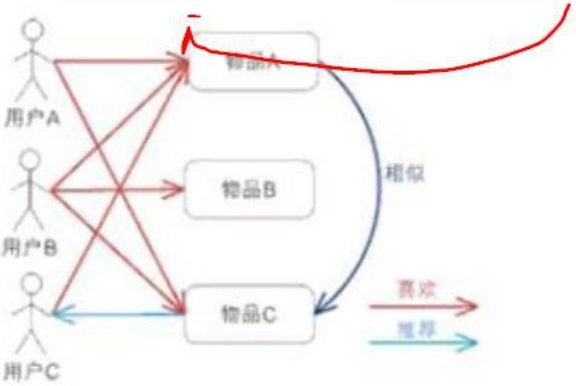
第三课

协同过滤及实现

基于物品的协同过滤推荐技术(评分)

用户/物品	物品A	物品B	物品C
用户A	√		√
用户B	√	√	√
用户C	√		推荐



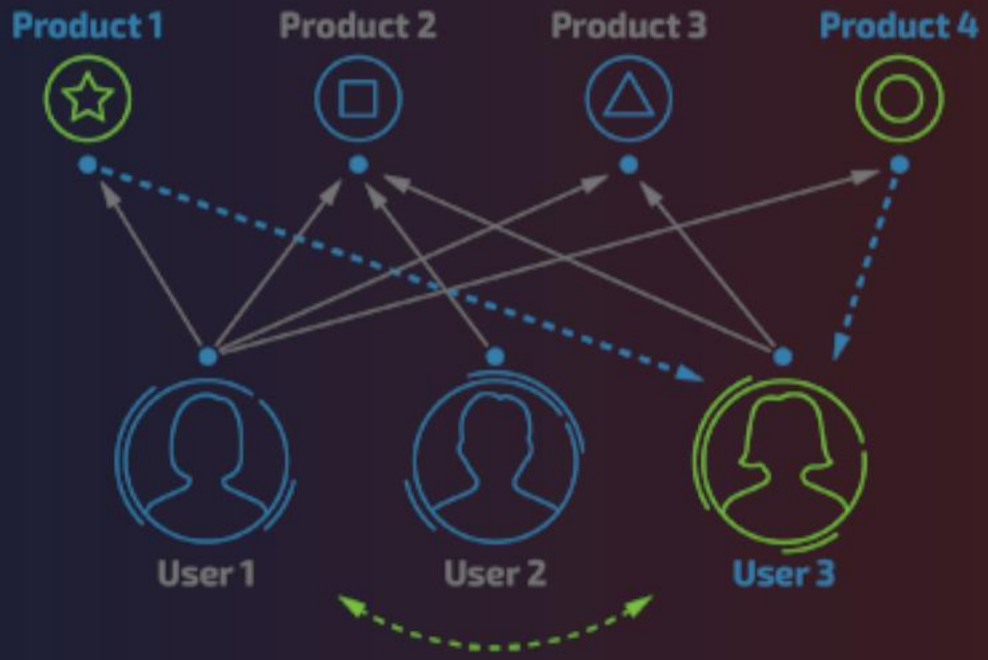
基于物品的协同过滤算法(Item_CF)

基于物品的协同过滤算法(item based collaborative filtering)，简称 Item CF，是目前业界应用最广的算法。该算法给用户推荐那些和他们之前喜欢的物品相似的物品。比如，该算法会因为你购买过《mahout 实战》而给你推荐《机器学习实战》。物品 A 和物品 B 具有很大的相似度是因为喜欢物品 A 的用户大都也喜欢物品 B。

