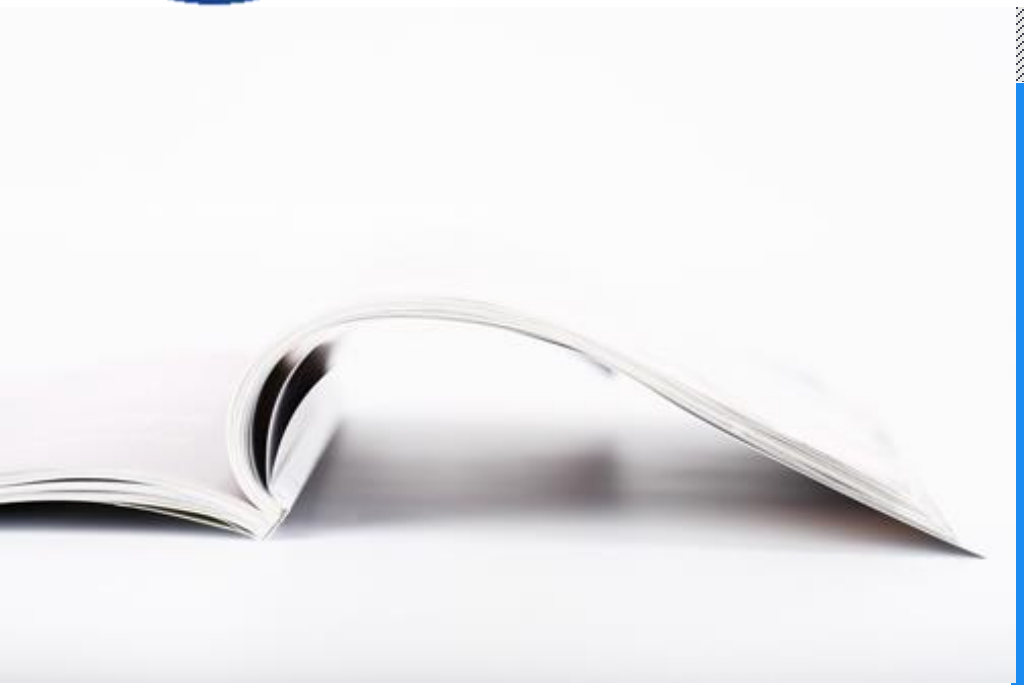


Web信息处理与应用

第八节

个性化检索（上）

徐童

2022.10.24

- **什么是个性化检索**

- 在先前的课程中，我们从千人一面的角度，介绍了如何表征和排序文档
- 然而，用户如今更期待更为个性化、精准化的信息服务
- 顾名思义，个性化检索是指考虑了用户的区别，利用用户的个性信息对检索结果进行修改或者过滤处理，以减轻用户的检索复杂度
  - 当然，大家更熟悉的概念是“推荐系统”




- **为什么我们需要提供个性化服务**

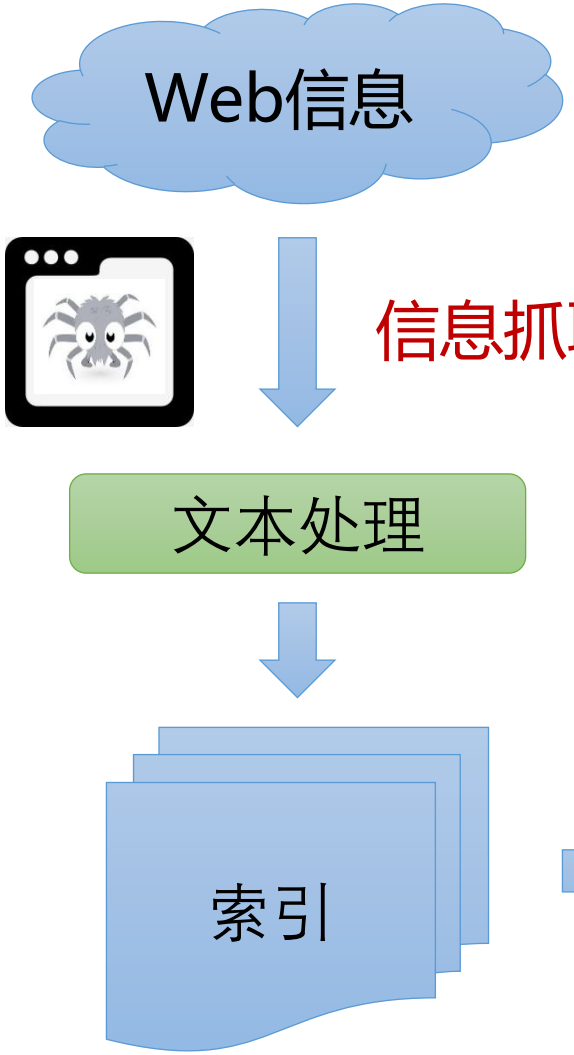
信息过载 + 用户千人千面，精准信息服务已成为大势所趋

- **为什么我们需要提供个性化服务**

- 个性化服务的诞生，背景是信息从稀缺到过剩的发展趋势
  - Web的发展，带来了无限趋近于零的信息/项目展示成本
    - 信息飞速增长，从稀缺到过剩
    - 过剩的信息需要进行有效的过滤

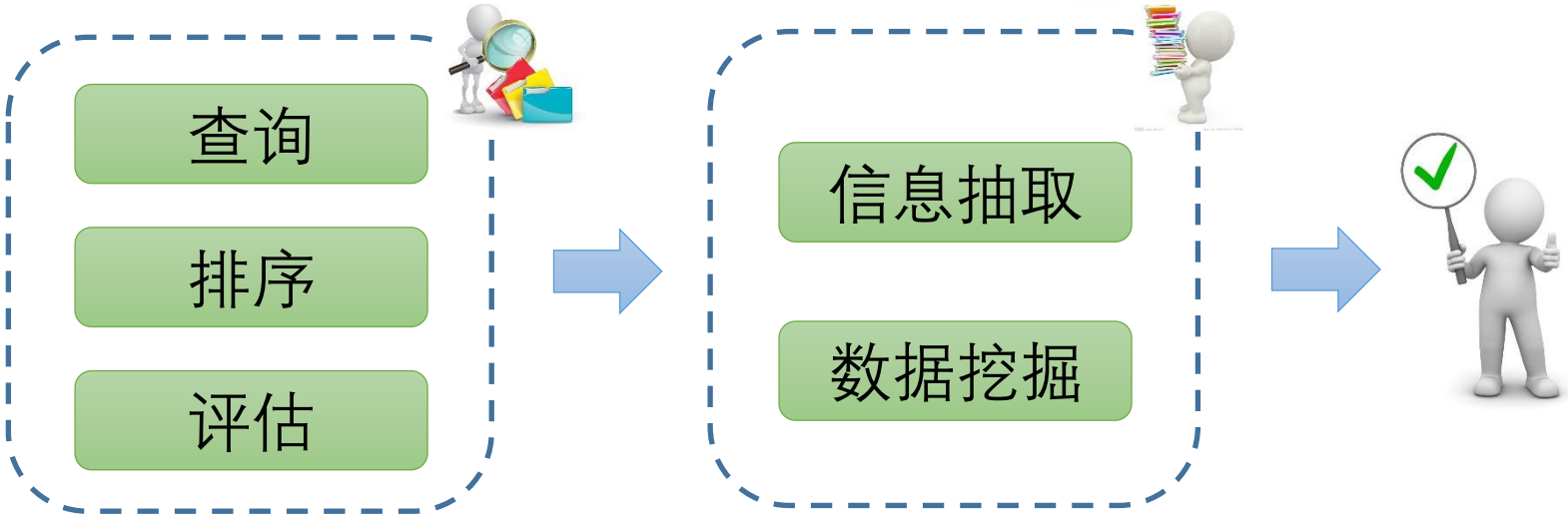
本节主题

- 本课程所要解决的问题



第八个问题：

如何更精准地提供信息服务？

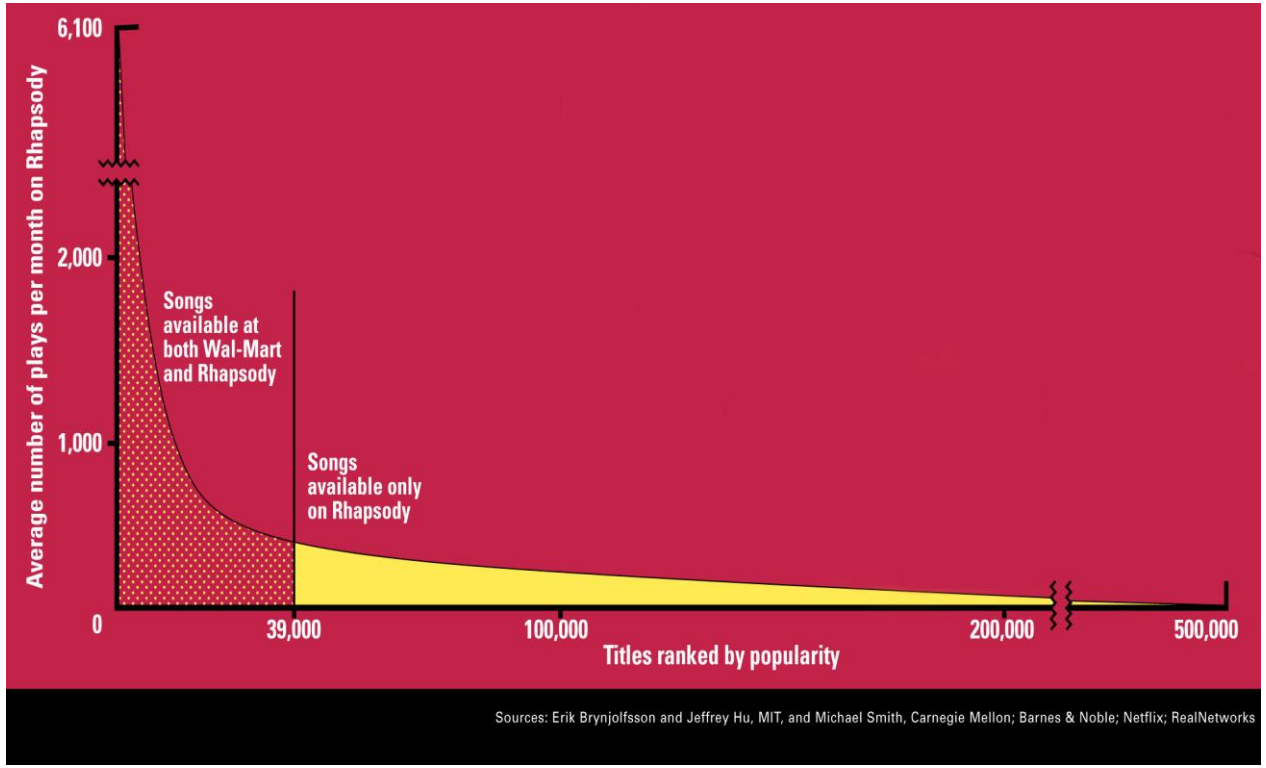


- **常见的个性化检索/推荐类型**

- 专家挑选
  - 例如，购物时常见的所谓“精选”
  - 人工挑选费时费力，且难以体现个性化因素
- 简单统计
  - 例如，“最受欢迎的XX”，“十大热门”等等
  - 简单统计能够反映一定的趋势，但受长尾现象影响
- 面向用户的个性化推荐技术

- **常见的个性化检索/推荐类型**

- 长尾（Long Tail）现象
  - 少数最热门条目所获得关注，远远高于尾部大批条目所获得的关注
  - 由于跟风现象，差距会进一步加大（马太效应）
  - 推荐的Fairness问题



