

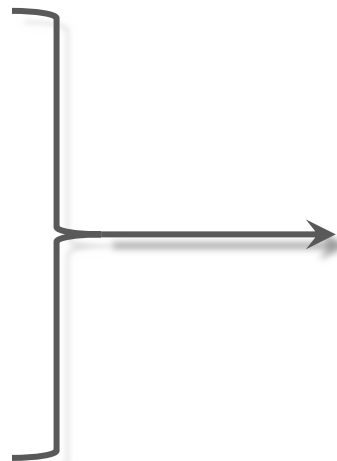
# ItemCF的原理

我喜欢看《笑傲江湖》

《笑傲江湖》与《鹿鼎记》相似

我没看过《鹿鼎记》

给我推荐《鹿鼎记》



# ItemCF的原理

我喜欢看《笑傲江湖》

《笑傲江湖》与《鹿鼎记》相似

我没看过《鹿鼎记》

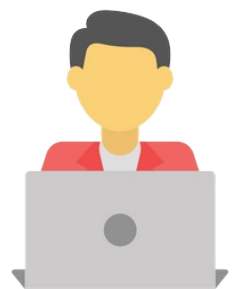
给我推荐《鹿鼎记》

推荐系统如何知道《笑傲江湖》与《鹿鼎记》相似？

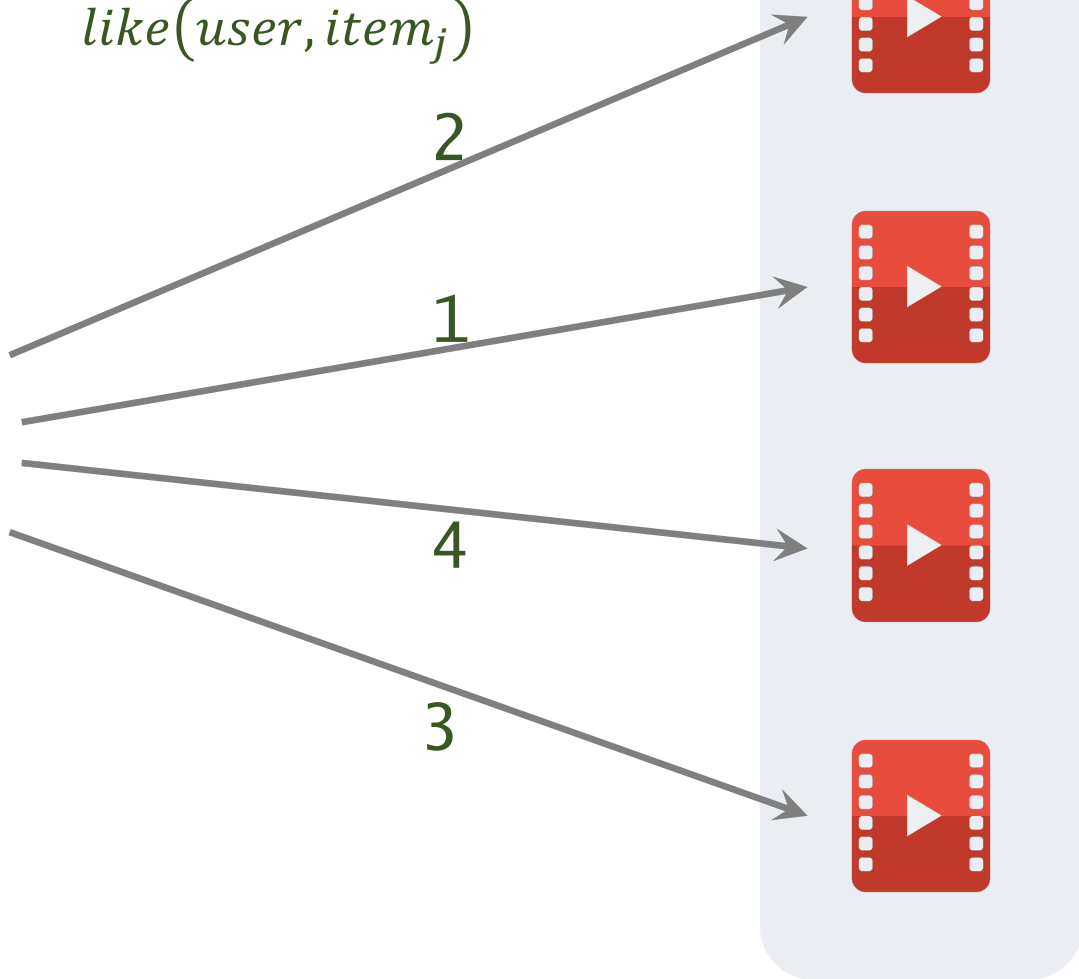
- 看过《笑傲江湖》的用户也看过《鹿鼎记》。
- 给《笑傲江湖》好评的用户也给《鹿鼎记》好评。