基于用户的协同过滤算法（user-based collaboratIve filtering）

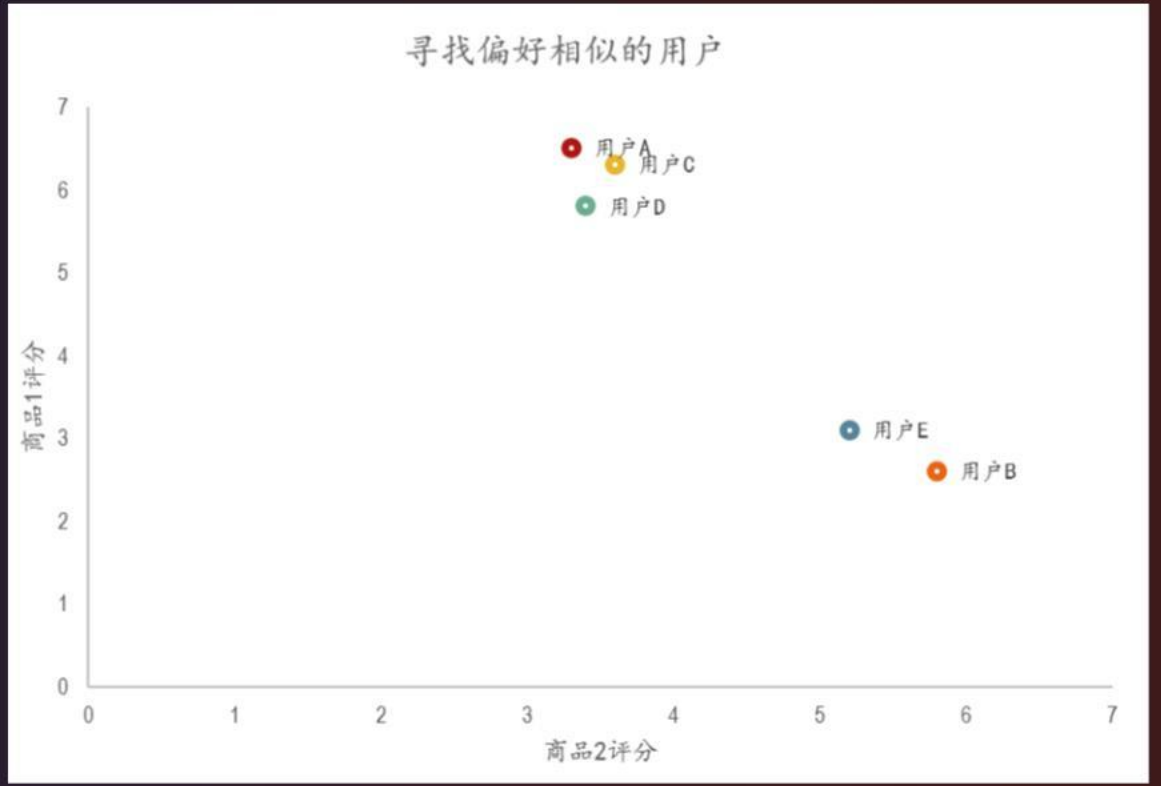
基于用户的协同过滤算法是通过用户的历史行为数据发现用户对商品或内容的喜欢（如商品购买，收藏，内容评论或分享），并对这些喜好进行度量和打分。根据不同用户对相同商品或内容的态度和偏好程度计算用户之间的关系。在有相同喜好的用户间进行商品推荐。简单的说就是如果A,B两个用户都购买了x,y,z三本图书，并且给出了5星的好评。那么A和B就属于同一类用户。可以将A看过的图书w也推荐给用户B。

UCF



UCF

	商品1	商品2
用户A	3.3	6.5
用户B	5.8	2.6
用户C	3.6	6.3
用户D	3.4	5.8
用户E	5.2	3.1



UCF

$$d(x,y) := \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\cdots+(x_n-y_n)^2} = \sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

欧几里德距离评价

	系数	倒数
用户A&B	4.63	0.18
用户A&C	0.36	0.73
用户A&D	0.71	0.59
用户A&E	3.89	0.20
用户B&C	4.30	0.19
用户B&D	4.00	0.20
用户B&E	0.78	0.56
用户C&D	0.54	0.65
用户C&E	3.58	0.22
用户D&E	3.24	0.24

