

基于用户的协同过滤 (UserCF)

王树森

ShusenWang@xiaohongshu.com

<http://wangshusen.github.io/>

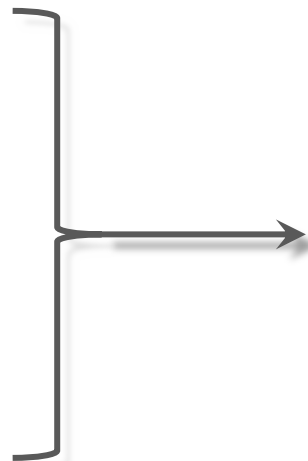


UserCF的原理

有很多跟我兴趣非常相似的网友

其中某个网友对某笔记点赞、转发

我没看过这篇笔记



给我推荐这篇笔记

UserCF的原理

有很多跟我兴趣非常相似的网友

其中某个网友对某笔记点赞、转发

我没看过这篇笔记

给我推荐这篇笔记

推荐系统如何找到跟我兴趣非常相似的网友呢？

- 方法一：点击、点赞、收藏、转发的笔记有很大的重合。
- 方法二：关注的作者有很大的重合。

UserCF 的实现

用户之间的相似度:

$$\text{sim}(\text{user}, \text{user}_j)$$



User

0.9

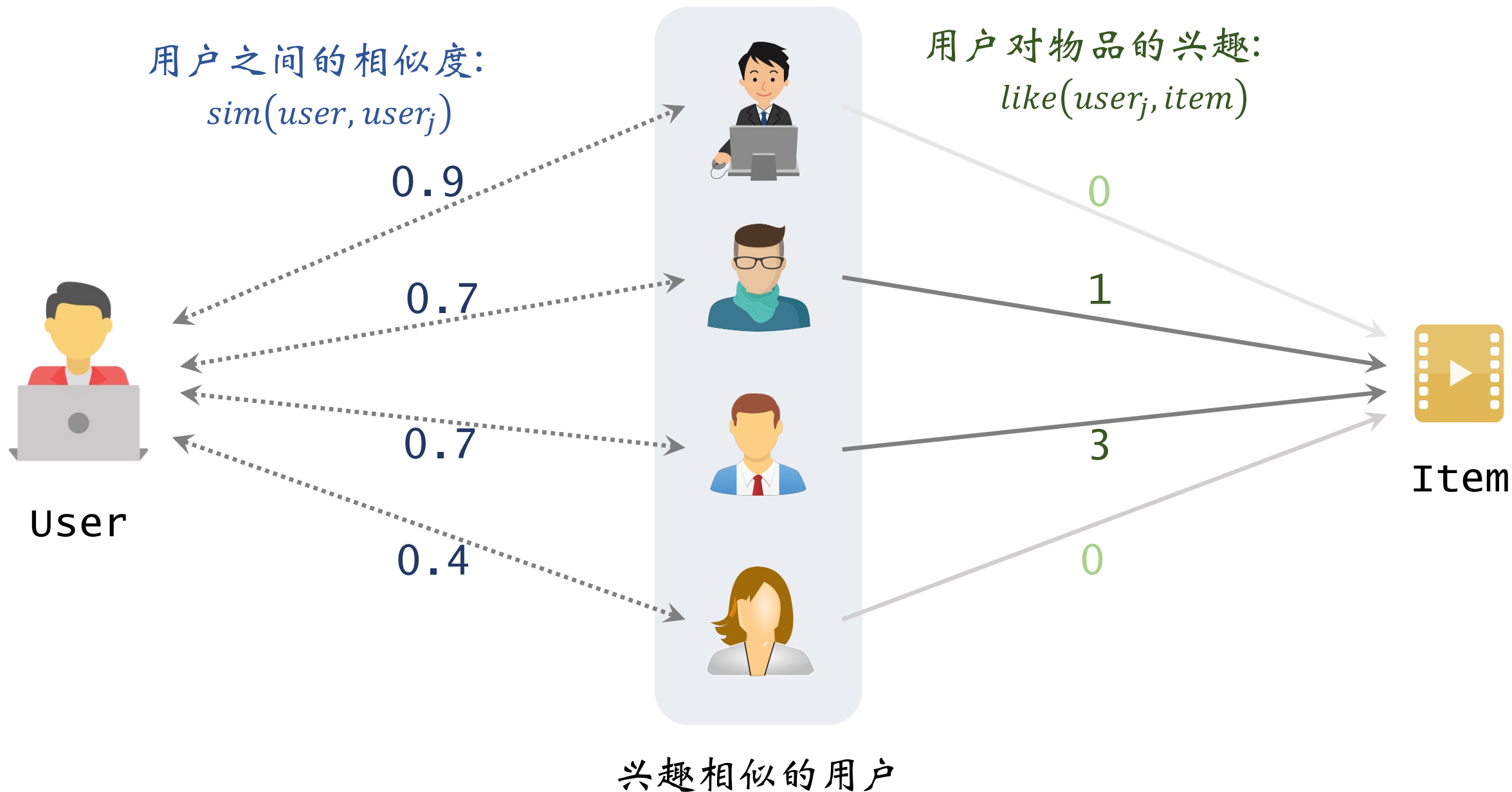
0.7

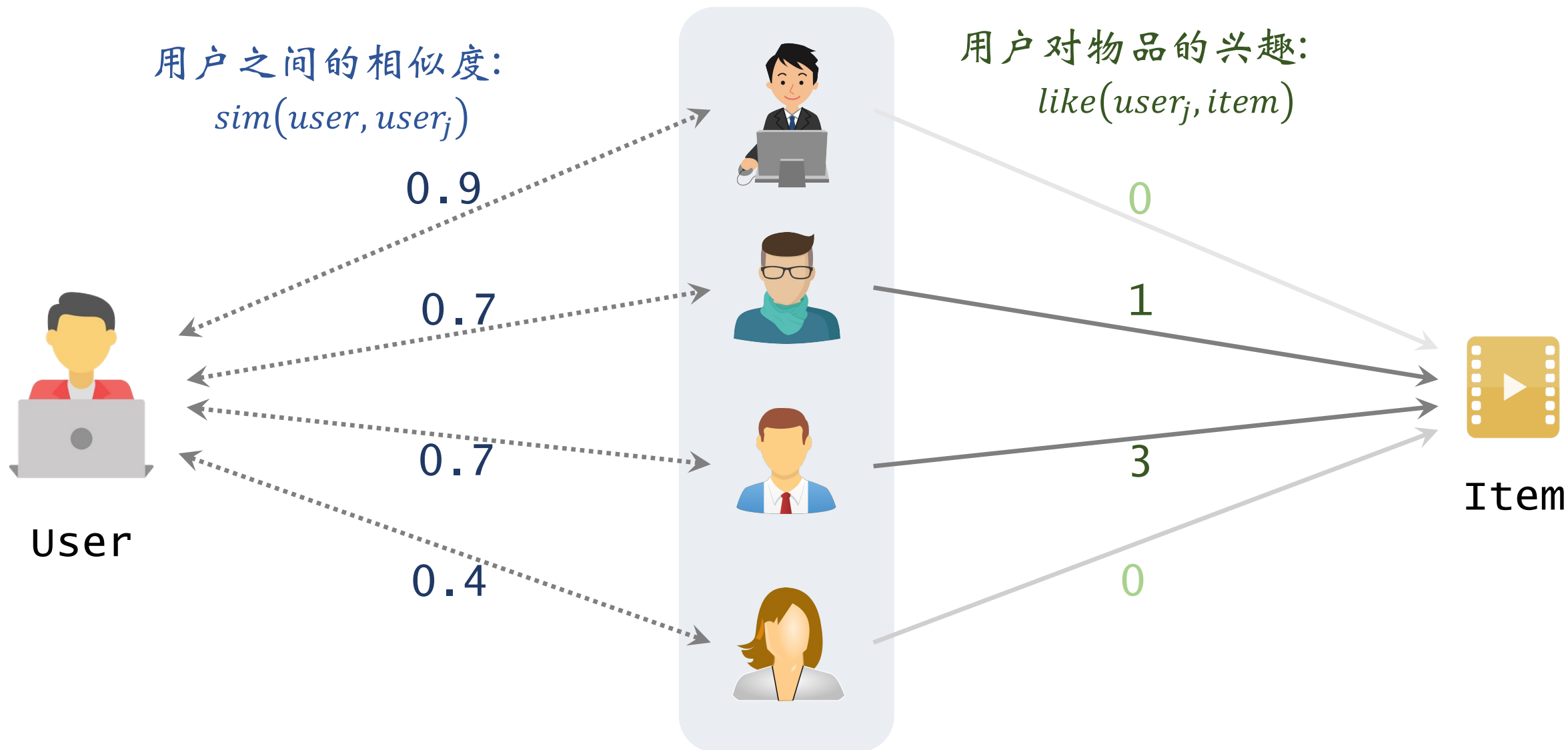
0.7

0.4

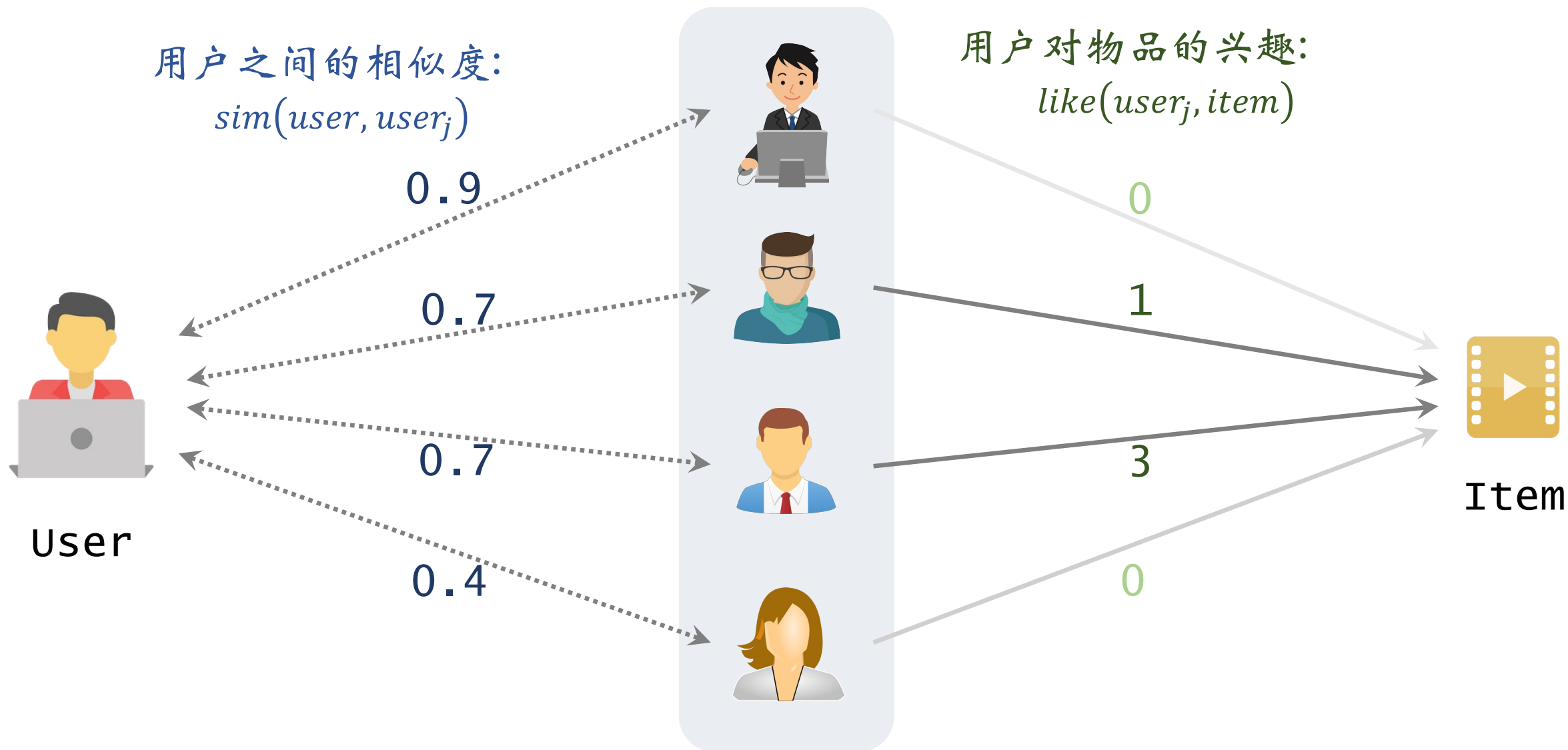


兴趣相似的用户





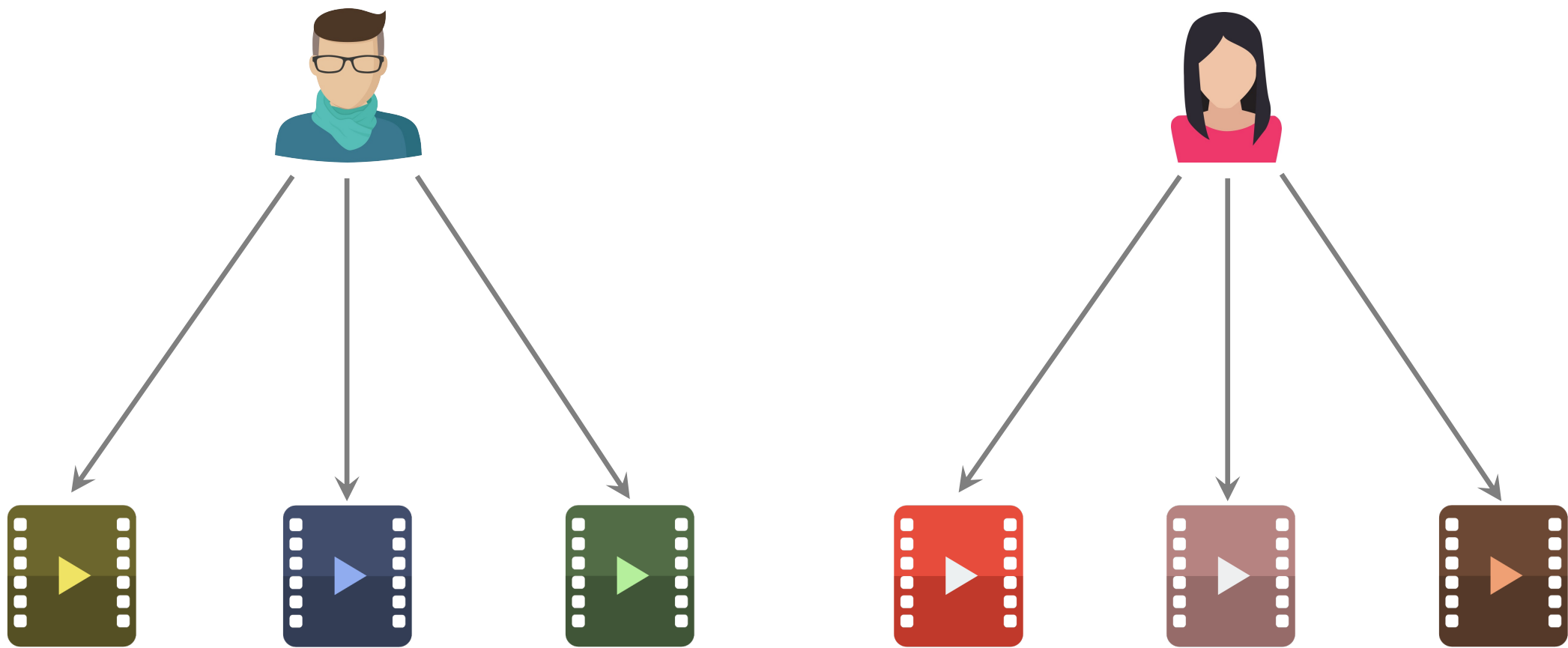