# ItemCF 的实现

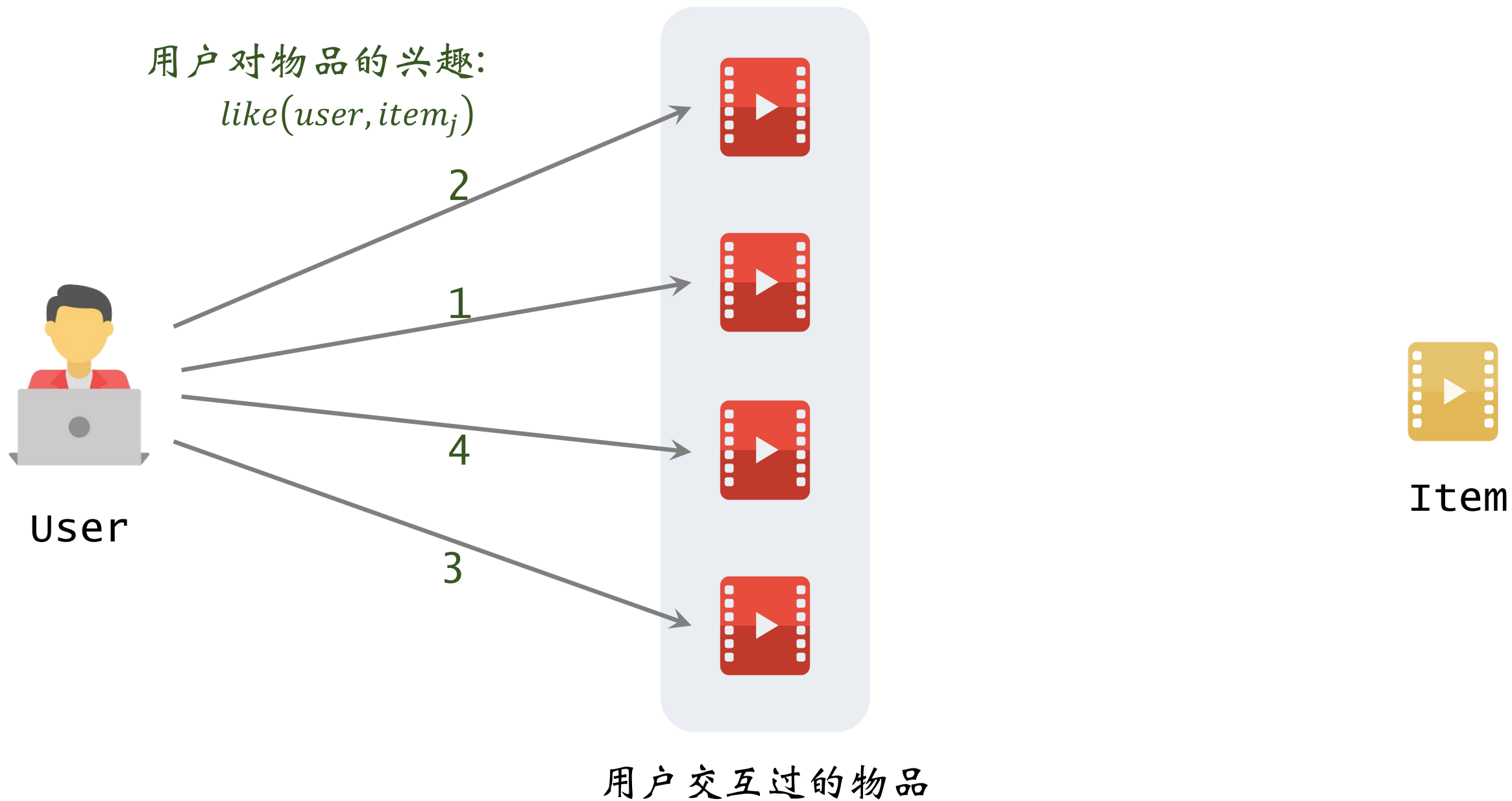
用户对物品的兴趣:  
 $like(user, item_j)$

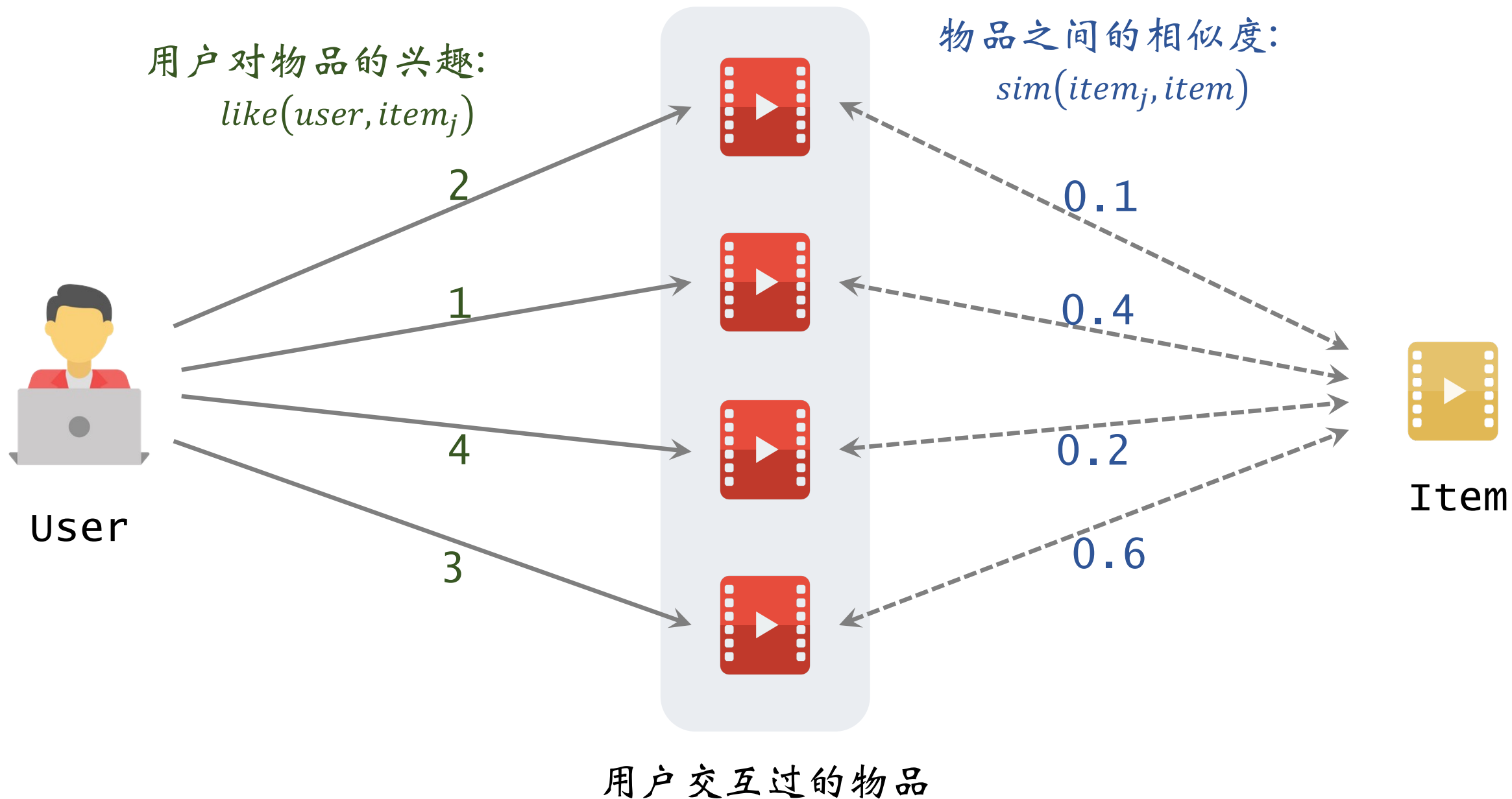