皮尔逊相关度评价

皮尔逊相关度评价是另一种计算用户间关系的方法。他比欧几里德距离评价的计算要复杂一些，但对于评分数据不规范时皮尔逊相关度评价能够给出更好的结果。以下是一个多用户对多个商品进行评分的示例。这个示例比之前的两个商品的情况要复杂一些，但也更接近真实的情况。我们通过皮尔逊相关度评价对用户进行分组，并推荐商品。

	商品1	商品2	商品3	商品4	商品5
用户A	3.3	6.5	2.8	3.4	5.5
用户B	3.5	5.8	3.1	3.6	5.1
用户C	5.6	3.3	4.5	5.2	3.2
用户D	5.4	2.8	4.1	4.9	2.8
用户E	5.2	3.1	4.7	5.3	3.1

皮尔逊相关系数

皮尔逊相关系数的计算公式如下，结果是一个在-1与1之间的系数。该系数用来说明两个用户间联系的强弱程度。

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y} = \frac{E(XY)-E(X)E(Y)}{\sqrt{E(X^2)-E^2(X)}\sqrt{E(Y^2)-E^2(Y)}}$$

UCF

- 0.8-1.0 极强相关
- 0.6-0.8 强相关
- 0.4-0.6 中等程度相关
- 0.2-0.4 弱相关
- 0.0-0.2 极弱相关或无相关

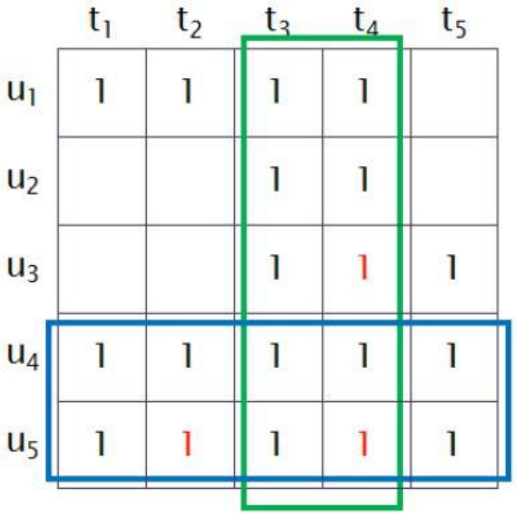
UCF

皮尔逊相关系数

	相似度
用户A&B	0.9998
用户A&C	-0.8478
用户A&D	-0.8418
用户A&E	-0.9152
用户B&C	-0.8417
用户B&D	-0.8353
用户B&E	-0.9100
用户C&D	0.9990
用户C&E	0.9763
用户D&E	0.9698

基本原理

协同过滤的基本原理



t_1：咕咚表
t_2：Hosee蓝牙耳机
t_3：运动手环
t_4：BIAZE ipad mini皮套

User-based CF

给浏览了“咕咚表”、“运动手环”的用户u_5推荐t2“Hosee蓝牙耳机”、t4“BIAZE ipad mini皮套”

Item-based CF

t3“运动手环”的浏览用户群体和t4“BIAZE ipad mini皮套”的用户群体相似，因此可以将向t3“运动手环”的用户u_3、u_5推荐t4“BIAZE ipad mini皮套”