- **个性化检索问题的数学定义**

- 基本要素：三个集合，一个函数（效用函数）
  - X，用户集合；S，项目集合；R，评分集合
    - R是一个由有序元素组成的集合
      - 例如，豆瓣的0-5星，[0,1]区间内的实数等
  - 基于各个集合，定义效用函数如右：$u: X \times S \rightarrow R$

- **个性化检索问题的数学定义**

- 从矩阵的角度来看，评分集合可视作一个$M \times N$维的矩阵
  - 对应M个用户，N个项目
  - 矩阵中的每个元素对应着相应的用户对项目的评分
  - 从本质上说，个性化检索/推荐问题是一个矩阵补全问题

	Avatar	LOTR	Matrix	Pirates
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

- **个性化检索/推荐系统中的关键问题**

- 个性化检索/推荐系统包含以下三个关键问题：
  1. 收集“已知”的评分信息，生成效用矩阵（R）
     - 如何收集数据？没有显式评分数据的情况如何处理？
  2. 基于已知评分，推断未知评分
     - 着重关注高分元素：我们更关注你喜欢什么
  3. 推荐结果的评估（指标部分已经介绍过了）
     - 如何有效评估个性化算法的效果，并比较不同算法？

- **个性化检索/推荐系统中的关键问题**

- 关键问题（1）收集评分数据
  - 显式（Explicit）数据
    - 可直接获得的用户评价数据，如果来源可靠则效果较好，但往往难以获得
    - 往往通过用户标注的方式获得，例如众包（Crowdsourcing）
  - 隐式（Implicit）数据
    - 基于用户行为等信息，间接地判断用户的倾向性
      - 例如，在先前介绍的间接相关性反馈方式
    - 显著的问题：如何保障负样本的质量？

- **个性化检索/推荐系统中的关键问题**

- 关键问题（2）推断未知评分
  - 存在挑战：效用矩阵R的严重稀疏性
    - 用户的行为是有限的，而且活跃用户极其稀少
    - 冷启动问题：
      - 新用户没有行为历史，难以判断偏好
      - 新项目没有评分记录，难以作为参考

- **个性化检索/推荐系统中的关键问题**

- 关键问题（3）评估个性化算法效果
  - 效果的评估，取决于问题的定义
    - 基本定义方式：用户评分预测，即预测指定用户对指定项目的打分
    - 相应的评估方法：常用均方根误差（Root-mean-square error，RMSE）
      - $\sqrt{\sum_{xi}\left(r_{xi}-r_{xi}^{*}\right)^{2}}$，其中 $\boldsymbol{r_{xi}}$ 为预测评分，$\boldsymbol{r_{xi}^{*}}$ 为真实评分

- **个性化检索/推荐系统中的关键问题**

- 关键问题（3）评估个性化算法效果
  - 效果的评估，取决于对于问题的定义
    - 其他方式（1）：视作分类问题，即推荐正确/错误
      - 一种情况为0/1效用矩阵，或基于某个阈值进行推荐
        - 此时为完全二分类，可通过Precision/Recall/F值等评估
      - 另一种情况为排序后返回Top N结果
        - 此时可通过Pre@N，Rec@N等评估，注意理论上限可能不为1

- **个性化检索/推荐系统中的关键问题**

- 关键问题（3）评估个性化算法效果
  - 效果的评估，取决于对于问题的定义
    - 其他方式（2）：视作排序问题，即推荐项目的相关性程度
    - 用户对于不同项目的接受/喜爱程度不同
      - 可基于预测评分对项目进行排序，然后采用排序评估方式进行评估
        - 参见“评估”部分有关指标的介绍

- **个性化检索/推荐系统中的关键问题**

- 关键问题（3）评估个性化算法效果
  - 如果只关注推荐的准确性，则可能受到一定的误导
    - 推荐的顺序值得研究：例如，音乐的播放序列
  - 在现实中，我们更关注获得高分的预测内容
    - 与搜索的相似性?
    - 然而，RMSE可能对于在高分段表现出色，其他部分表现不佳的方法加以惩罚（没有focus到用户喜好)

- **个性化检索/推荐系统中的关键问题**

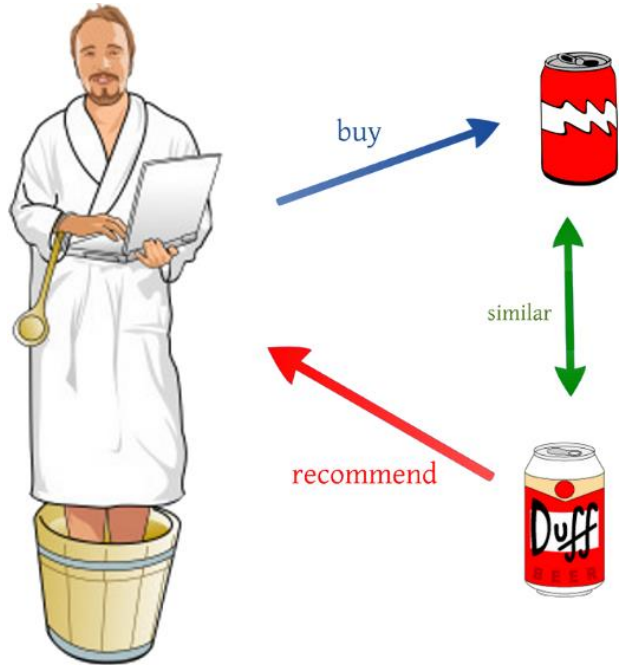
- 关键问题（3）评估个性化算法效果
  - 推荐的多样性同样重要：用户期待一些Surprise
  - 警惕推荐的“茧房效应”！

本节目录

- **基于内容的推荐**
- 基于协同过滤的推荐
  - 基于内存（Memory-based）
  - 基于模型（Model-based）
  - 应用示例：Netflix Prize

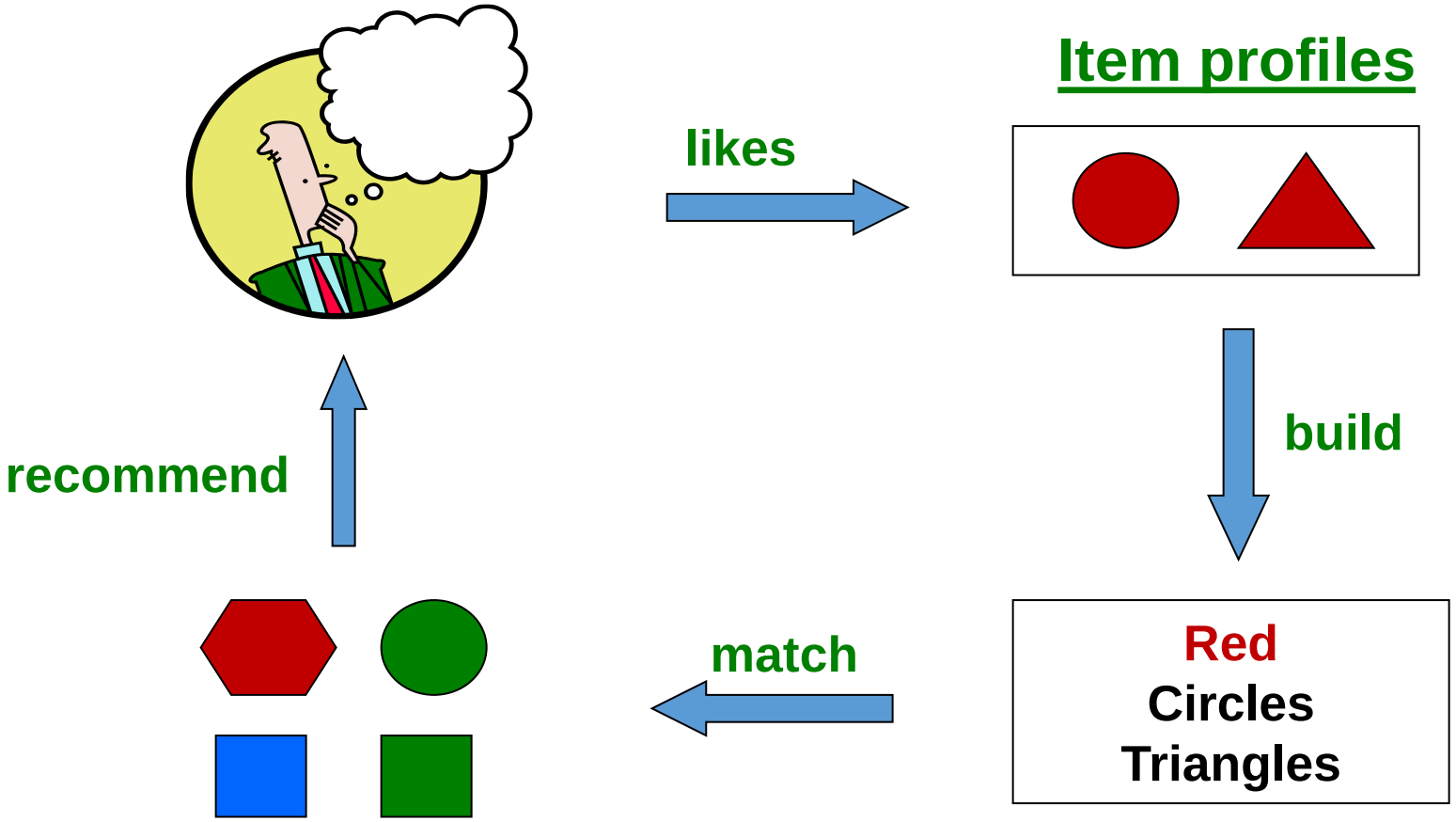
- **基于内容的推荐：核心思想**

- 基本想法：用户的偏好一般相对稳定
  - 因此，给用户推荐他/她以前喜欢的项目准没错（偏保守的想法）
    - 例如：电影推荐
      - 推荐同演员、同导演、同主题……
    - 例如：新闻推荐
      - 推荐类似主题/倾向性的文章
        - 可能造成局限性和错觉



- **基于内容的推荐：基本流程**

- 基于内容推荐的基本流程



- **基于内容的推荐：项目画像**

- 为实现基于内容的推荐，对于每个备选项目，需要给出相应的画像（Profile）
  - 更一般的形式：画像往往以向量的格式存在
    - 例如：电影推荐
      - 可以将元数据，如标题、类型、演员、导演等抽象为向量
    - 例如：新闻推荐
      - 可以从文章中抽取关键词，或采用类似tf-idf、主题模型等方法

- **基于内容的推荐：用户画像与评分预测**

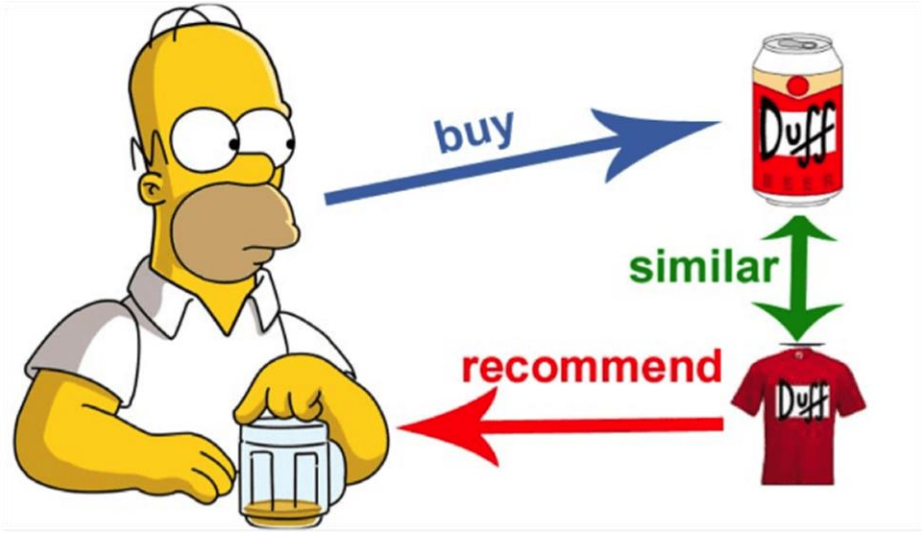
- 相应的，用户的画像可由他/她曾经评分过的项目画像所估计
  - 一般采用加权平均的方式得到用户画像向量（基于评分进行加权）
- 基于用户与项目画像，可采用相似性度量进行评分
  - 一般采用两个向量之间的余弦相似度（Cosine Similarity）
    - $u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$
    - 也可根据实际情况选择其他度量方式