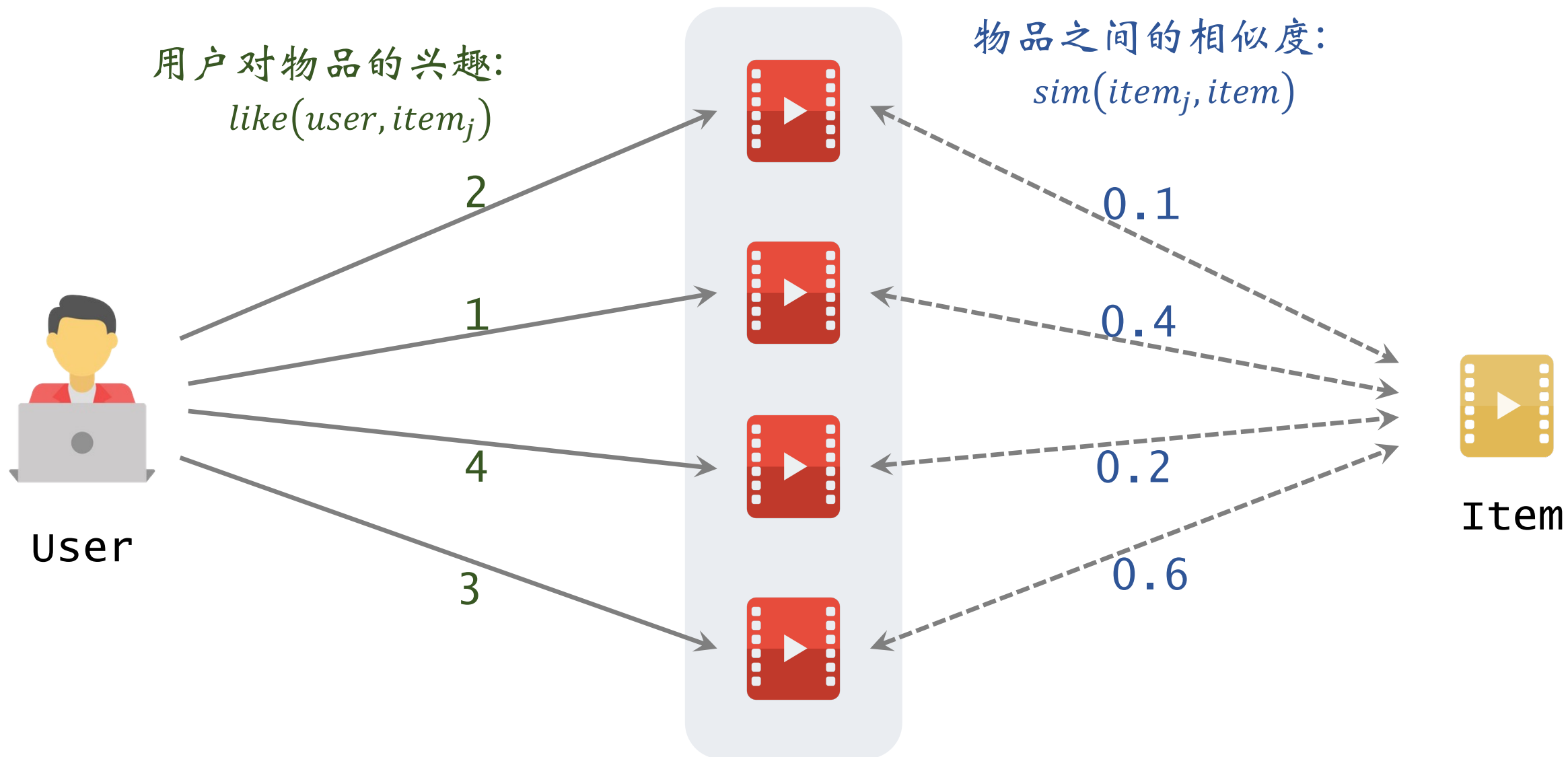
User



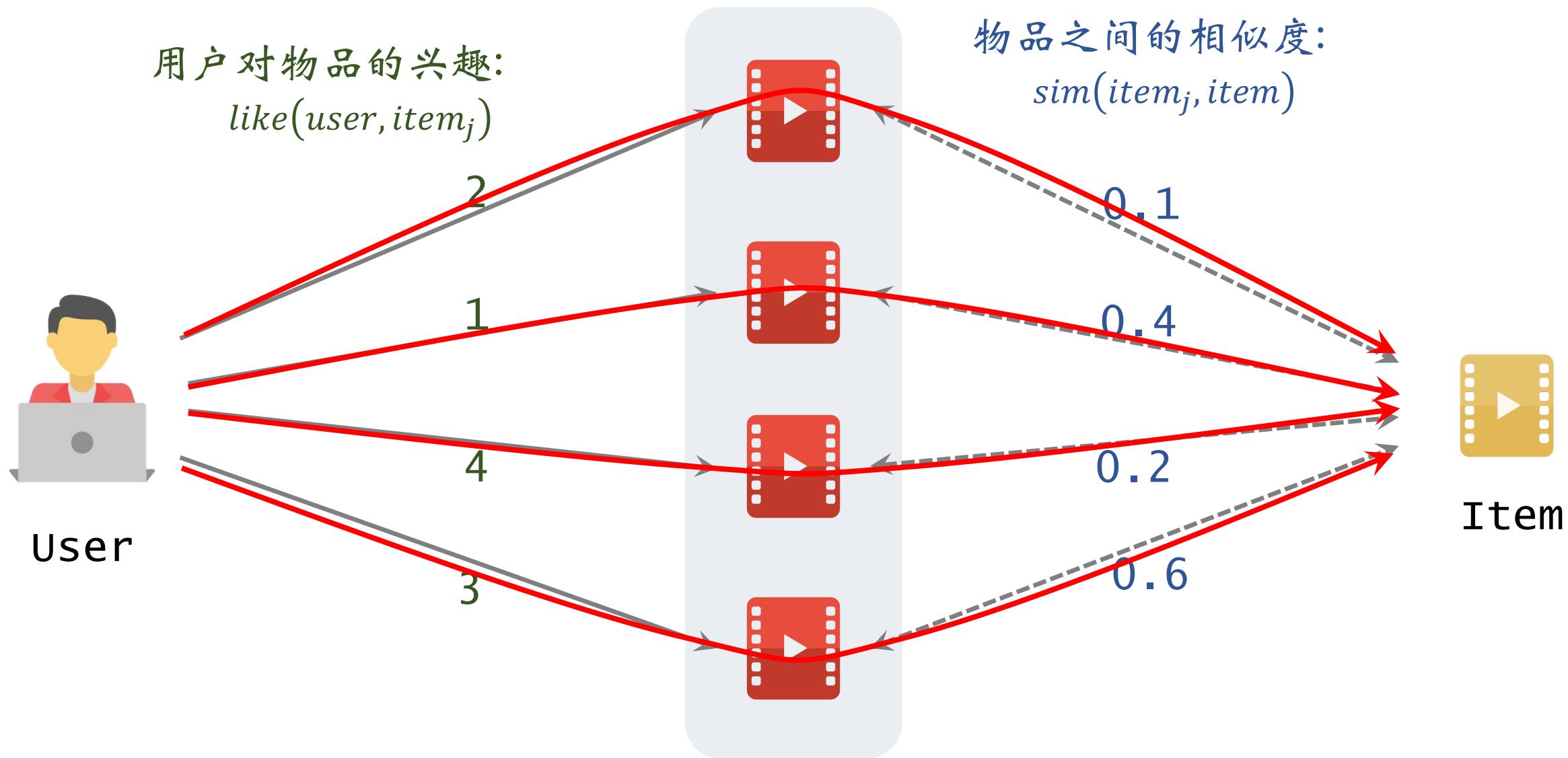
用户交互过的物品







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



预估用户对候选物品的兴趣：  
 $2 \times 0.1 + 1 \times 0.4 + 4 \times 0.2 + 3 \times 0.6 = 3.2$