ICF

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。

步骤

1、计算物品之间的相似度

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

其中，|N(i)|是喜欢物品 i 的用户数，|N(j)|是喜欢物品 j 的用户数，|N(i)&N(j)|是同时喜欢物品 i 和物品 j 的用户数。

计算物品之间相似度

用户 A 对物品 a、b、d 有过行为，用户 B 对物品 b、c、e 有过行为。

依此构建用户—物品倒排表：物品 a 被用户 A、E 有过行为，等等。

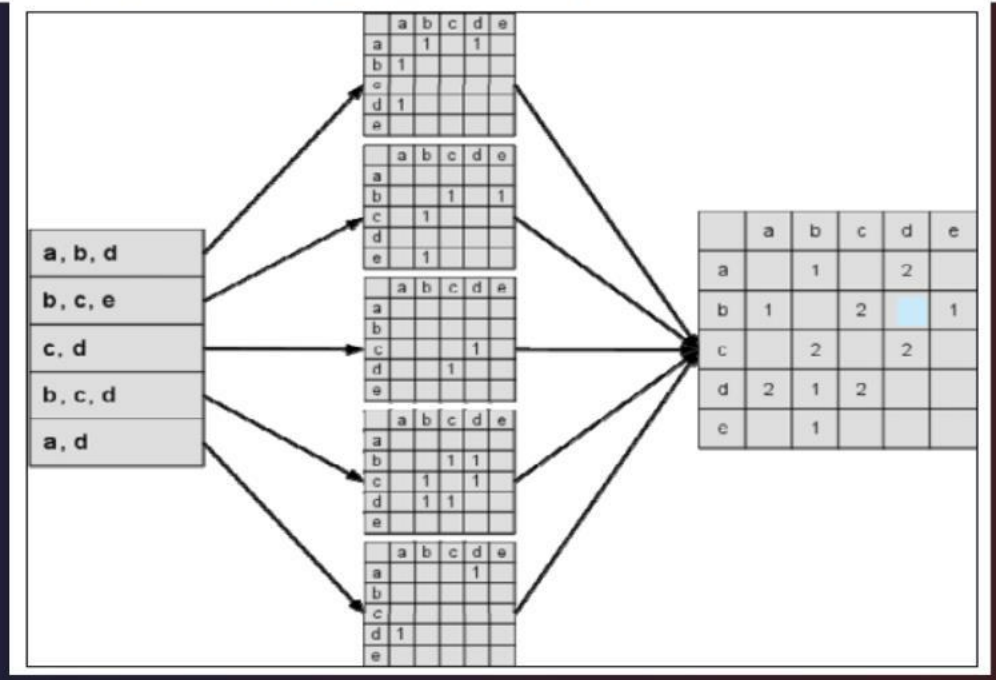
A	a	b	d
B	b	c	e
C	c	d	
D	b	c	d
E	a	d	

a	A	E		
b	A	B	D	
c	B	C	D	
d	A	C	D	E
e	B			

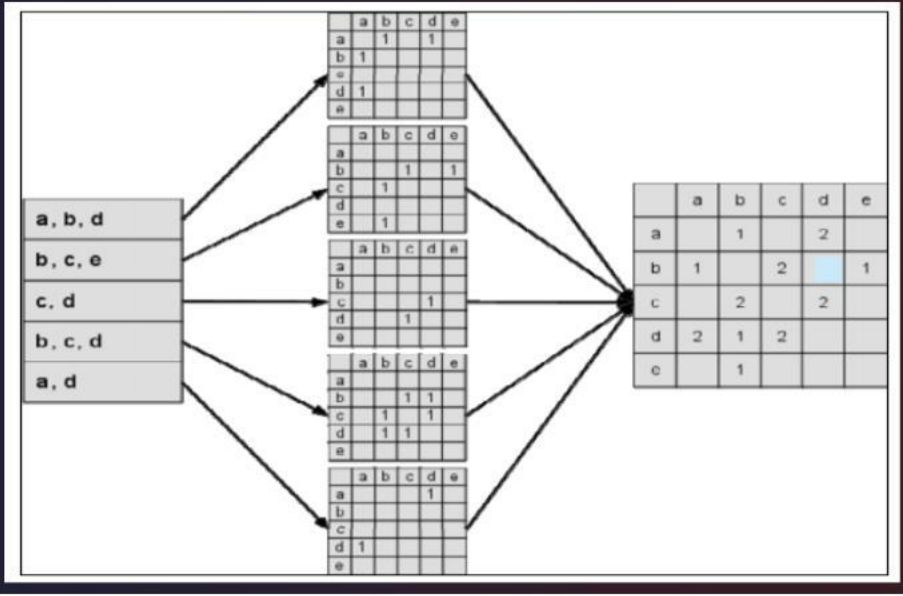
我们的目的是建立物品-物品间的同现矩阵，用于计算物品间的相似度，可以不建立上述倒排表，直接按如下方式。从每条数据中直接构建物品—物品同现矩阵如下。