- **基于内容的推荐：优点**

- 每个人的推荐过程相互独立，不需要其他用户的数据
- 可以为具有独特偏好的用户进行有效推荐
  - 不受大众倾向性和热度的影响
- 可以推荐新项目或非热门项目
- 推荐结果有着较好的可解释性
  - 可列举内容特征作为推荐的依据

- **基于内容的推荐：缺点**

- 找到合适的特征是一件困难的事
  - 对于非结构化信息，如图像、视频、音频等尤其如此
  - 部分特征的提取可能存在误导性（归因错误）



基于内容的推荐：缺点

- 给新用户推荐项目，永远是一个困难的任务
  - 如何建立新用户的用户画像?
- 过度特化（Overspecialization）现象
  - 永远只能给用户推荐局限于其画像中的内容——信息茧房问题
  - 用户的多方面兴趣难以体现
  - 难以通过他人的评价对推荐结果进行评估

- **衍生问题：如何解决多样化问题**

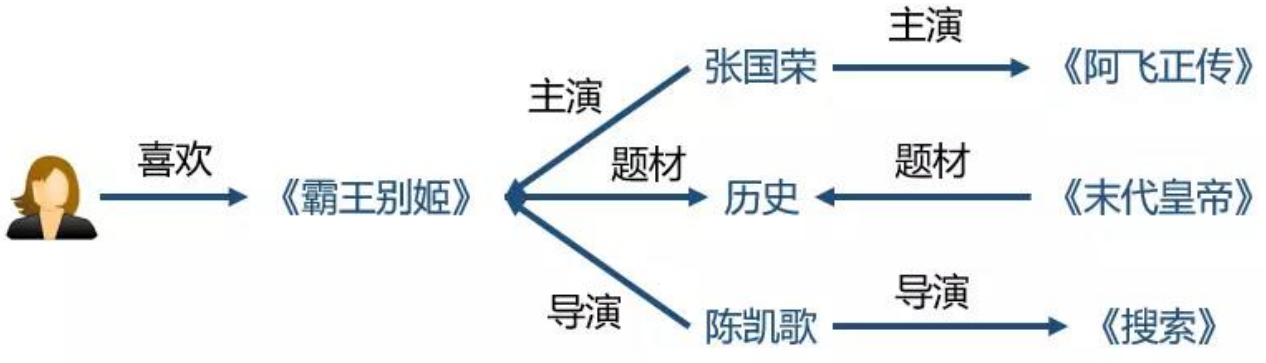
- 主题过于集中的推荐结果可能会影响用户体验，需要适度区分
- 解决方案：在准确度评估的同时，引入多样性评估
  - 复习一下：最大边界相关性（Maximal Marginal Relevance，MMR）

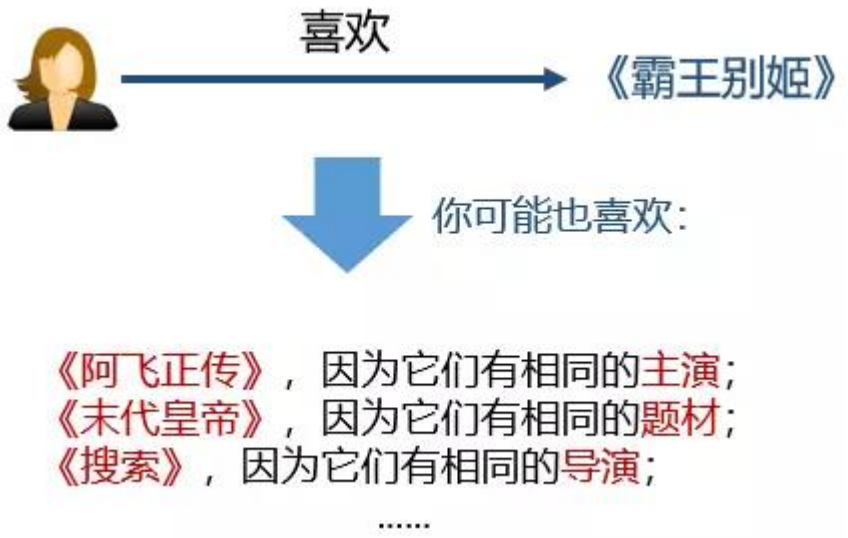
$$MMR^{def} = Argmax_{d_i \in R|S}[\lambda P(d_i \left| q) - (1-\lambda) max_{dj \in S} P(d_i | d_j)\right]$$

  - 其他方案可见 “评估”中有关多样化评价的部分

- **衍生问题：从基于内容推荐到基于路径推荐**

- 采用更为结构化的知识图谱代替向量化的画像方式
- 基于图谱上的游走实现推荐，路径可作为推荐的依据
  - 具体将在本课程第十五讲“知识图谱应用”部分加以介绍





- **衍生问题：推荐中的偏见（Bias）问题**

- 用户的反馈不一定精准反映用户的真实偏好，而可能受到各种因素干扰
  - 右图展示了多种不同的Bias来源
    - 位置效应：第一更受关注
    - 模态效应：与众不同的模态更吸引关注
    - 关键词效应：标题党的威力
  - 在建模用户画像时，需要设计模型描述Bias干扰，并还原用户真实兴趣

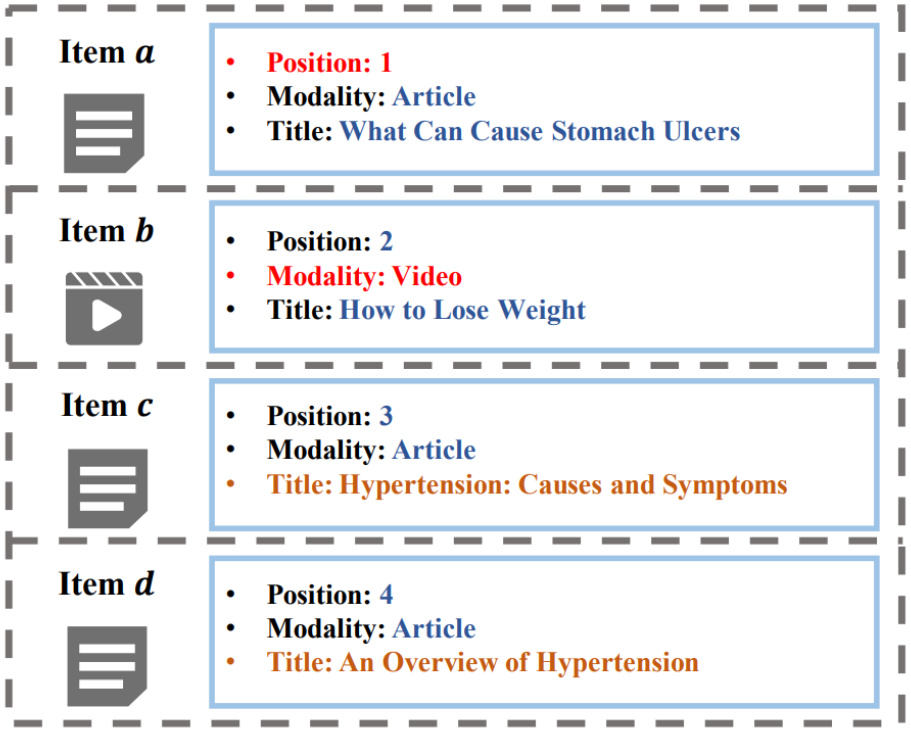
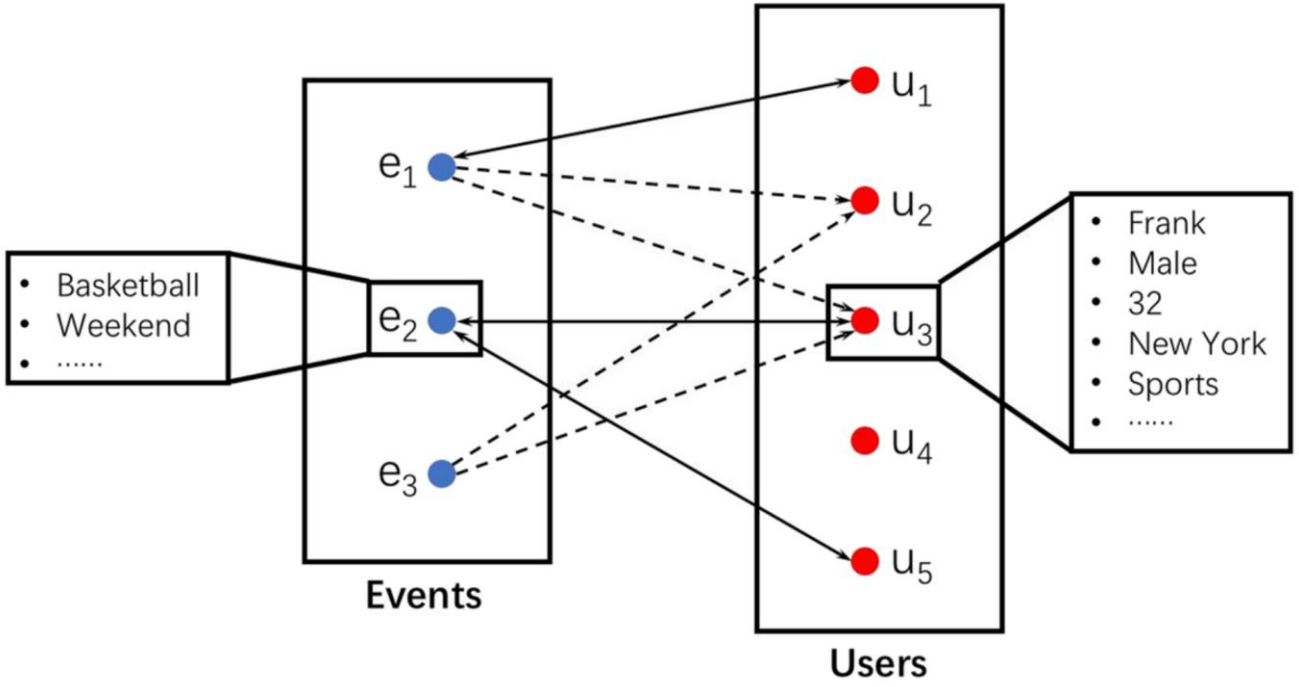


Figure 1: An example for context bias.