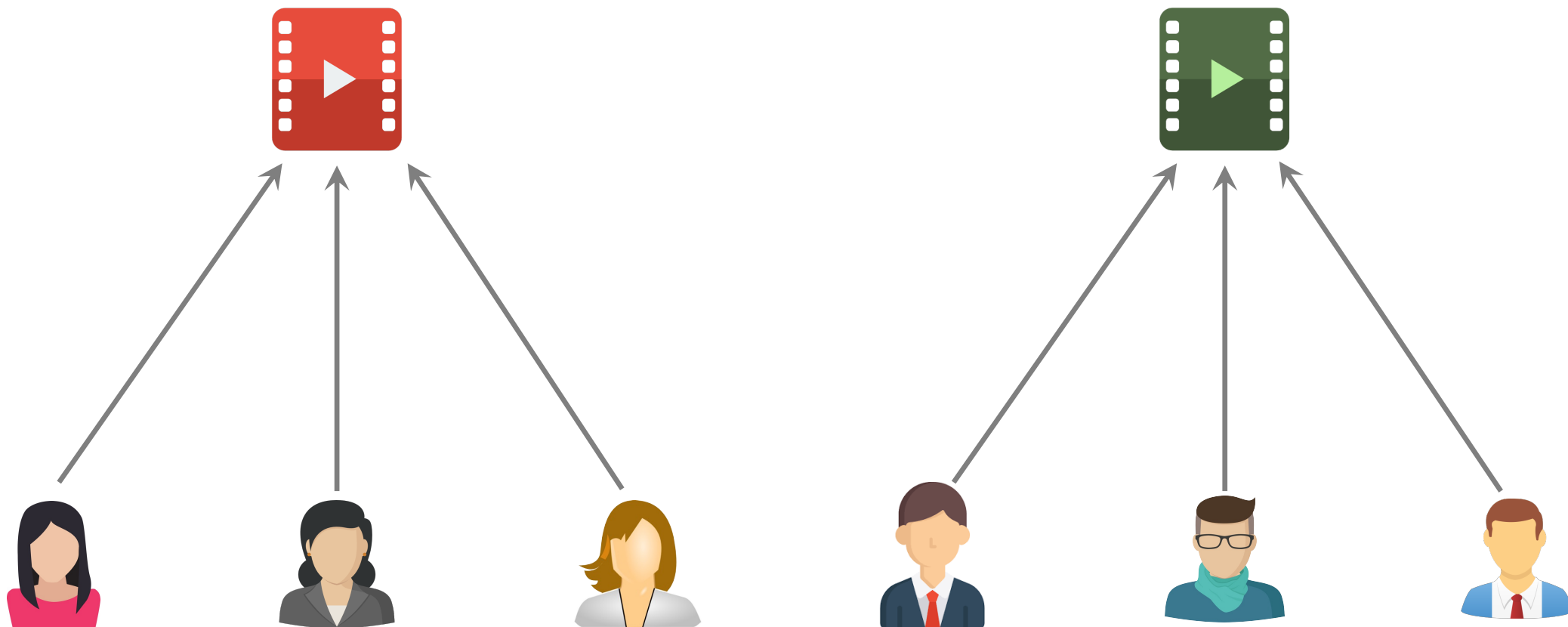


# 物品的相似度

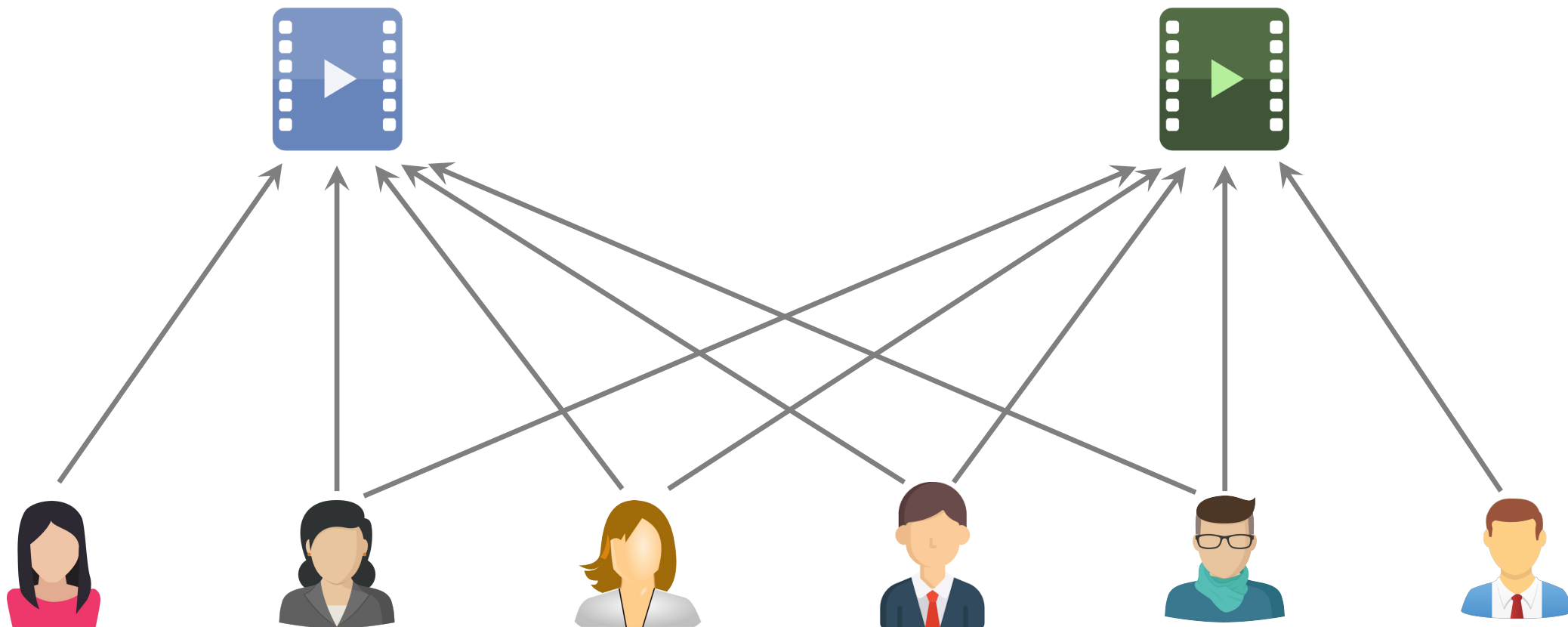
# 物品相似度

- 两个物品的受众重合度越高，两个物品越相似。
- 例如：
  - 喜欢《射雕英雄传》和《神雕侠侣》的读者重合度很高。
  - 可以认为《射雕英雄传》和《神雕侠侣》相似。

# 两个物品不相似



# 两个物品相似



# 计算物品相似度

- 喜欢物品  $i_1$  的用户记作集合  $\mathcal{W}_1$ 。
- 喜欢物品  $i_2$  的用户记作集合  $\mathcal{W}_2$ 。
- 定义交集  $\mathcal{V} = \mathcal{W}_1 \cap \mathcal{W}_2$ 。

# 计算物品相似度

- 喜欢物品  $i_1$  的用户记作集合  $\mathcal{W}_1$ 。
- 喜欢物品  $i_2$  的用户记作集合  $\mathcal{W}_2$ 。
- 定义交集  $\mathcal{V} = \mathcal{W}_1 \cap \mathcal{W}_2$ 。
- 两个物品的相似度：

$$sim(i_1, i_2) = \frac{|\mathcal{V}|}{\sqrt{|\mathcal{W}_1| \cdot |\mathcal{W}_2|}}.$$

注：公式没有考虑喜欢的程度  $like(user, item)$

# 计算物品相似度

- 喜欢物品  $i_1$  的用户记作集合  $\mathcal{W}_1$ 。
- 喜欢物品  $i_2$  的用户记作集合  $\mathcal{W}_2$ 。
- 定义交集  $\mathcal{V} = \mathcal{W}_1 \cap \mathcal{W}_2$ 。
- 两个物品的相似度：

$$sim(i_1, i_2) = \frac{\sum_{v \in \mathcal{V}} like(v, i_1) \cdot like(v, i_2)}{\sqrt{\sum_{u_1 \in \mathcal{W}_1} like^2(u_1, i_1)} \cdot \sqrt{\sum_{u_2 \in \mathcal{W}_2} like^2(u_2, i_2)}}$$

余弦相似度 (cosine similarity)