预估用户对候选物品的兴趣： $\sum_j \underline{sim(user, user_j)} \times \underline{like(user_j, item)}$



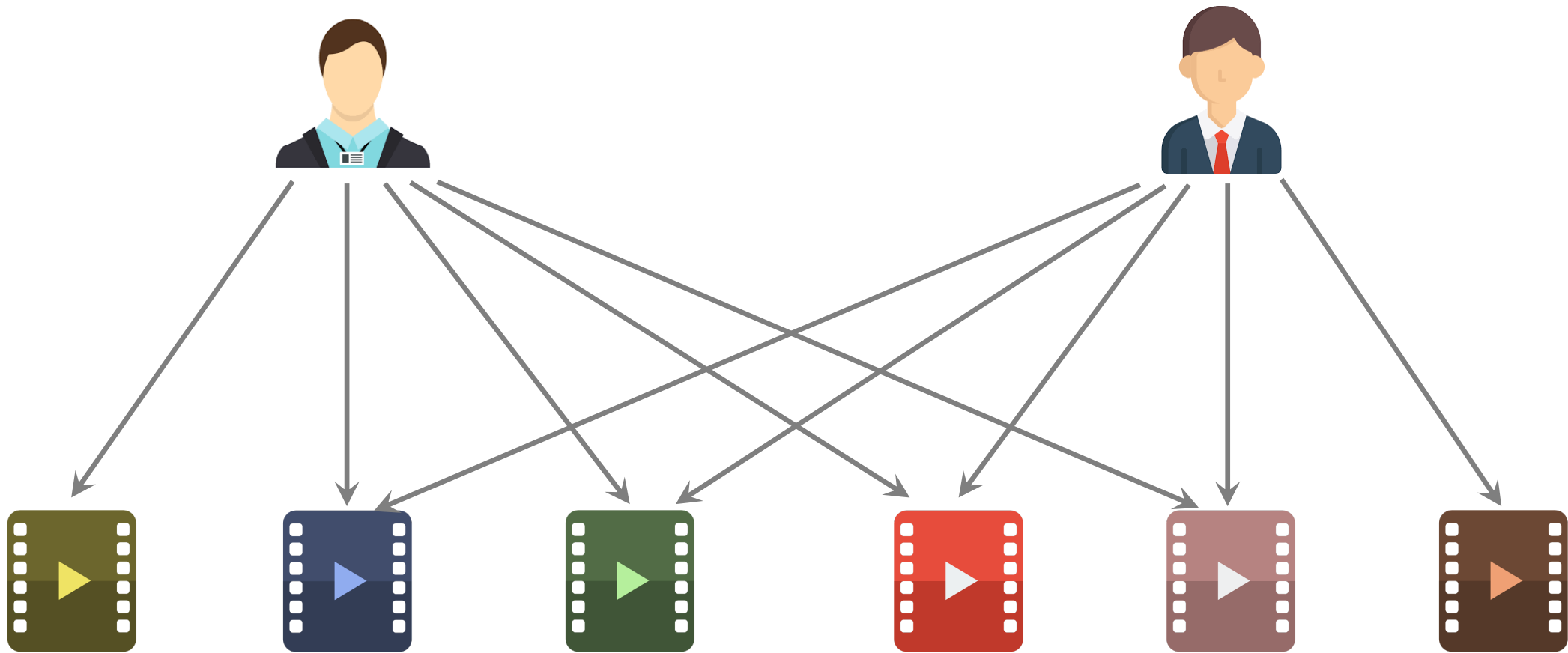
预估用户对候选物品的兴趣：
 $0.9 \times 0 + 0.7 \times 1 + 0.7 \times 3 + 0.4 \times 0 = 2.8$

