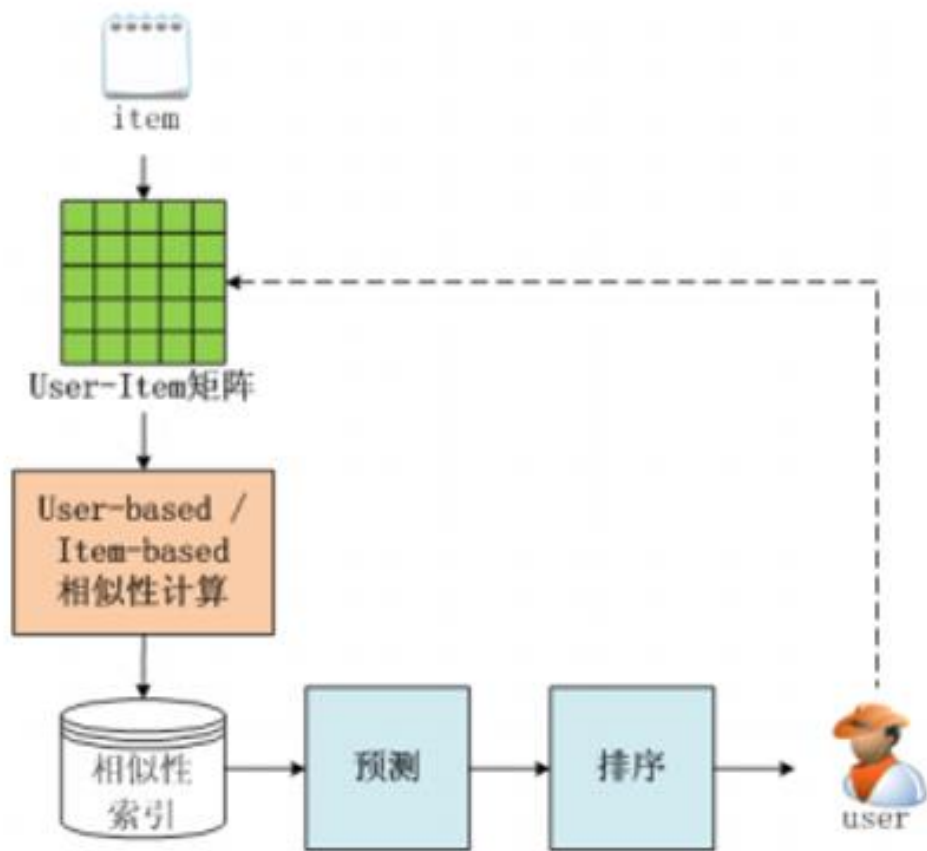

基于协同的推荐

Outline

协同过滤基础

【实践】基于协同的音乐推荐系统

系统框架



• 优点

- 充分利用群体智慧
- 推荐精度高于CB
- 利于挖掘隐含的相关性

• 缺点

- 推荐结果解释性较差
- 对时效性强的Item不适用
- 冷启动问题

协同算法

- User-Based CF
- Item-Based CF



User-Based CF

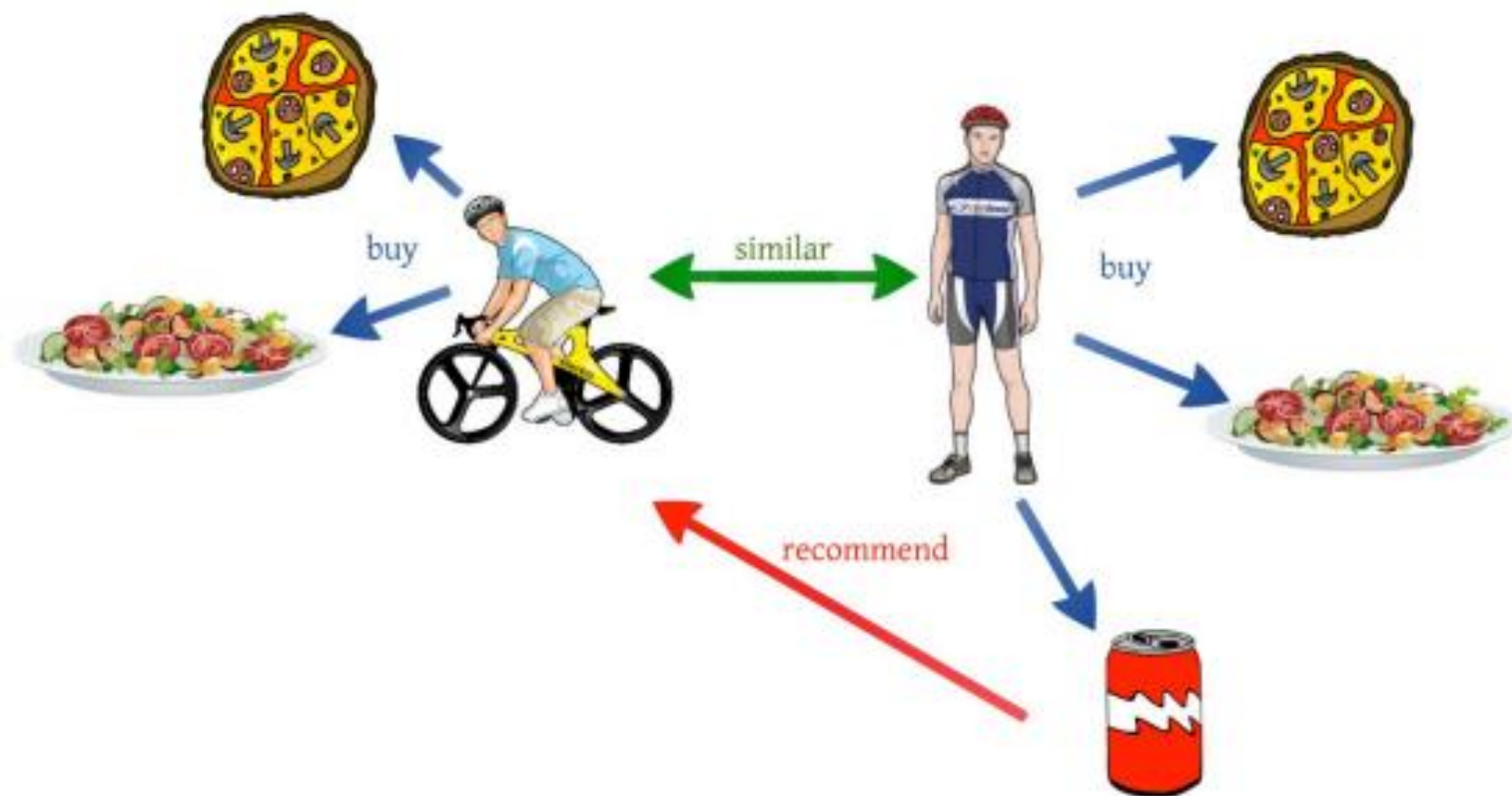
- 假设

- 用户喜欢那些跟他有相似爱好的用户喜欢的东西
- 具有相似兴趣的用户在未来也具有相似兴趣

- 方法

- 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.

User-Based CF



User-Based CF

	Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
A	5	1	?	2	2
B	1	5	2	5	5
C	2	?	3	5	4
D	4	3	5	3	?



	A	B	C	D
A		0.59	0.73	0.91
B	0.59		0.97	0.77
C	0.73	0.97		0.87
D	0.91	0.77	0.87	

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|}$$

$$r(C, Titanic) = \frac{0.97 * 5 + 0.87 * 3}{0.97 + 0.87} \approx 4.05$$

Item - 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 .

Item - Based CF

	Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
A	5	1	?	2	2
B	1	5	2	5	5
C	2	?	3	5	4
D	4	3	5	3	?

	Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
Matrix		0.57	0.99	0.69	0.63
Titanic	0.57		0.80	0.99	0.98
Die Hard	0.99	0.80		0.84	0.95
Forrest Gump	0.69	0.99	0.84		0.99
Wall-E	0.63	0.98	0.95	0.99	

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij} r_{uj}}{\sum_{j \in \mathcal{N}_u(i)} |w_{ij}|}$$

$$r(C, Titanic) = \frac{0.57 * 2 + 0.80 * 3 + 0.99 * 5 + 0.98 * 4}{0.57 + 0.80 + 0.99 + 0.98} \approx 3.72$$

User vs. Item

	User-based	Item-based
性能	适用用户较少场合，如果用户多，计算用户相似矩阵代价太大	适用于物品数明显小于用户数的场合，如果物品很多,计算物品相似度矩阵代价很大
领域	时效性强，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，不能立即对她进行个性化推荐，因为用户相似度表是每个一段时间离线计算的 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对它产生行为的用户兴趣相似的其他用户	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 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

冷启动

- 分为三类

- 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

冷启动

- 分为三类

- 物品冷启动

- 给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

Outline

协同过滤基础

【实践】基于协同的音乐推荐系统

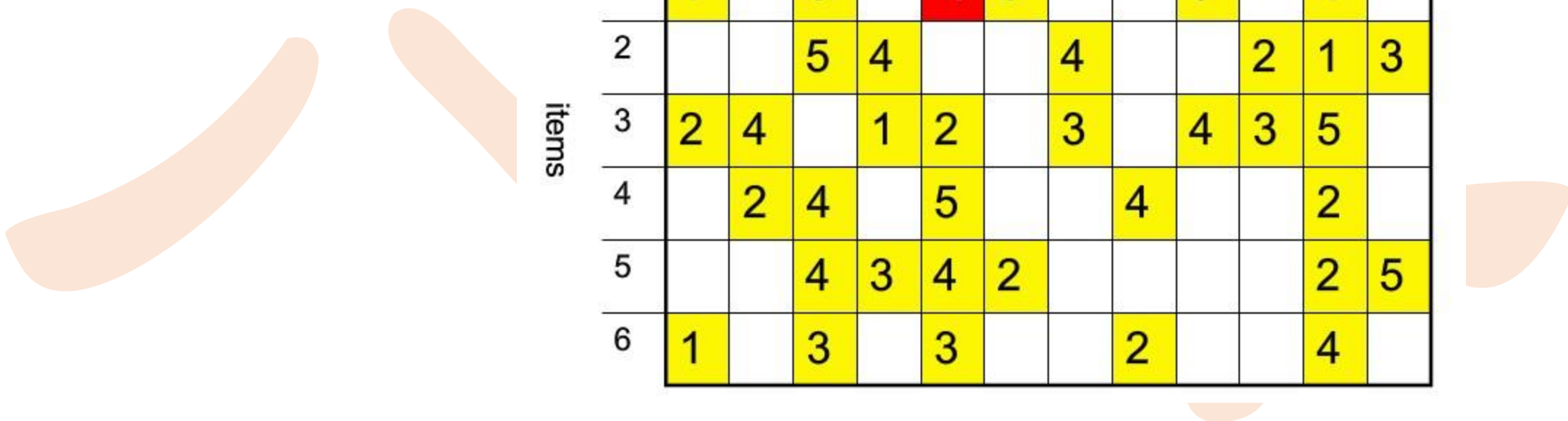
实现方案

- 倒排式
- 分块式



实现方案——倒排式

• 输入数据



		users											
		1	2	3	4	5	6	7	8	9	10	11	12
items	1	1		3		?	5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	

实现方案——倒排式

• 相似度计算公式

$$w_{ij} = \frac{(\sum_{u \in U(i,j)} r'_{ui} * r'_{uj}) * (|U(i,j)| - 1)}{\sqrt{\sum_{u \in U(i,j)} r'^2_{ui} * \sum_{u \in U(i,j)} r'^2_{uj}} * (|U(i,j)| - 1 + \lambda)}$$