计算物品之间相似度

我们的目的是建立物品-物品间的同现矩阵，用于计算物品间的相似度，可以不建立上述倒排表，直接按如下方式。从每条数据中直接构建物品—物品同现矩阵如下。建立物品相似度矩阵 C：



计算物品之间相似度



其中， $C[i][j]$ 记录了同时喜欢物品 i 和物品 j 的用户数。除以分母部分，这样我们就可以得到物品之间的相似度矩阵 $W[i][j]$。

相似度计算方式

2.1 欧几里得相似度

欧几里得相似度根据欧几里得距离计算而来，距离越近相似度越高，反之相反。

欧几里得距离公式

$$d_{X,Y} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

欧几里得相似度公式

$$sim(x, y) = \frac{1}{1 + d(x, y)}$$

相似度计算方式

2.2 皮尔逊相似度

皮尔逊相关系数，即概率论中的相关系数，取值范围【-1，+1】。当大于零时，两个变量正相关，当小于零时表示两个向量负相关。

计算公式为

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_x \sigma_y} = \frac{E((X-\mu_x)(Y-\mu_y))}{\sigma_x \sigma_y}$$

$$= \frac{E(XY)-E(X)E(Y)}{\sqrt{E(X^2)-E^2(X)}\ \sqrt{E(Y^2)-E^2(Y)}}$$

\end{equation}

相似度计算方式

2.4 Tanimoto 相似度