用户的相似度

两个用户不相似



两个用户相似



计算用户相似度

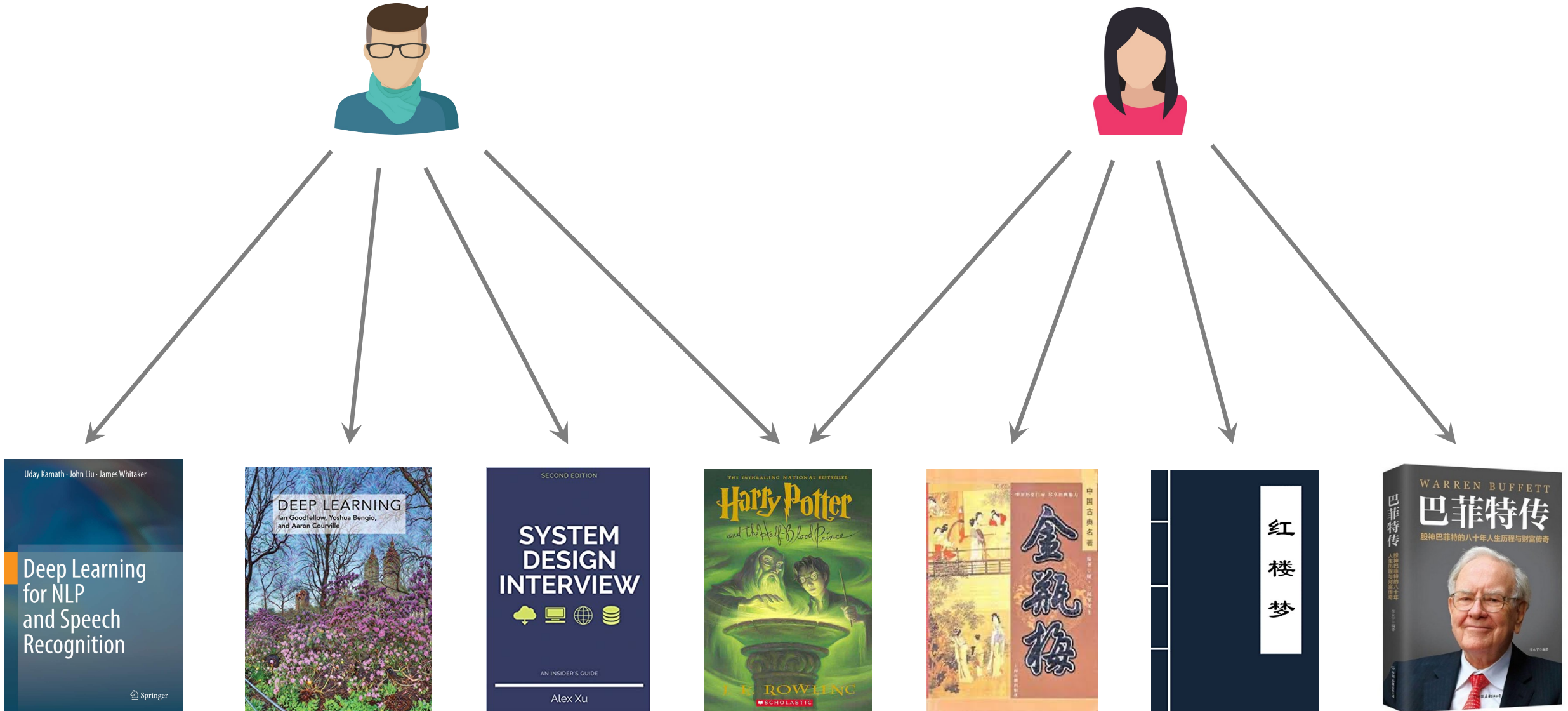
- 用户 u_1 喜欢的物品记作集合 \mathcal{J}_1 。
- 用户 u_2 喜欢的物品记作集合 \mathcal{J}_2 。
- 定义交集 $I = \mathcal{J}_1 \cap \mathcal{J}_2$ 。