- **衍生问题：双向选择问题**

- 如果被选择对象存在名额限制，用户的偏好不一定能够得到满足
  - 此时，采用双向选择描述推荐过程更为合理
    - 用户在选择项目，同时项目也在“选择”用户
  - 此时，最后的行为实际上反应了稳定匹配的结果
    - 2012年，稳定匹配理论获得诺贝尔经济学奖



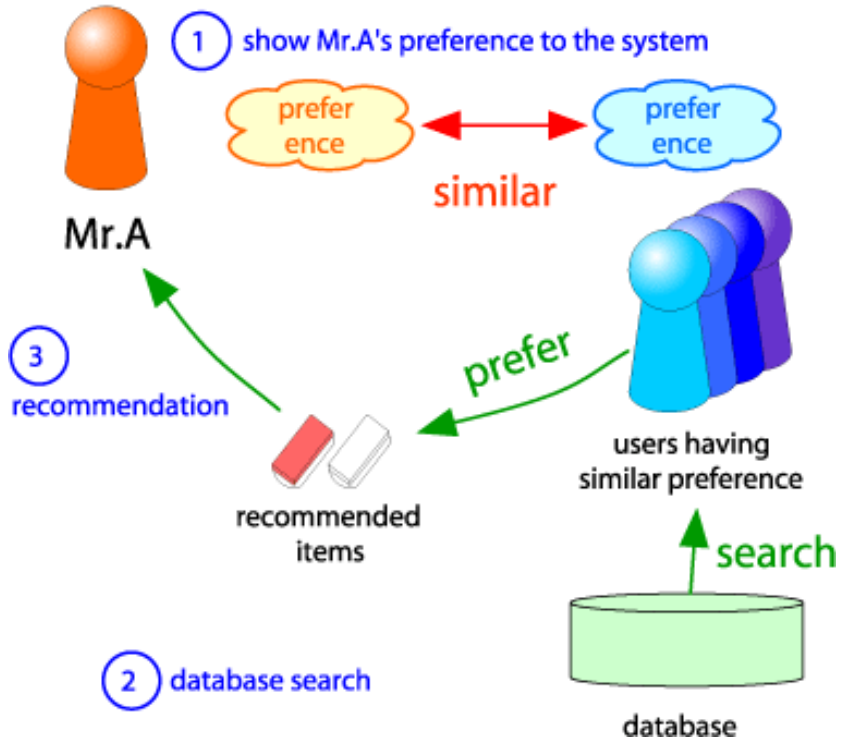
本节目录

- 基于内容的推荐
- **基于协同过滤的推荐**
  - 基于内存（Memory-based）
  - 基于模型（Model-based）
  - 应用示例：Netflix Prize

- **协同过滤的基本思想**

- 基于内容的推荐方案只基于单一用户记录向该用户进行推荐

- 在实际应用中，我们发现，其他用户的浏览行为对当前用户有借鉴作用
  - 例如，你和你的狐朋狗友们往往具有相似的口味和行为
  - 相应的，这种相似的口味导致了在评分上的相似性



- **协同过滤的基本思想**

- 如前所述，推荐系统的本质就是矩阵补全问题
- 相应的，协同过滤的思想在于基于矩阵的其他行，协助填补本行的空缺

Target User	Item 1	Item 2	Item 3	Item 4	Item 5	Average
Alice	5	3	4	4	???	16/4
User1	3	1	2	3	3	9/4
User2	4	3	4	3	5	14/4
User3	3	3	1	5	4	12/4
User4	1	5	5	2	1	13/4

协同过滤的基本类别

- 大体上，可将协同过滤技术分为以下两大类
  - 基于内存（Memory-based）的协同过滤
    - 基于现有数据与简单度量运算进行推荐
    - 可进一步细分为基于用户（User-based）与基于项目（Item-based）
  - 基于模型（Model-based）的协同过滤
    - 基于现有数据训练模型，通过模型进行推荐
    - 代表性方法如矩阵分解（Matrix Factorization）、深度学习等

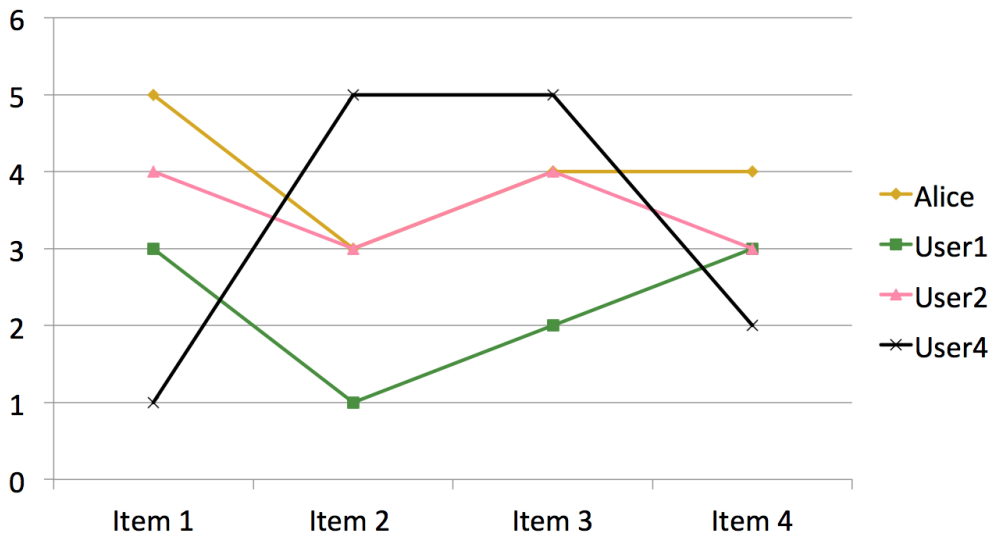
本节目录

- 基于内容的推荐
- **基于协同过滤的推荐**
  - **基于内存（Memory-based）**
  - 基于模型（Model-based）
  - 应用示例：Netflix Prize

- **基于用户推荐（User-based CF）**

- 如前所述，具有相似偏好的用户，往往在过去与未来的评分行为上相似

- 基于用户（User-based）推荐的目的，即在于找到这些相似用户，并基于这些用户的历史行为进行推荐
  - 相似用户往往被称作“邻居”，类似于寻找最近邻的思想



- **最近邻的寻找过程**

- 如前所述，具有相似偏好的用户，往往在过去与未来的评分行为上相似
- 相应的，寻找最近邻的依据，应该从用户过去的评分行为上着手
  - 历史评分行为越相似，用户之间未来行为的相似性就越高
  - 基于共同评分的项目，衡量用户之间的相似性如下：

$$sim(a,b) = \frac{\sum_{p \in product(P)} (r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in product(P)} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in product(P)} (r_{b,p} - \bar{r}_b)^2}}$$

Average rating of user ***b***

计算相似度时，如果某个user对某个item未评分，则对应的r直接设为0，不用减去平均分

- **基于用户推荐的评分预测**

- 在得到用户之间的相似性后，针对待预测的项目，可以根据历史上其他用户对于该项目的评分，结合用户之间的相似性作为加权，预测评分结果
  - 相似性计算与评分预测中，都通过减去平均值来抹去个人评分偏好的影响
    - 不同用户打分范围不同，有些人倾向于打高分，有些人更苛刻

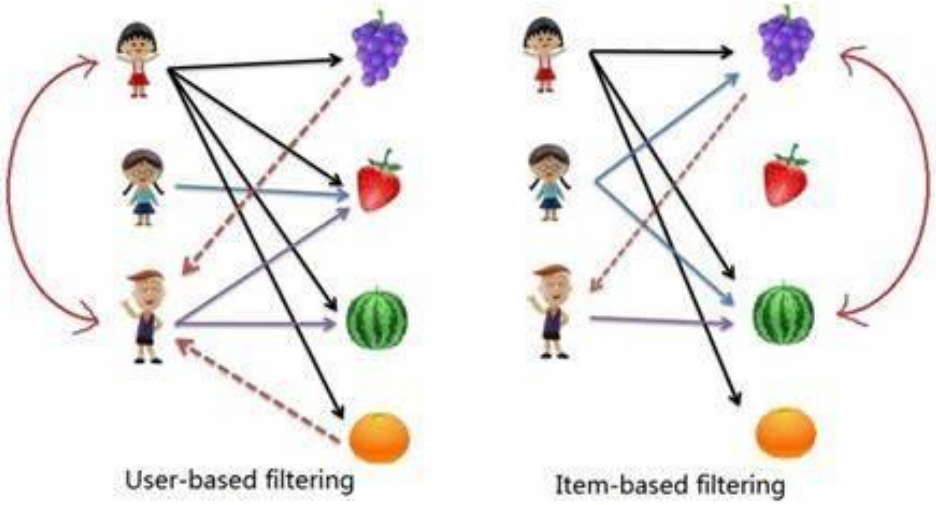
$$pred(a,p) = \bar{r}_a + \frac{\sum_{b \in neighbors(n)} sim(a,b) \cdot (r_{b,p} - \bar{r}_b)}{\sum_{b \in neighbors(n)} sim(a,b)}$$

- **基于用户推荐的评分预测**

- 有些时候，我们不需要考虑所有邻居的评分，而只考虑K-最近邻的情况
  - 相应的，前页公式中的neighbor(n)即缩小为 K-最近邻 的集合
  - 几点需要注意的情况：
    - 只考虑对指定Item有评分的邻居，无评分的跳过（即使Similarity较高）
    - 通常情况下，低于0的相关性可以被忽略
      - 因此，K-最近邻集合可能实际上小于K个邻居
    - 基于项目（Item-based）的推荐采用类似的设定

- **另一个角度：基于项目推荐**

- 与用户行为相似性类似，相似的项目，往往在大众眼中的评分也比较接近
- 与基于内容的推荐不同，基于项目（Item-based）的推荐，其衡量相似项目的标准，并不是项目本身的属性，而是不同项目的评分历史
- 同一个人给两个项目打出相似分数，说明他认为两个项目相似；越多这样的人，两个项目越相似



- **另一个角度：基于项目推荐（Item-based CF）**

- 基于项目推荐的计算公式

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

计算方式与User-based的Similarity相似

  - 其中，相似度计算与User-based类似
    - 即构造两个向量，然后计算Pearson相关系数
    - 此外，同样可以基于平均分对分数估计进行修正
      - 如果未评分，则直接设为0，不用减去平均数
      - 思考：最终的r_{ix}为何不需要平均分修正？因为只针对该用户自己的评分算加权和

- **另一个角度：基于项目推荐**

- 基于项目推荐的实例

users

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

movies

- **另一个角度：基于项目推荐**

- 基于项目推荐的实例：只考虑 2 -近邻

users

movies

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

sim(1,m)
1.00
-0.18
0.41
-0.10
-0.31
0.59

Movie 1向量计算实例

求打分均值：

$m_1 = (1+3+5+5+4)/5 = 3.6$

去中心化：

movie1: *[-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]*

基于向量的Pearson系数计算两两相似度

- **另一个角度：基于项目推荐**

- 基于项目推荐的实例：只考虑 2 -近邻

users

movies	12	11	10	9	8	7	6	5	4	3	2	1		sim(1,m)
		4		5			5	2.6		3		1	1	1.00
	3	1	2			4			4	5			2	-0.18
		5	3	4		3		2	1		4	2	3	0.41
		2			4			5		4	2		4	-0.10
	5	2					2	4	3	4			5	-0.31
		4			2			3		3		1	6	0.59