# 小结

- ItemCF 的基本思想：
  - 如果用户喜欢物品  $item_1$ ，而且物品  $item_1$  与  $item_2$  相似，
  - 那么用户很可能喜欢物品  $item_2$ 。



# 小结

- ItemCF 的基本思想：
  - 如果用户喜欢物品  $item_1$ ，而且物品  $item_1$  与  $item_2$  相似，
  - 那么用户很可能喜欢物品  $item_2$ 。
- 预估用户对候选物品的兴趣：

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

# 小结

- ItemCF 的基本思想：
  - 如果用户喜欢物品  $item_1$ ，而且物品  $item_1$  与  $item_2$  相似，
  - 那么用户很可能喜欢物品  $item_2$ 。

- 预估用户对候选物品的兴趣：

$$\sum_j like(user, item_j) \times sim(item_j, item) .$$

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

# ItemCF 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

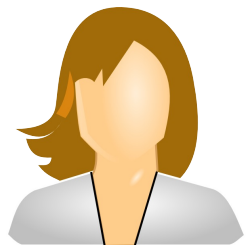
- 记录每个用户最近点击、交互过的物品ID。
- 给定任意用户ID，可以找到他近期感兴趣的物品列表。

## 建立“物品→物品”的索引

- 计算物品之间两两相似度。
- 对于每个物品，索引它最相似的  $k$  个物品。
- 给定任意物品ID，可以快速找到它最相似的  $k$  个物品。

# “用户→物品”的索引

用户：







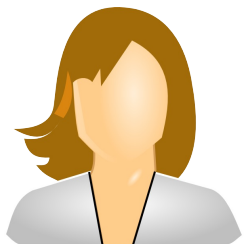
# “用户→物品”的索引

用户：

(物品ID，兴趣分数) 的列表：



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

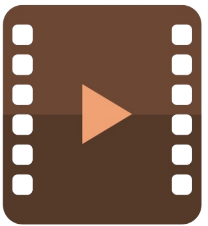
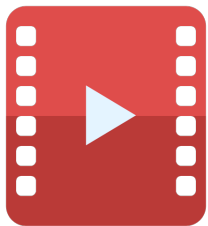


	， 3		， 1		， 1		， 1	...
--	-----	--	-----	--	-----	--	-----	-----



# “物品→物品”的索引

物品：



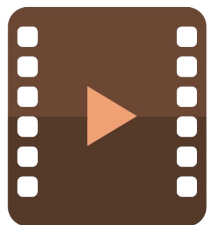


# “物品→物品”的索引

物品：                      最相似的  $k$  个物品的 (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 $\rightarrow$ 物品”索引，找到用户近期感兴趣的物品列表（last-n）。
2. 对于last-n列表中每个物品，通过“物品 $\rightarrow$ 物品”的索引，找到 top-k 相似物品。

# 线上做召回

1. 给定用户ID，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）。
2. 对于last-n列表中每个物品，通过“物品 $\rightarrow$ 物品”的索引，找到 top-k 相似物品。
3. 对于取回的相似物品（最多有  $nk$  个），用公式预估用户对物品的兴趣分数。
4. 返回分数最高的100个物品，作为推荐结果。

# 线上做召回

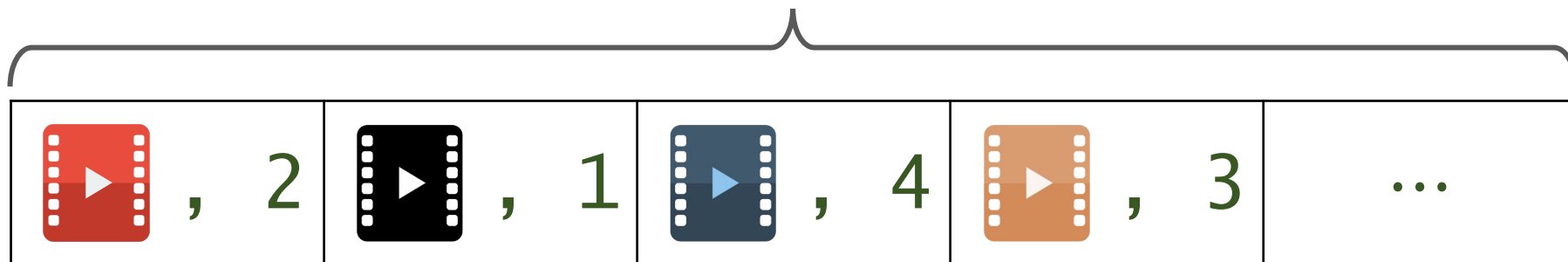
索引的意义在于避免枚举所有的物品。

1. 记录用户最近感兴趣的  $n = 200$  个物品。
2. 取回每个物品最相似的  $k = 10$  个物品。
3. 给取回的  $nk = 2000$  个物品打分（用户对物品的兴趣）。
4. 返回分数最高的 100 个物品作为 ItemCF 通道的输出。

用索引，离线计算量大，线上计算量小。

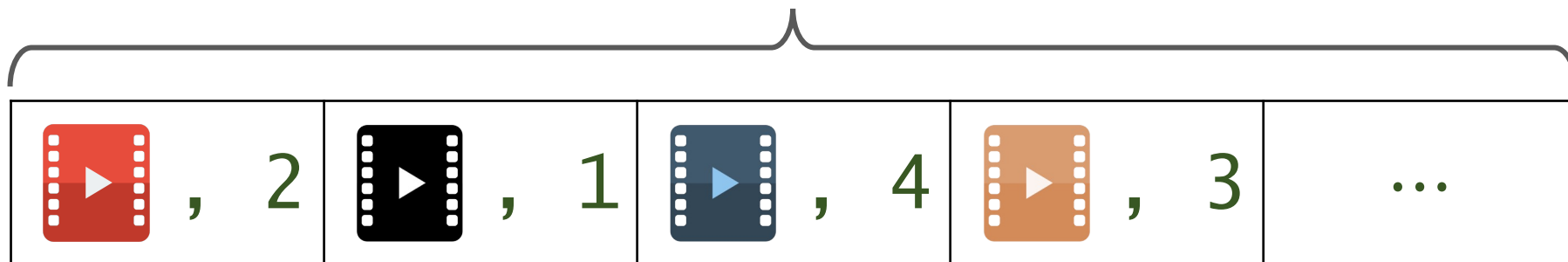
# 线上做召回

用户感兴趣的物品 (ID, 兴趣分数)