计算用户相似度

- 用户 u_1 喜欢的物品记作集合 \mathcal{J}_1 。
- 用户 u_2 喜欢的物品记作集合 \mathcal{J}_2 。
- 定义交集 $I = \mathcal{J}_1 \cap \mathcal{J}_2$ 。
- 两个用户的相似度：

$$sim(u_1, u_2) = \frac{|I|}{\sqrt{|\mathcal{J}_1| \cdot |\mathcal{J}_2|}}$$

降低热门物品权重



降低热门物品权重

- 用户 u_1 喜欢的物品记作集合 \mathcal{J}_1 。
- 用户 u_2 喜欢的物品记作集合 \mathcal{J}_2 。
- 定义交集 $I = \mathcal{J}_1 \cap \mathcal{J}_2$ 。
- 两个用户的相似度：

不论冷门、热门，
物品权重都是 1。

$$\text{sim}(u_1, u_2) = \frac{\sum_{l \in I} 1}{\sqrt{|\mathcal{J}_1| \cdot |\mathcal{J}_2|}} = |I|$$

降低热门物品权重

- 用户 u_1 喜欢的物品记作集合 \mathcal{J}_1 。
- 用户 u_2 喜欢的物品记作集合 \mathcal{J}_2 。
- 定义交集 $I = \mathcal{J}_1 \cap \mathcal{J}_2$ 。
- 两个用户的相似度：

$$\text{sim}(u_1, u_2) = \frac{\sum_{l \in I} \frac{1}{\log(1 + n_l)}}{\sqrt{|\mathcal{J}_1| \cdot |\mathcal{J}_2|}}.$$

n_l ：喜欢物品 l 的用户数量，反映物品的热门程度

小结

- UserCF 的基本思想：
 - 如果用户 $user_1$ 跟用户 $user_2$ 相似，而且 $user_2$ 喜欢某物品，
 - 那么用户 $user_1$ 也很可能喜欢该物品。

小结

- UserCF 的基本思想：
 - 如果用户 $user_1$ 跟用户 $user_2$ 相似，而且 $user_2$ 喜欢某物品，
 - 那么用户 $user_1$ 也很可能喜欢该物品。
- 预估用户 $user$ 对候选物品 $item$ 的兴趣：

$$\sum_j \text{sim}(user, user_j) \times \text{like}(user_j, item).$$

小结

- UserCF 的基本思想：
 - 如果用户 $user_1$ 跟用户 $user_2$ 相似，而且 $user_2$ 喜欢某物品，
 - 那么用户 $user_1$ 也很可能喜欢该物品。

- 预估用户 $user$ 对候选物品 $item$ 的兴趣：

$$\sum_j \boxed{sim(user, user_j)} \times like(user_j, item).$$

- 计算两个用户的相似度：
 - 把每个用户表示为一个稀疏向量，向量每个元素对应一个物品。
 - 相似度 sim 就是两个向量夹角的余弦。

UserCF 召回的完整流程

事先做离线计算

建立“用户→物品”的索引

- 记录每个用户最近点击、交互过的物品ID。
- 给定任意用户ID，可以找到他近期感兴趣的物品列表。

事先做离线计算

建立“用户→物品”的索引

- 记录每个用户最近点击、交互过的物品ID。
- 给定任意用户ID，可以找到他近期感兴趣的物品列表。

建立“用户→用户”的索引





- 对于每个用户，索引他最相似的 k 个用户。
- 给定任意用户ID，可以快速找到他最相似的 k 个用户。

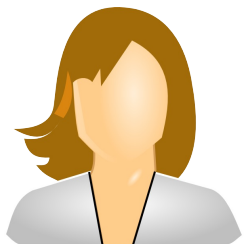
“用户→物品”的索引

用户：

(物品ID，兴趣分数) 的列表：



	， 2		， 1		， 4		， 3	...
---	-----	---	-----	---	-----	---	-----	-----



	， 3		， 1		， 1		， 1	...
--	-----	--	-----	--	-----	--	-----	-----



“用户→用户”的索引






用户：








“用户→用户”的索引

用户：最相似的 k 个用户的 (ID, 相似度)：



	, 0.7		, 0.6		, 0.6		0.3		, 0.3
---	-------	---	-------	---	-------	---	-----	---	-------



	, 0.9		, 0.6		, 0.6		, 0.5		, 0.4
--	-------	--	-------	--	-------	--	-------	--	-------



线上做召回

1. 给定用户ID，通过“用户 \rightarrow 用户”索引，找到 top-k 相似用户。
2. 对于每个 top-k 相似用户，通过“用户 \rightarrow 物品”索引，找到用户近期感兴趣的物品列表 (last-n)。