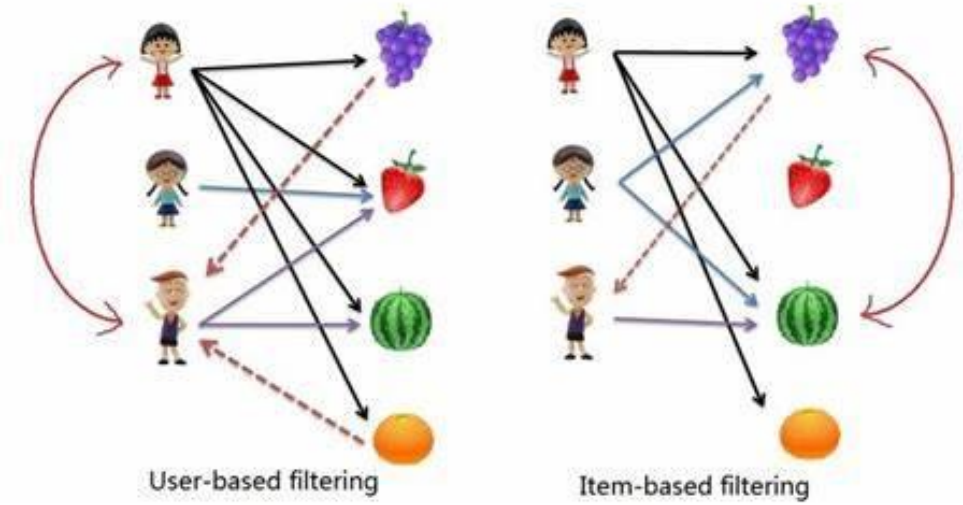
$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$

(0.41*2 + 0.59*3) /

(0.41+0.59) = 2.6

- **两类方法的对比**

- 在实践中，往往基于项目的推荐效果更好，为什么？
  - 项目的属性相对单一，而用户的偏好则更为丰富多样
  - 相应的，某样项目受欢迎的理由相对固定，而用户可能在不同情境下体现出不同的偏好
    - 在你没有办法区分这些不同情境时，得到的用户偏好反而是不准的



- **基于内存的推荐的优缺点**

- 优点：可适用于任意种类项目，效果较好
  - 不受多模态、非结构化信息表征与特征选取的困扰
- 缺点：冷启动、稀疏性、热度偏差等
  - 冷启动：用户与项目都存在冷启动问题
  - 稀疏性：用户评分记录严重稀疏，很难找到评价过同一项目的用户
  - 热度偏差：更倾向于推荐热门项目，对具有独特偏好的用户推荐效果差
    - 寻找相似用户/项目时，小众偏好很容易被热门偏好所淹没

- **衍生问题（1）用户冷启动问题**

- 给新用户推荐项目，永远是一个困难的任务
- 一般而言，面向新用户的推荐通常采用如下方案
  1. 提供非个性化的推荐，直至收集足够多的个性化数据
  2. 借助用户的其他信息（如个人信息、其他网站记录）
     - 可能存在侵犯用户隐私的问题！
  3. 诱导式推荐：选取少量具有代表性和多样性的项目进行推荐
     - 通过迭代式收集用户反馈，快速获得近似用户画像

- **衍生问题（2）项目冷启动问题**

- 采用混合方法解决Item维度的冷启动问题
  - 例如，引入基于内容的方法，通过项目的元信息得到初步推荐结果
- 除此之外，一些边信息（Side Information）也有助于补充更多依据
  - 例如，当项目缺乏元信息时，可采用众包文本加以补充
    - 如媒体文件标注时，可采用评论信息进行补充
  - 又如，引入知识图谱进行推荐
    - 即前述有关基于知识图谱的路径推荐相关内容

本节目录

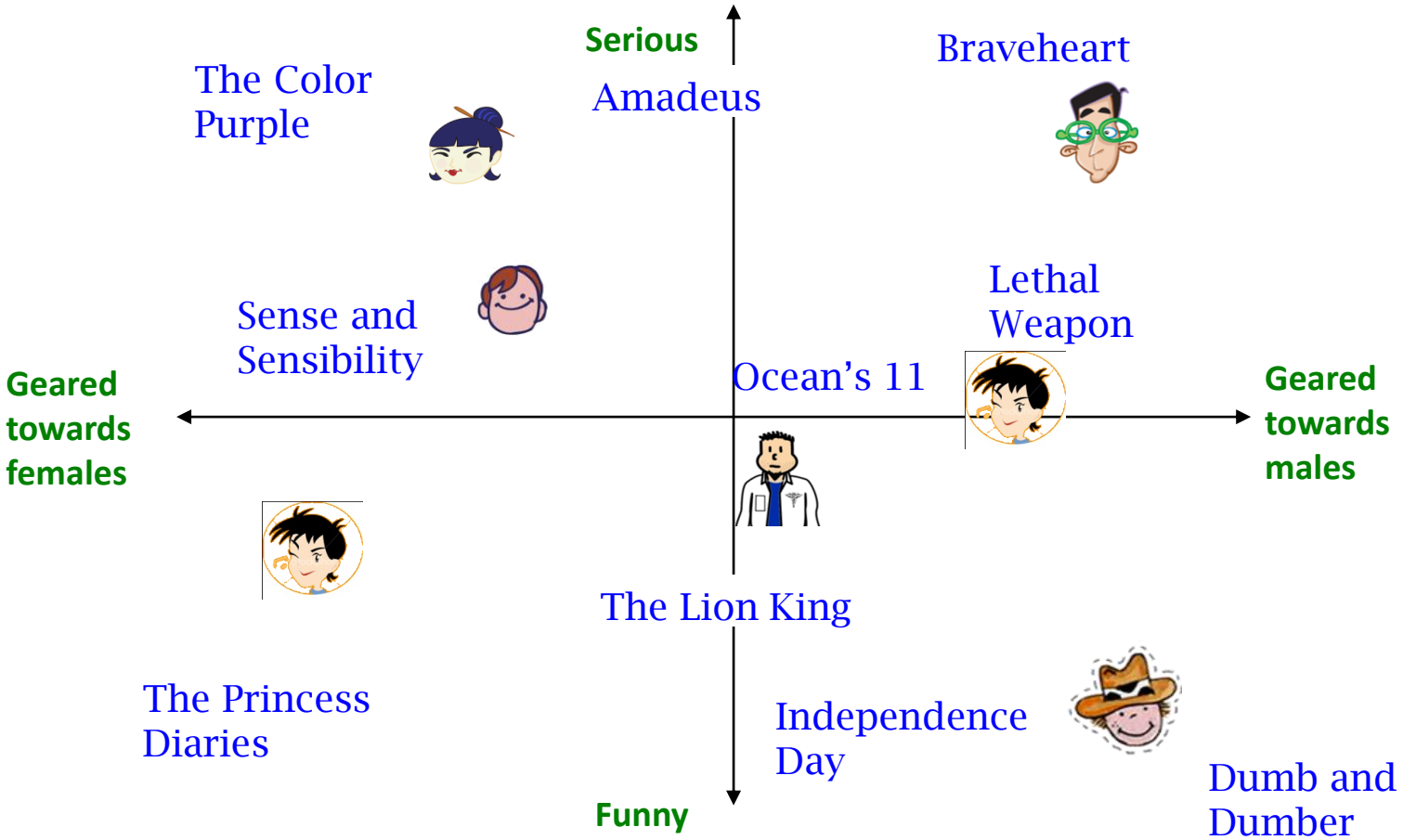
- 基于内容的推荐
- **基于协同过滤的推荐**
  - 基于内存（Memory-based）
  - **基于模型（Model-based）**
  - 应用示例：Netflix Prize

基于模型的推荐：基本思路

- 基于内存的推荐技术，仅对数据进行简单处理，适用于各种数据
- 然而，数据的稀疏性、计算最近邻的高复杂度，限制了其有效性
- 与此同时，我们知道，从效用矩阵的视角来看，推荐系统的本质是矩阵补全
- 那么，矩阵的各个元素是如何生成的？
  - 基本思路：用户对项目的评分，本质上是用户的偏好，与项目的属性之间的相似度。相似度越高，评分越高
  - 那么，用户的偏好与项目的属性，如何表示？

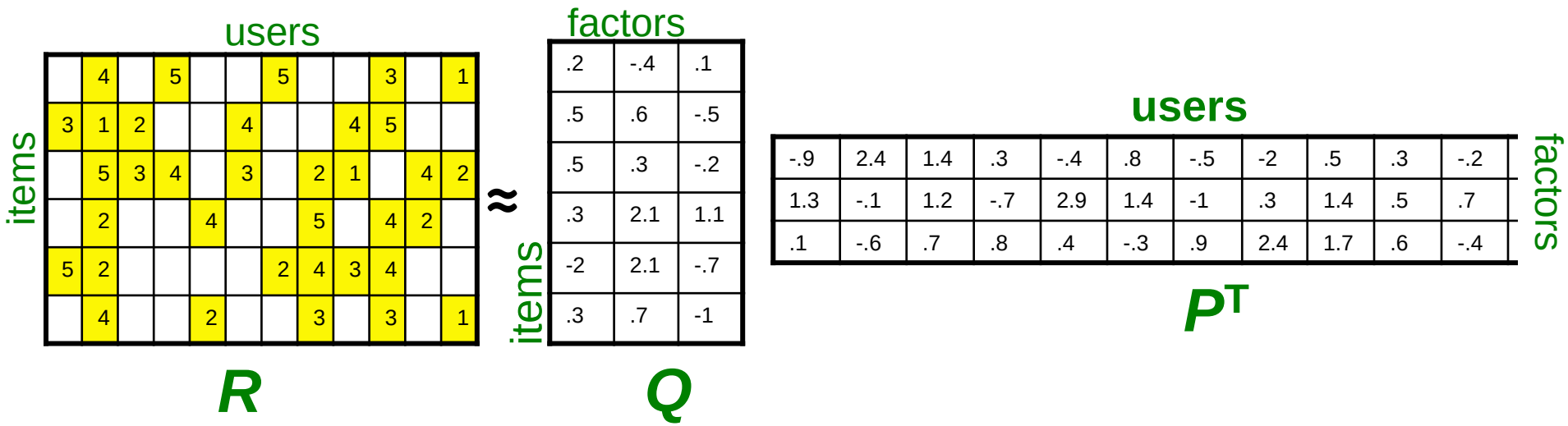
- **基于模型的推荐：潜在因子**

- 事实上，用户偏好与项目属性，可以采用同样的归约方式进行精简。如何实现？



• 基于模型的推荐：基本形式

- 借鉴矩阵分解（Matrix Factorization）的思路，揭示潜在因子



	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

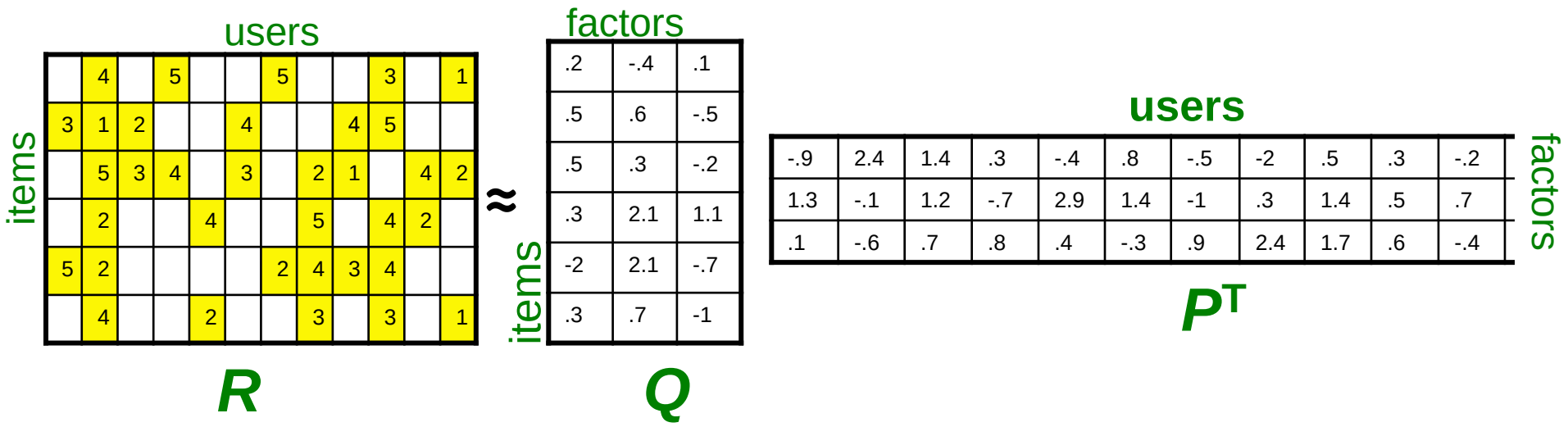
.2	-.4	.1
.5	.6	-.5
.5	.3	-.2
.3	2.1	1.1
-2	2.1	-.7
.3	.7	-1