Tanimoto相似度也称为Jaccard系数，是Cosine相似度扩展，多用于文档相似度就算。此相似度不考虑评价值，只考虑两个集合共同个体数量。

公式为

$$sim(x,y) = \frac{X \cap Y}{\|X\| + \|Y\| - \|X \cap Y\|}$$

生成推荐列表

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。

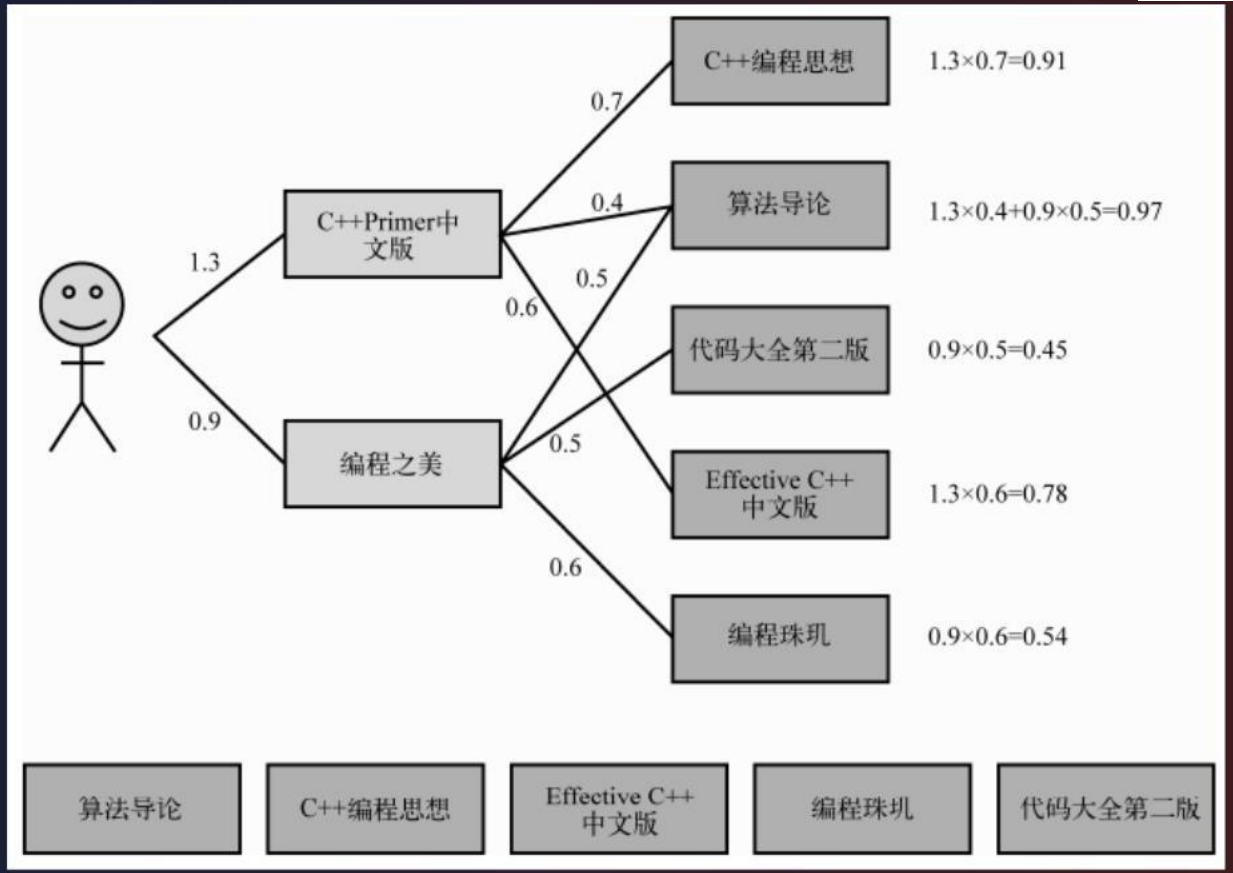
用户 u 对一个物品 j 的兴趣：

$$p(u,j)=\sum_{i\in N(u)\cap S(j,K)} w_{ji}r_{ui}$$

其中，$p(u,j)$ 表示用户 u 对物品 j 的兴趣，$N(u)$ 表示用户喜欢的物品集合（i 是该用户喜欢的某一个物品），$S(j,K)$ 表示和物品 j 最相似的 K 个物品集合（j 是这个集合中的某一个物品），w_{ji} 表示物品 j 和物品 i 的相似度，r_{ui} 表示用户 u 对物品 i 的兴趣（这里简化 r_{ui} 都等于 1）。

该公式的含义是：和用户历史上感兴趣的物品越相似的物品，越有可能在用户的推荐列表中获得比较高的排名。

生成推荐列表


C++编程思想
1.3×0.7=0.91
0.7
0.4
算法导论
1.3×0.4+0.9×0.5=0.97
C++Primer中文版
1.3
0.5
0.6
代码大全第二版
0.9×0.5=0.45
0.9
0.5
Effective C++中文版
1.3×0.6=0.78
0.6
编程珠玑
0.9×0.6=0.54
编程之美
算法导论
C++编程思想
Effective C++中文版
编程珠玑
代码大全第二版