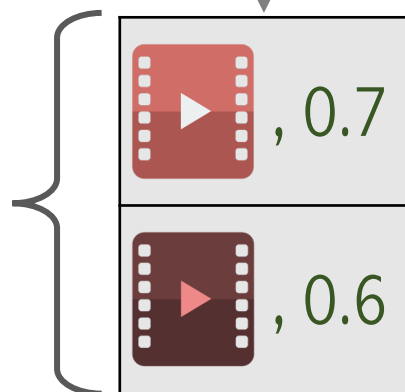


# 线上做召回

用户感兴趣的物品 (ID, 兴趣分数)

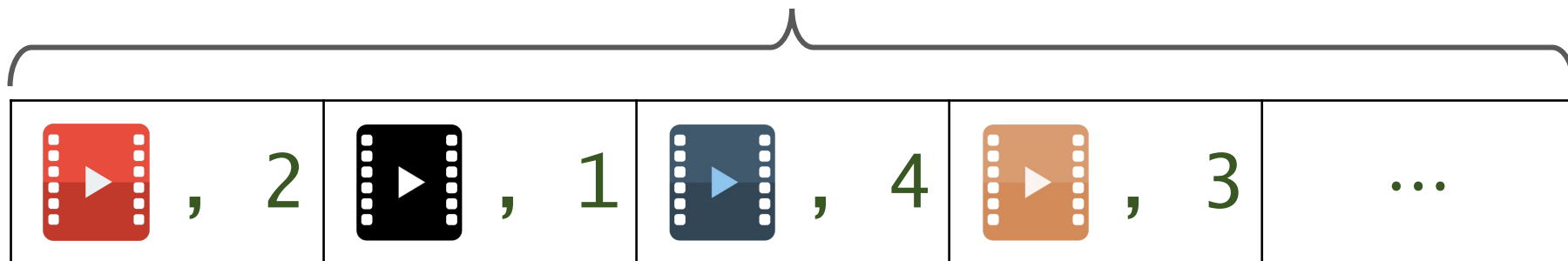


Top-k 相似



# 线上做召回

用户感兴趣的物品 (ID, 兴趣分数)