其中：

w_{ij} 表示标号为*i*, *j*的两个item的相似度

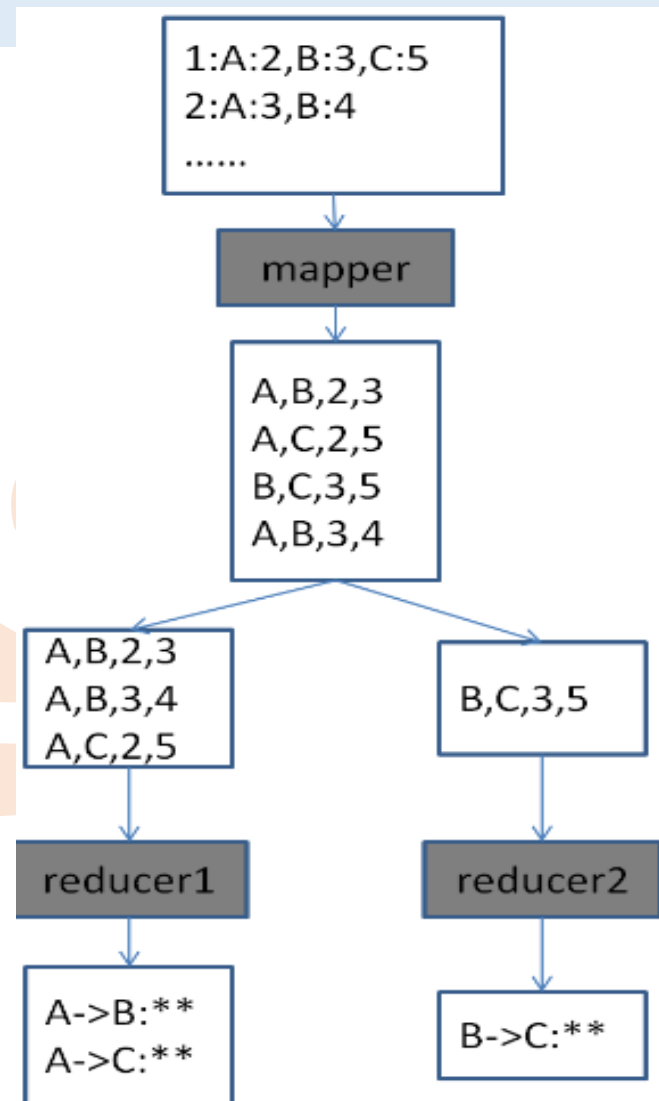
$U(i,j)$ 表示同时对*i*, *j*有评分的用户的集合

R_{ui} 表示用户*u*对item *i*的评分

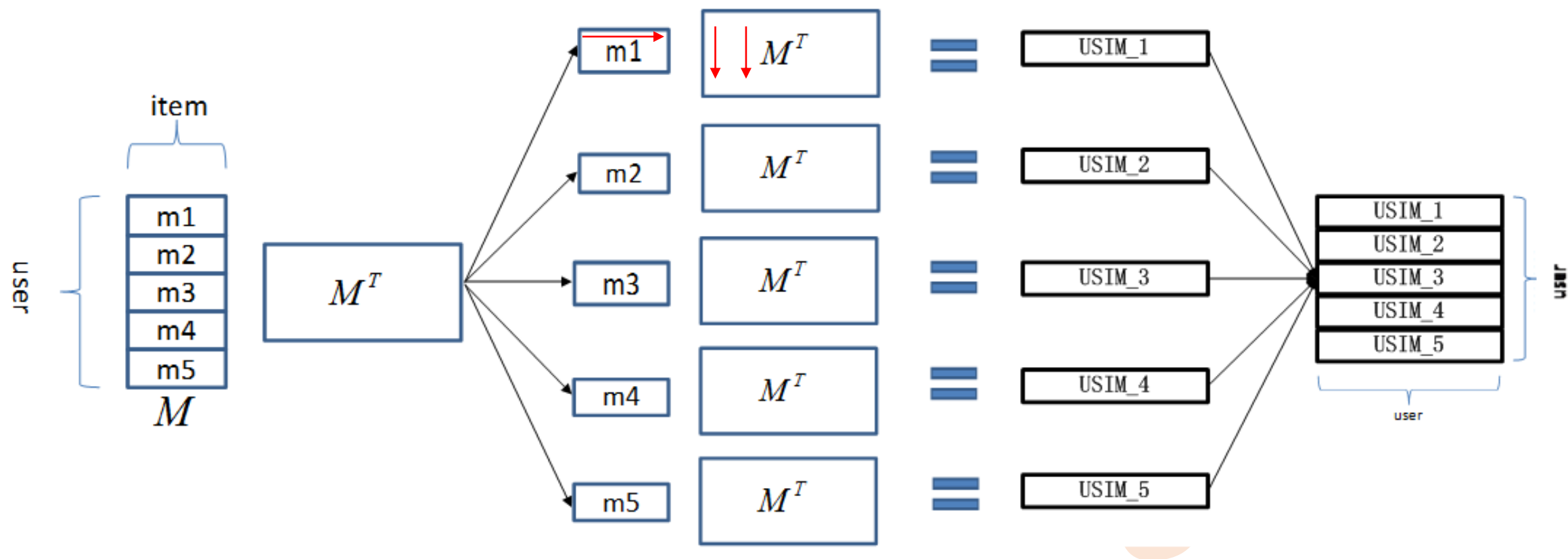
参数 λ 为平滑（惩罚）参数

实现方案——倒排式

- 详细Hadoop过程：
 - 具体做法就是：
 - 在MR的map阶段将每个用户的评分item组合成pair
 $\langle \text{left}, \text{right}, \text{leftscore}, \text{rightscore} \rangle$ 输出，left作为分发键，left+right作为排序键。
 - 在reduce阶段，将map中过来的数据扫一遍即可求得所有item的相似度。



实现方案——分块式



实现方案——倒排式开发步骤

- 第一步：准备数据
 - 数据格式：uid,itemid,score

1,100001,5

1,100002,3

1,100003,4

1,100004,3

1,100005,3

1,100007,4

1,100008,1

1,100009,5

1,1000011,2

实现方案——倒排式

- 第二步：开发MapReduce程序
 - 1、归一UI矩阵
 - Map
 - In
 - » Line: u, i, s
 - Out
 - » Key: i
 - » Value: u, s
 - Reduce
 - In
 - » Key: i
 - » Value: list((u, s))
 - Out
 - » Key: u
 - » Value: i, s_new

实现方案——倒排式

- 第二步：开发MapReduce程序
 - 2、衍生II Pair对
 - Map
 - In
 - » Key: u
 - » Value: i, s_new
 - Out
 - » Key: u
 - » Value: i, s_new
 - Reduce
 - In
 - » Key: u
 - » Value: list((i, s_new))
 - Out
 - » Key_1: i
 - » Value_1: j, s_new_i * s_new_j
 - » Key_2: j
 - » Value_2: i, s_new_i * s_new_j

实现方案——倒排式

- 第三步：开发MapReduce程序
 - 3、生成结果
 - Map
 - In
 - » Key: i
 - » Value: j, $s_{new_i} * s_{new_j}$
 - Out
 - » Key: $\langle i, j \rangle$
 - » Value: $s_{new_i} * s_{new_j}$
 - Reduce
 - In
 - » Key: $\langle i, j \rangle$
 - » Value: $list((s_{new_i} * s_{new_j}))$
 - Out
 - » Key: i
 - » Value: j, score

Q & A

@八斗学院