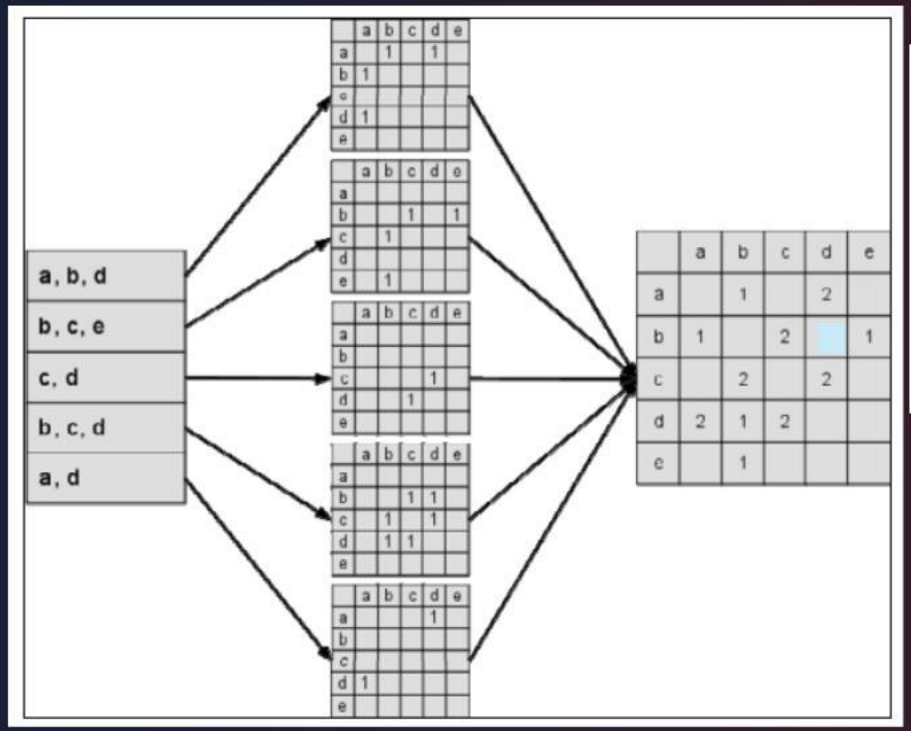
代码片段

```
#定义基于物品的协同过滤算法类
class ItemBasedCF:
    #初始化对象
    def __init__(self,train_file,test_file):
        #训练数据
        self.train_file = train_file
        #测试数据
        self.test_file = test_file
        #读取数据
        self.readData()

    #数据读取函数
    def readData(self):
        #读取文件，并生成用户-物品的评分表和测试集
        #用户-物品的评分表
        #训练集
        self.train = dict()
        #打开文件，按行读取训练数据
        for line in open(self.train_file):
            #获得用户、物品、评分数据，丢弃时间戳数据
            user,item,score,_ = line.strip().split("\t")
            #用户-物品评分矩阵
            self.train.setdefault(user,{})
            #分数赋值
            self.train[user][item] = int(score)

        #测试集
        self.test = dict()
        #打开文件，按行读取训练数据
        for line in open(self.test_file):
            #获得用户、物品、评分数据，丢弃时间戳数据
            user,item,score,_ = line.strip().split("\t")
            #用户-物品评分矩阵
            self.test.setdefault(user,{})
            #分数赋值
```

代码片段

```
#物品间相似度
def ItemSimilarity(self):
    #建立物品-物品的共现矩阵
    C = dict()  #物品-物品的共现矩阵
    N = dict()  #物品被多少个不同用户购买

    #遍历训练数据，获得用户对有过行为的物品
    for user,items in self.train.items():
        #遍历该用户每件物品项
        for i in items.keys():
            #该物品被用户购买计数加1
            N.setdefault(i,0)
            N[i] += 1
            #物品-物品的共现矩阵数据加1
            C.setdefault(i,{})
            #遍历该用户每件物品项
            for j in items.keys():
                #若该项为当前物品，跳过
                if i == j : continue
                #同一个用户下，其他物品项
                #遍历到其他不同物品则加1
                #初始化为0
                C[i].setdefault(j,0)
                #加1
                C[i][j] += 1
```