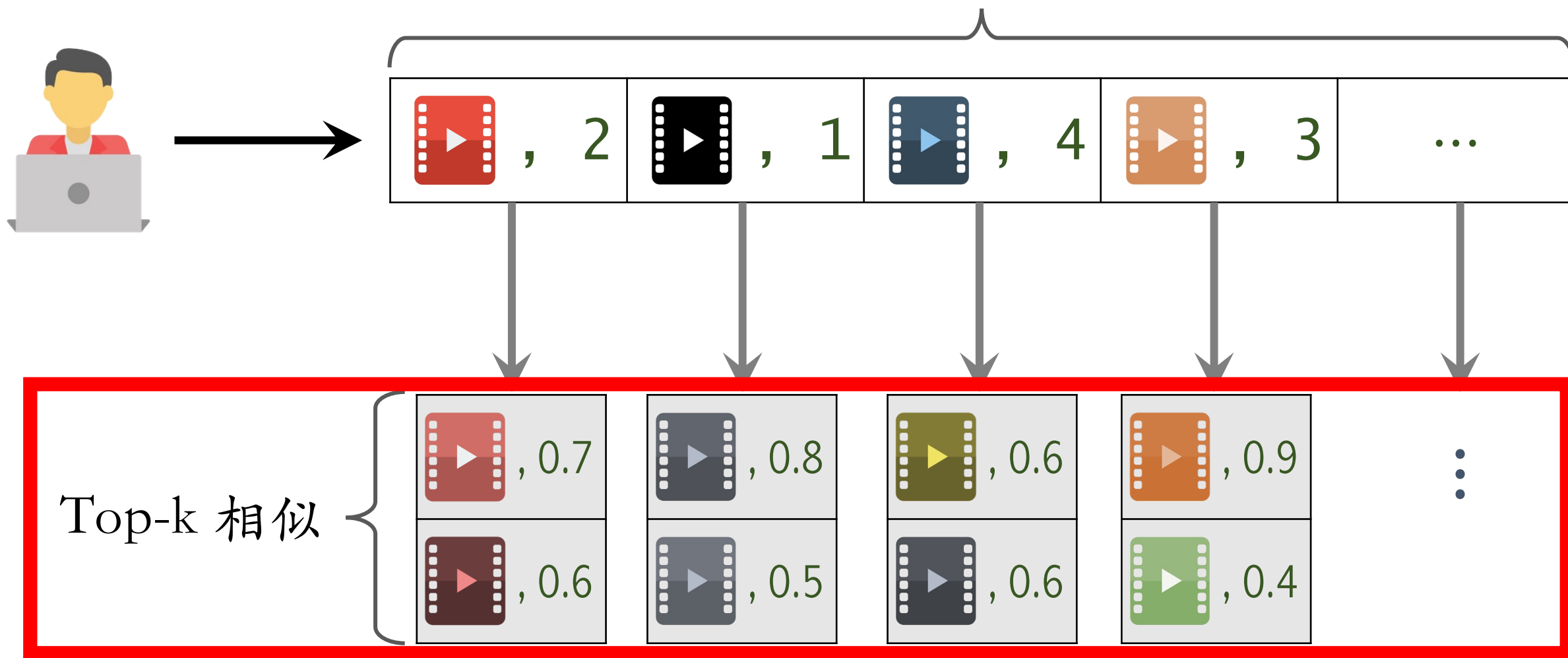


Top-k 相似



# 线上做召回

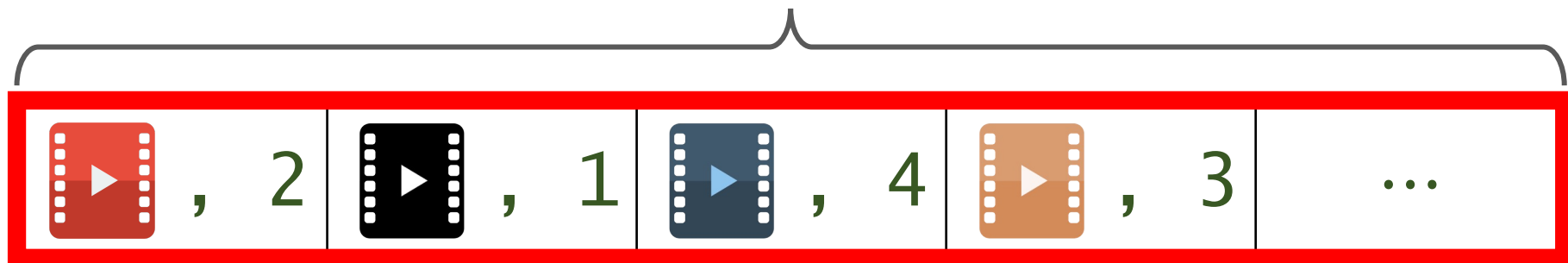
用户感兴趣的物品 (ID, 兴趣分数)



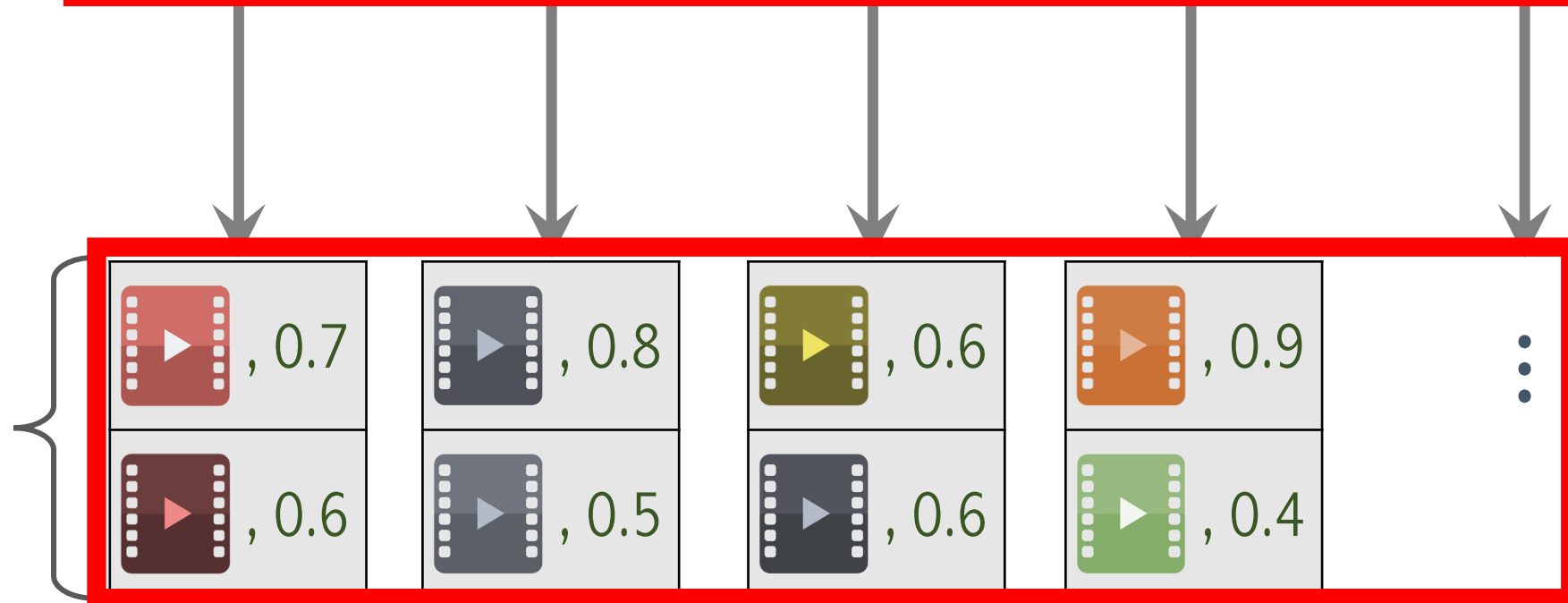


# 线上做召回

用户感兴趣的物品 (ID, 兴趣分数)



Top-k 相似



# 总结

# ItemCF的原理

- 用户喜欢物品  $i_1$ ，那么用户喜欢与物品  $i_1$  相似的物品  $i_2$ 。
- 物品相似度：
  - 如果喜欢  $i_1$ 、 $i_2$  的用户有很大的重叠，那么  $i_1$  与  $i_2$  相似。
  - 公式：
$$\text{sim}(i_1, i_2) = \frac{|\mathcal{W}_1 \cap \mathcal{W}_2|}{\sqrt{|\mathcal{W}_1| \cdot |\mathcal{W}_2|}}$$
。

# ItemCF召回通道

- 维护两个索引：
  - 用户 $\rightarrow$ 物品列表：用户最近交互过的  $n$  个物品。
  - 物品 $\rightarrow$ 物品列表：相似度最高的  $k$  个物品。
- 线上做召回：
  - 利用两个索引，每次取回  $nk$  个物品。
  - 预估用户对每个物品的兴趣分数：

$$\sum_j like(user, item_j) \times sim(item_j, item).$$

- 返回分数最高的100个物品，作为召回结果。