-.9	2.4	1.4	.3	-.4	.8	-.5	-2	.5	.3	-.2
1.3	-.1	1.2	-.7	2.9	1.4	-1	.3	1.4	.5	.7
.1	-.6	.7	.8	.4	-.3	.9	2.4	1.7	.6	-.4

- 评分矩阵R被近似视作项目属性矩阵Q与用户偏好矩阵P的乘积
- P与Q的维度，一方面与用户/项目的数量有关，另一方面体现了潜在因子的数量
  - 开放问题：Latent Factor的维度如何确定？

- **基于模型的推荐：评分预测**

- 借鉴矩阵分解（Matrix Factorization）的思路，揭示潜在因子



	4		5			5			3		1
3	1	2			4			4	5		
	5	3	4		3		2	1		4	2
	2			4			5		4	2	
5	2					2	4	3	4		
	4			2			3		3		1

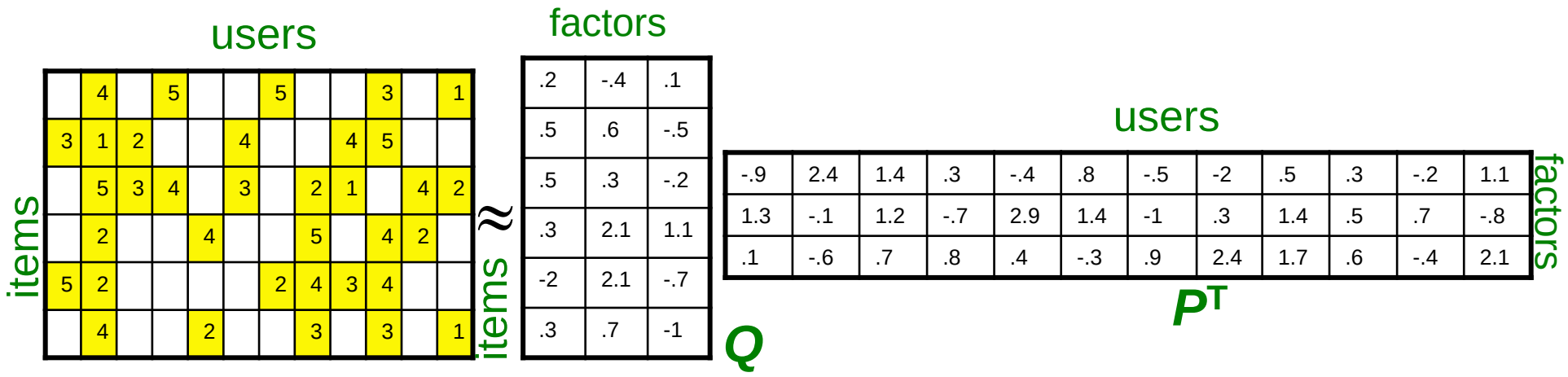
.2	-.4	.1
.5	.6	-.5
.5	.3	-.2
.3	2.1	1.1
-2	2.1	-.7
.3	.7	-1

-.9	2.4	1.4	.3	-.4	.8	-.5	-2	.5	.3	-.2
1.3	-.1	1.2	-.7	2.9	1.4	-1	.3	1.4	.5	.7
.1	-.6	.7	.8	.4	-.3	.9	2.4	1.7	.6	-.4

- 当用户与项目的潜在因子已知，则任何缺失的评分，均可以通过对应的P、Q矩阵相应的行列运算估计得到 $\hat{r}_{xi} = q_i \cdot p_x = \sum_k q_{ik} \cdot p_{xk}$

- **基于模型的推荐：潜在因子**

- 那么，如何估计出两个潜在因子矩阵?
  - 既然用户评分是根据潜在因子的乘积所得，那么，基于这种方法得到的用户评分，应与历史评分记录尽可能接近，即 $\min_{P,Q} \sum_{(i,x)\in R} (r_{xi} - q_i \cdot p_x)^2$



- **基于模型的推荐：过拟合与正则化**

- $\min_{P,Q}\sum_{(i,x)\in R}(r_{xi}-q_i\cdot p_x)^2$ 这一公式，其本质是一个误差平方和（SSE）。
- 因此，我们的目标正是通过优化这一SSE，以获得潜在因子的估计值
  - 然而，潜在因子的维度 K ，也是一个潜在的麻烦
    - 我们需要训练的参数一共 (M+N)*K个，随着K的增长而增长
    - 如果K足够的大，在数据稀疏的情况下，可能并没有那么多训练样本
    - 因此，过高的 K 可能导致过拟合的问题

• 基于模型的推荐：过拟合与正则化

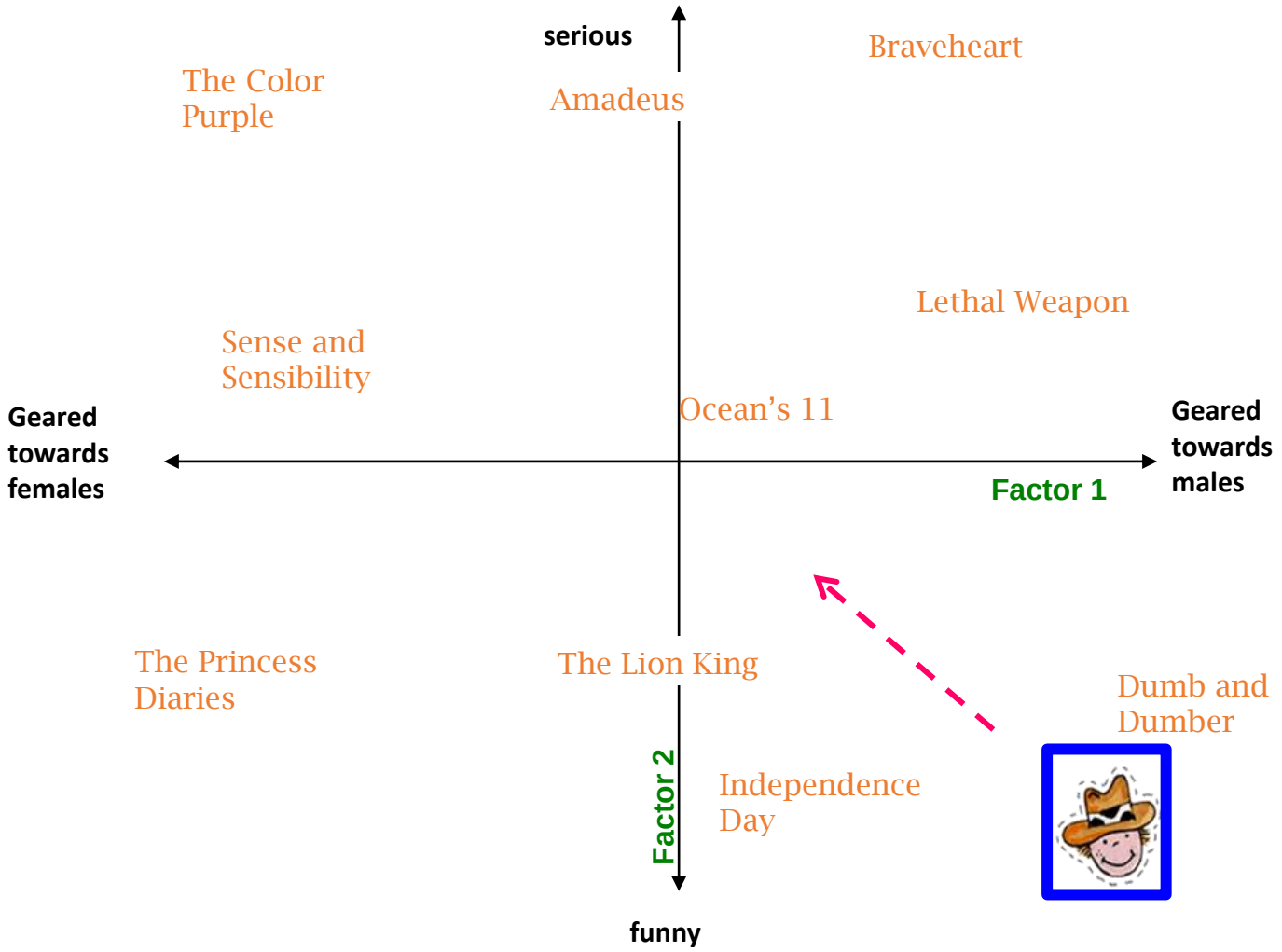
- 如何解决过高的 K 带来的过拟合问题？
  - 一种思路是引入更多训练数据，但这并非易事
  - 另一种思路则尝试通过“收缩”（Shrinkage）参数的方式来提升泛化性
    - 过拟合问题的本质是对于训练样本的过分迁就，从而影响了模型的泛化能力
    - 相应的，我们的目的在于通过控制参数数值，使其不那么“迁就”训练样本
  - 解决方法：引入正则项
    - 避免过大的参数值

$$\min_{P,Q} \underbrace{\sum_{training}(r_{xi} - q_i p_x)^2}_{\text{“error”}} + \underbrace{\left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2\right]}_{\text{“length”}}$$