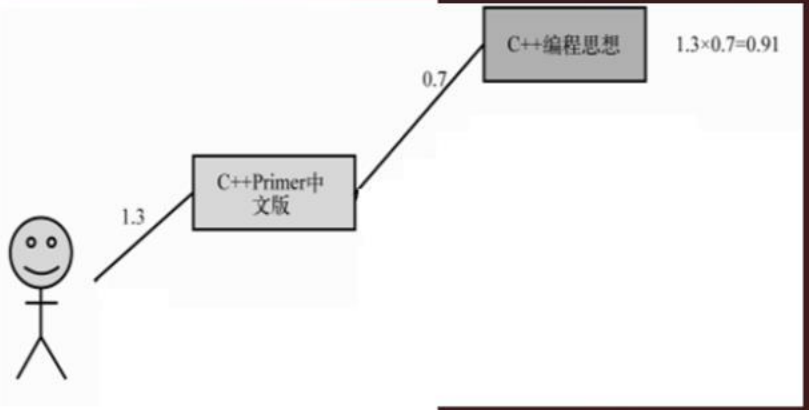
代码片段




```
        #计算相似度矩阵
        #计算物品-物品相似度，余弦相似度
        self.W = dict()
        #遍历物品-物品共现矩阵的所有项，
        #每行物品、该行下的其他物品
        for i,related_items in C.items():
            #存放物品间相似度
            self.W.setdefault(i,{})
            #遍历其他每一个物品及对应的同现矩阵的值，即分子部分
            for j,cij in related_items.items():
                #余弦相似度
                self.W[i][j] = cij / (math.sqrt(N[i] * N[j]))
        #返回物品相似度
        return self.W
```

代码片段

```
#给用户user推荐，前K个相关用户喜欢的，
#用户user未产生过行为的物品
#默认3个用户，推荐10个物品
def Recommend(self,user,K=3,N=10):
    #用户user对物品的偏好值
    rank = dict()
    #用户user产生过行为的物品项item和评分
    action_item = self.train[user]

    #找用户user产生过行为的物品item，
    #与物品item按相似度从大到小进行排列
    #取与物品item相似度最大的K个物品
    for item,score in action_item.items():
        #遍历与物品item最相似的前K个物品，获得这些物品及相似分数
        for j,wj in sorted(self.W[item].items(),key=lambda x:x[1],reverse=True)[0:K]:
            #若该物品当前物品，跳过
            if j in action_item.keys():
                continue
            #计算用户user对物品i的偏好值
            #初始化该值为0
            rank.setdefault(j,0)
            #通过与其相似物品对物品i的偏好值相乘并相加
            rank[j] += score * wj
    #按评分值大小，为用户user推荐结果的取前N个物品
    return dict(sorted(rank.items(),key=lambda x:x[1],reverse=True)[0:N])
```



运行

为用户ID为3的用户推荐其未关注的10部电影。

```
if __name__ == '__main__':
    cf = ItemBasedCF('u.data','u.data')
    cf.ItemSimilarity()
    print cf.Recommend('3')
```

结果

```
{'678': 2.876273579448667,
 '313': 5.911009727650591,
 '50': 3.531530583572753 3,
 '315': 5.623188132296642,
 '316': 3.819076860450629 8,
 '69': 2.741796413603276,
 '269': 5.026259738111738,
 '174': 2.964541078124643,
 '172': 2.939559408818695,
 '286': 4.154831912556865}
```

区别

	UserCF	ItemCF
性能	适用于用户较少的场合，如果用户很多，计算用户相似度矩阵代价很大	适用于物品数明显小于用户数的场合，如果物品很多（网页），计算物品相似度矩阵代价很大
领域	时效性较强，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的立即变化	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
	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和对它产生行为的用户兴趣相似的其他用户	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

偏好收集

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。

评价指标

为了评估推荐算法的好坏需要各方面的评估指标。召回率(Recall)和精度(Precision)是广泛用于信息检索和统计学分类领域的两个度量值，用来评价结果的质量。

召回率(Recall)：推荐给出且正确的结果/总的正确结果

精确率(Precision)：推荐给出且正确的结果/分类器给出的所有结果

覆盖率：覆盖率表示推荐给出的物品/物品全集空间的比例。

新颖度：新颖度是为了推荐长尾区间的物品。用推荐列表中物品的平均流行度度量推荐结果的新颖度。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。

评价指标

举例来说：一个数据库有 500 个文档，其中有 50 个文档符合定义的问题。系统检索到 75 个文档，但是只有 45 个符合定义的问题。

分类器给出且正确的结果为 45 个，总的正确结果 50 个，分类器给出的所有结果 75 个。

召回率

$$R = \frac{45}{50} = 90\%$$

精确率

$$P = \frac{45}{75} = 60\%$$