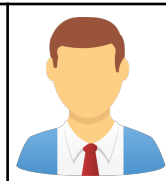
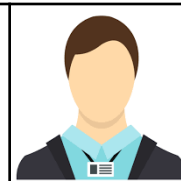
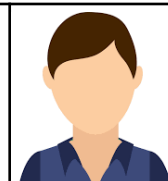
线上做召回

1. 给定用户ID，通过“用户 \rightarrow 用户”索引，找到 top-k 相似用户。
2. 对于每个 top-k 相似用户，通过“用户 \rightarrow 物品”索引，找到用户近期感兴趣的物品列表 (last-n)。
3. 对于取回的 nk 个相似物品，用公式预估用户对每个物品的兴趣分数。
4. 返回分数最高的100个物品，作为召回结果。

线上做召回

Top-k 相似的用户 (ID, 相似度)



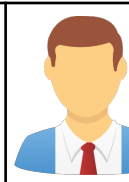
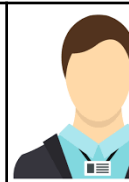
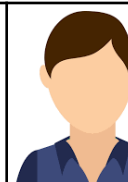


	, 0.7		, 0.6		, 0.6		0.3		0.3

线上做召回



Top-k 相似的用户 (ID, 相似度)



 , 0.7					 , 0.6					 , 0.6					 0.3					 , 0.3				
---	--	--	--	--	---	--	--	--	--	---	--	--	--	--	---	--	--	--	--	---	--	--	--	--








用户感兴趣的
n个物品

	, 1
	, 3

线上做召回

Top-k 相似的用户 (ID, 相似度)







 , 0.7					 , 0.6					 , 0.6					 0.3					 , 0.3				
---	--	--	--	--	---	--	--	--	--	---	--	--	--	--	---	--	--	--	--	---	--	--	--	--