有关正则与范数选取可参考论文：https://www.jstor.org/stable/2346178

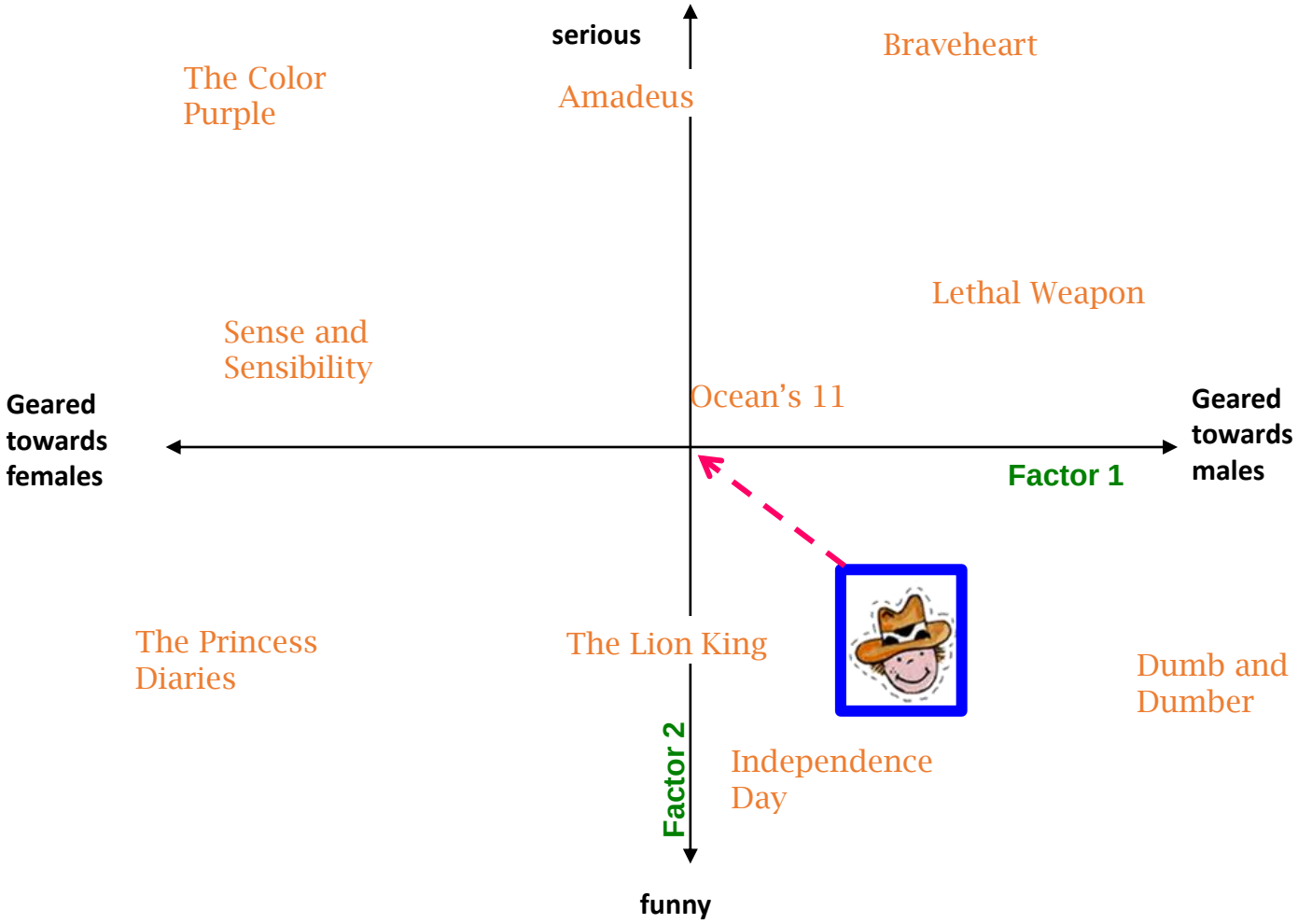
- **基于模型的推荐：过拟合与正则化**

- 图示：正则化的效果
  - 例如，图中右下角的用户偏好，可能是大量样本得到的合理结果，也可能是过拟合后的结果
  - 正则化的目的则是将参数往原点“拉”



- **基于模型的推荐：过拟合与正则化**

- 图示：正则化的效果
  - 一般而言，用户的历史行为越少，记录越稀疏，模型过拟合的风险越大
  - 相应的，正则化所起到的“回归原点”的作用也就越强



• 矩阵分解的衍生模型（1）非负矩阵分解

- 基础的矩阵分解方法存在一个问题：潜在因子矩阵元素可能为负
- 然而，部分情况下，矩阵元素为负是不符合要求的
  - 例如，文档属性在各个主题上有强弱之分，但不会出现负分布
- 因此，需要对求解过程添加非负约束，以保障元素的非负性
  - 非负矩阵分解
    - Non-negative MF

$$\mathbf{R} \approx \mathbf{P}^T\mathbf{Q}$$

$$\text{s.t.} \quad \mathbf{P} \geq 0$$
$$\mathbf{Q} \geq 0$$

Lee D D, Seung H S. Algorithms for Non-negative Matrix Factorization, NIPS 2000

• 矩阵分解的衍生模型（1）非负矩阵分解

- 如何在求解过程中保证元素的非负性？利用梯度下降法优化
  - 已知，我们引入噪声矩阵 E = R – PQ，则目标在于最小化噪声矩阵
    - 如果考虑优化噪声所对应的2-范数，通过梯度下降法，可得迭代式：
    - $p_{ik} \leftarrow p_{ik} + \alpha_1[(R - PQ)Q^T]_{ik}$
    - $q_{kj} \leftarrow q_{kj} + \alpha_2[P^T(R - PQ)]_{kj}$
      - 其中中括号部分为对矩阵元素的偏导，前面+为两个负号合并

Lee D D, Seung H S. Algorithms for Non-negative Matrix Factorization, NIPS 2000

- **矩阵分解的衍生模型（1）非负矩阵分解**

- 如何在求解过程中保证元素的非负性？利用梯度下降法优化
  - 这个时候的问题在于如何选择合适的α，使得迭代中始终保持非负
    - 如果取$\alpha_1 = \frac{p_{ik}}{[PQQ^T]_{ik}}$，而$\alpha_2 = \frac{q_{kj}}{[P^TPQ]_{kj}}$
    - 则迭代更新过程变为$p_{ik} \leftarrow p_{ik} \frac{[RQ^T]_{ik}}{[PQQ^T]_{ik}}$，$q_{kj} \leftarrow q_{kj} \frac{[P^TR]_{kj}}{[P^TPQ]_{kj}}$
      - 由于R矩阵非负，在P、Q初始值非负的情况下，迭代一直非负

Lee D D, Seung H S. Algorithms for Non-negative Matrix Factorization, NIPS 2000

• 矩阵分解的衍生模型（2）概率矩阵分解

- 另一种思路则着眼于参数本身的生成规律
  - 在数据稀疏且有噪声的情况下，是否可能引入某些规律，可以实现对于参数更好的描述。比如，参数符合高斯分布？

$$P(R_{ij} - U_i^T V_j | 0, \delta^2)$$

    - 假设1：参数本身与噪声均符合高斯分布
    - 假设2：两个潜在因子矩阵相互独立，各用户/项目独立同分布
    - 概率矩阵分解（Probabilistic MF）

Salakhutdinov et al. Probabilistic matrix factorization. NIPS 2008

• 矩阵分解的衍生模型（2）概率矩阵分解

- 另一种思路则着眼于参数本身的生成规律
  - 在概率矩阵分解中，两个参数矩阵均服从均值为0，预设方差的高斯分布

$$p(U|\sigma_U^2) = \prod_{i=1}^{N} N(U_i|0, \sigma_U^2 I) \qquad p(V|\sigma_V^2) = \prod_{i=1}^{N} N(V_i|0, \sigma_V^2 I)$$

  - 开放问题：如何求解PMF中的矩阵参数？基于R与U、V联合分布优化求解
  - 在参数完成训练后，即可简单地得到相应的完整评分矩阵

$$P(R|U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} [N(R_{ij}|U_i^T V_j, \sigma^2)]^{I_{ij}}$$

Salakhutdinov et al. Probabilistic matrix factorization. NIPS 2008

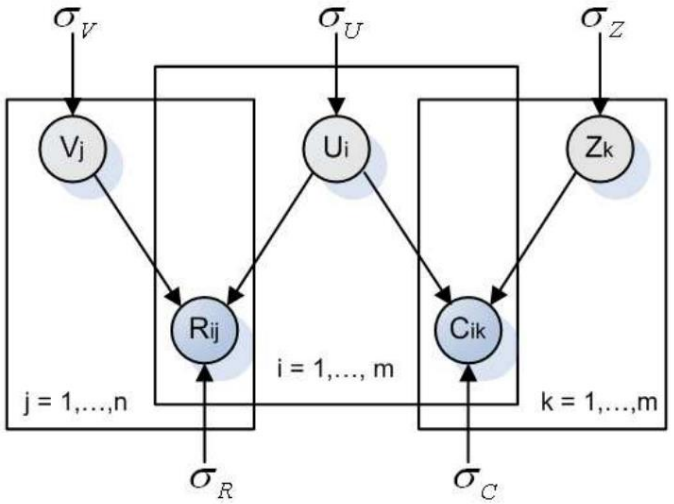
- **矩阵分解的拓展**

- 前面我们介绍了基于矩阵分解推荐技术，然而，这只是最基本的形式
  - 如何在模型中加入更多的信息，以获得更符合我们需要的结果？
  - 行内的黑话：给概率图加圈，给矩阵分解加约束，给神经网络加层
  - 通过在矩阵分解中加入约束的方式来加入更多信息与假设，往往可以提升效果
    - 例如，右边的公式实际上融入了基于内存的协同过滤思想

$$
\begin{aligned}
L \quad = \quad & \min_{U,V} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 \\
& + \frac{\alpha}{2} \sum_{i=1}^{m} \sum_{f \in \mathcal{F}^+(i)} s_{if} \|\mathbf{u}_i - \mathbf{u}_f\|_F^2 \\
& + \frac{\beta}{2} \sum_{j=1}^{n} \sum_{q \in \mathcal{Q}^+(j)} s_{jq} \|\mathbf{v}_j - \mathbf{v}_q\|_F^2 \\
& + \frac{\lambda_1}{2} \|U\|_F^2 + \frac{\lambda_2}{2} \|V\|_F^2,
\end{aligned}
$$

- **矩阵分解的拓展：社交约束**

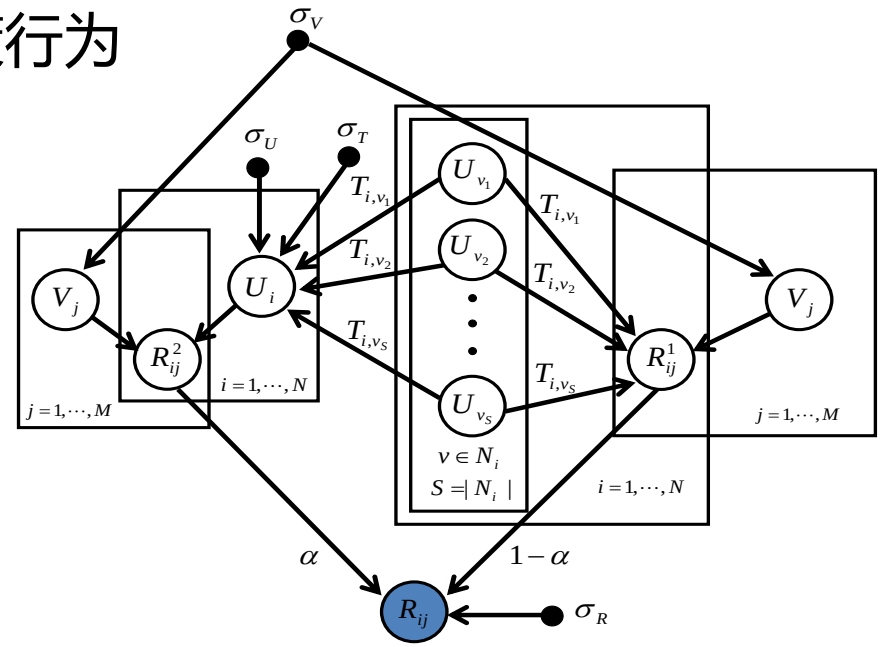
- 一种最常见的约束方式：社交约束
  - 合理而常见的假设：好友们在偏好与行为上十分相似
  - 因此，如果我们知道好友关系作为Side Information，可以对模型进行补充
  - 一个基于社交约束方面的经典工作：SoRec
    - 右半边表示社交约束，C表示社交关系
      - 存在社交关系的，属性必然相似



H Ma, et al., SoRec: social recommendation using probabilistic matrix factorization, CIKM 2008

- **矩阵分解的拓展：社交约束**

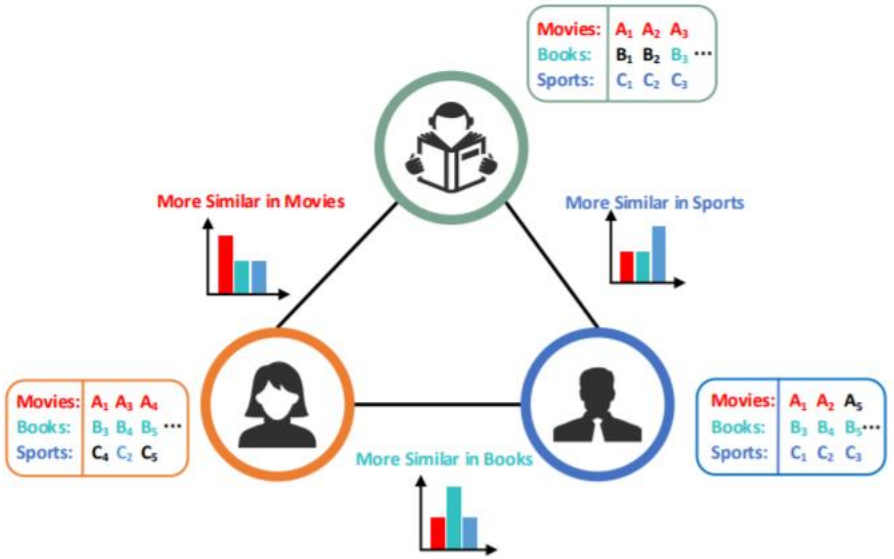
- 基于偏好相似的社交约束虽然效果很好，但未必始终成立
  - 例如：社交关系可能并非通过偏好间接影响决策，而是直接影响决策
    - 社交影响力的体现：“碍于面子”的决策行为
  - 右图的框架同时体现了这两种思路
    - R1为好友直接影响评分
    - R2为好友通过影响属性间接影响评分



杜东舫等，基于信任机制下概率矩阵分解的用户评分预测，软件学报，2018, 29(12):3747-3763.