用户感兴趣的
n个物品





 , 1
 , 3

 , 4
 , 1

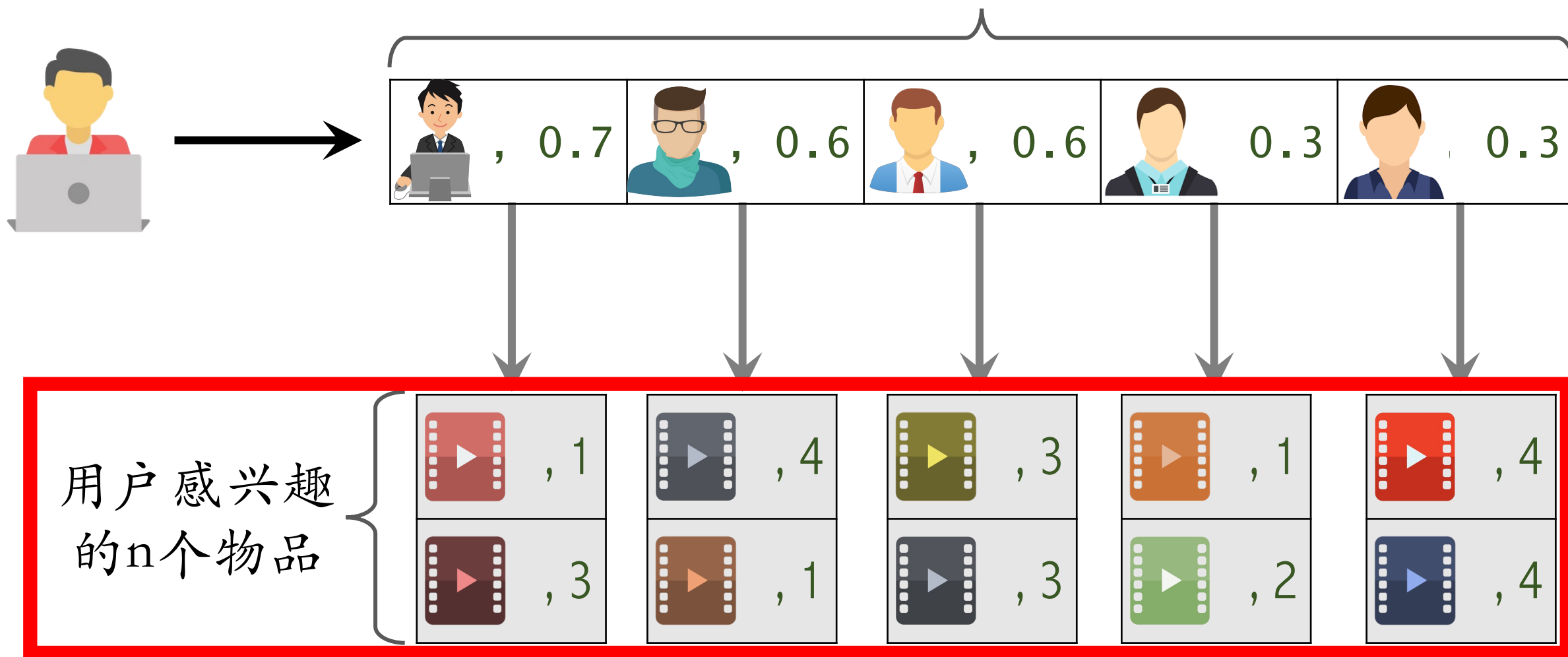
 , 3
 , 3

 , 1
 , 2

 , 4
 , 4

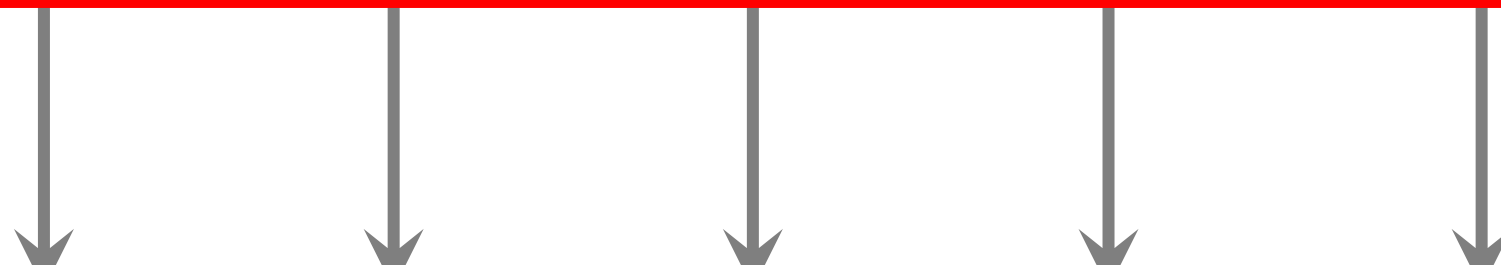
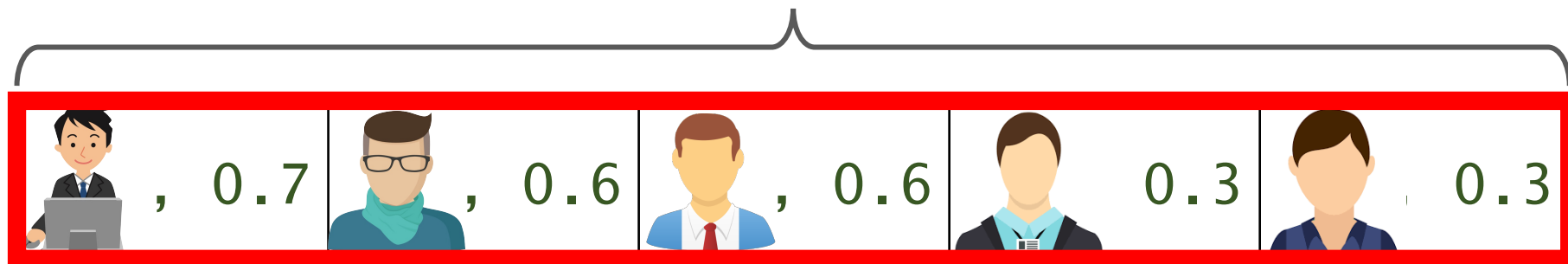
线上做召回

Top-k 相似的用户 (ID, 相似度)



线上做召回

Top-k 相似的用户 (ID, 相似度)



用户感兴趣的
n个物品



总结

UserCF的原理

- 用户 u_1 跟用户 u_2 相似，而且 u_2 喜欢某物品，那么 u_1 也可能喜欢该物品。
- 用户相似度：
 - 如果用户 u_1 和 u_2 喜欢的物品有很大的重叠，那么 u_1 和 u_2 相似。
 - 公式：
$$\text{sim}(u_1, u_2) = \frac{|J_1 \cap J_2|}{\sqrt{|J_1| \cdot |J_2|}}$$
。

UserCF召回通道

- 维护两个索引：
 - 用户 \rightarrow 物品列表：用户近期交互过的 n 个物品。
 - 用户 \rightarrow 用户列表：相似度最高的 k 个用户。
- 线上做召回：
 - 利用两个索引，每次取回 nk 个物品。
 - 预估用户 $user$ 对每个物品 $item$ 的兴趣分数：
$$\sum_j sim(user, user_j) \times like(user_j, item).$$
 - 返回分数最高的100个物品，作为召回结果。

长期招聘优秀的算法工程师

- 部门：小红书社区技术部。
- 方向：搜索、推荐。
- 职位：校招、社招、实习。
- 地点：上海、北京。
- 联系方式：`ShusenWang@xiaohongshu.com`

Thank You!

<http://wangshusen.github.io/>