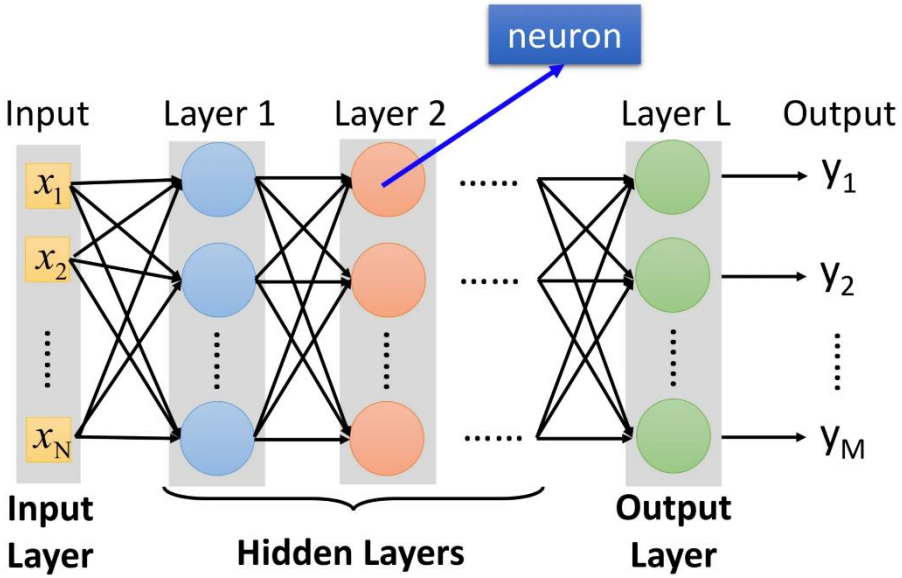
- **矩阵分解的拓展：社交约束**

- 一种最常见的约束方式：社交约束
  - 开放性问题：如何衡量用户在不同社交关系中体现出的不同偏好影响？
  - 人们在不同的社交圈子中扮演不同角色，也体现不同偏好
  - 笼统地计算所有偏好的相似性有假设过强的问题，如何解决？



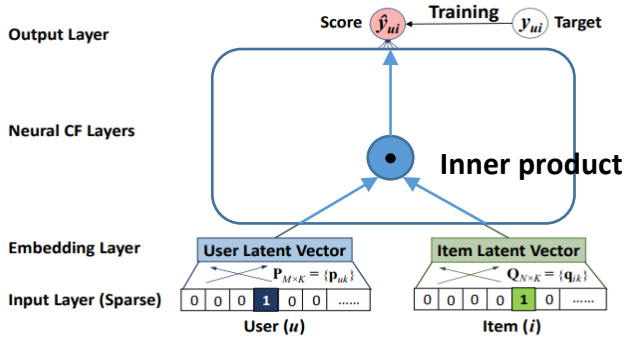
- **基于深度学习的方法**

- 前面我们主要介绍了一些传统的推荐模型，深度学习推荐系统近些年来也得到了快速发展
- 深度学习对推荐系统带来了革命性的贡献，主要体现在对模型结构的改进
  - 更强的拟合能力和特征组合的挖掘能力
  - 结构灵活性和多场景适应性



- **基于深度学习的方法: 协同过滤 feat. 深度学习**

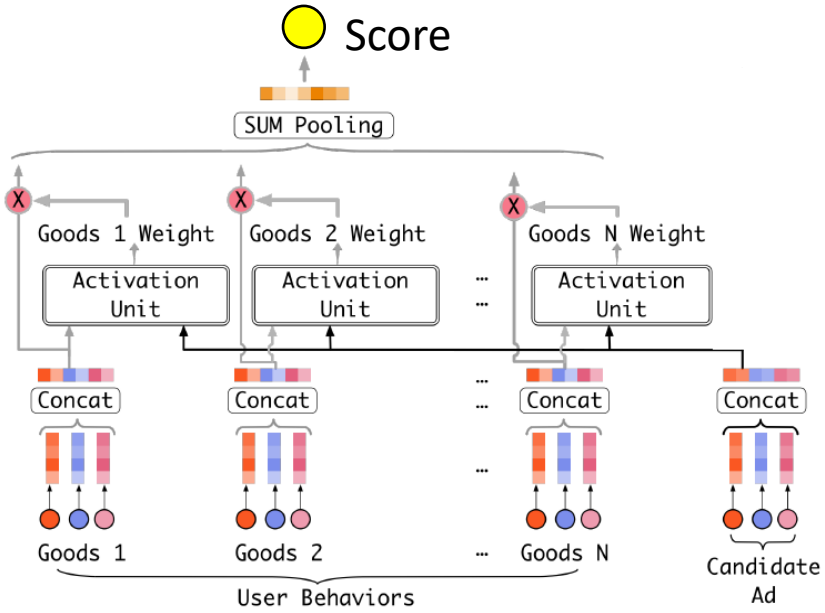
- 早期的深度推荐算法存在一个关键问题：仍然采用矩阵分解法，对用户和项目的潜在特征进行内积运算
- 然而，大部分情况下，采用简单内积作为交互函数往往是不够用的
- 用“多层神经网络 + 输出层”替代了矩阵分解中简单的内积操作
  - 让用户和项目做更加充分的交叉，获得更多有价值的特征组合信息
  - 引入更多非线性特征，让模型表达能力增强



He X, Liao L, Zhang H, et al. Neural collaborative filtering, WWW 2017

- **基于深度学习的方法: 引入注意力机制**

- 进阶思路：已有技术虽然可以借助深度学习对用户/项目进行表征，并深度模拟其复杂的交互行为，但对于项目本身没有区分其重要程度
  - 用户特征组中的项目信息被简单平均池化后送入上层网络，缺乏区分度
  - 解决思路：利用候选项目和历史行为项目之间的相关性计算出一个权重，代表“注意力”强弱

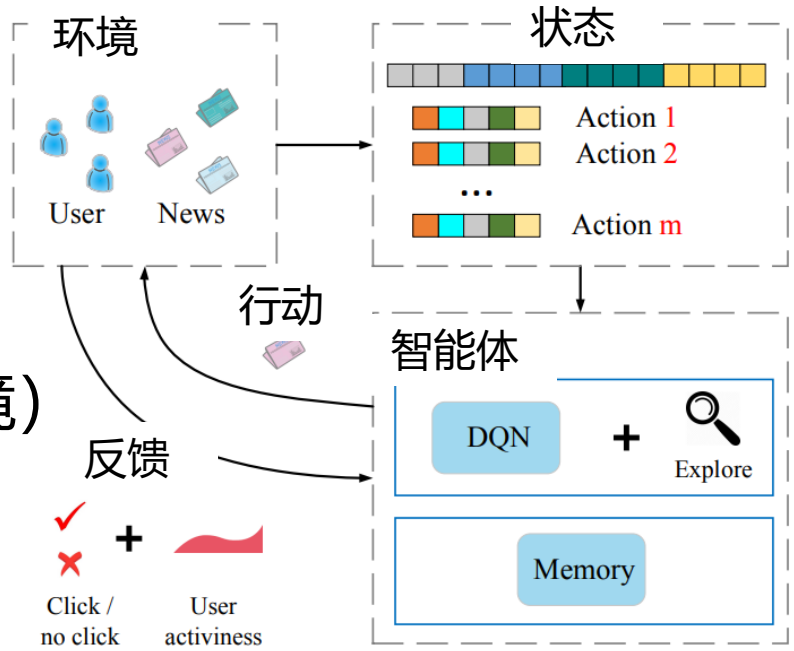


Zhou G, Zhu X, Song C, et al. Deep interest network for click-through rate prediction, SIGKDD 2018

- **基于深度学习的方法：强化学习推荐系统**

- 通过强化学习框架，推荐算法的学习过程可以不断迭代
- 强化学习推荐框架：
  - 初始化推荐系统（智能体）
  - 推荐系统会根据当前收集到的用户数据（状态）

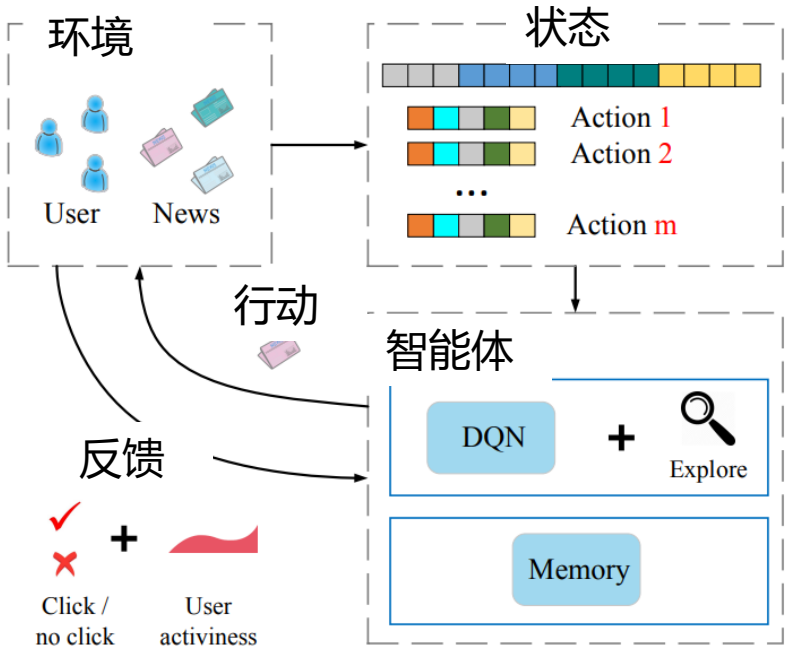
    对项目排序（行动），并推送至网站或App（环境）



Zheng G, Zhang F, Zheng Z, et al. DRN: A deep reinforcement learning framework for news recommendation, WWW 2018

- **基于深度学习的方法：强化学习推荐系统**

- 通过强化学习框架，推荐算法的学习过程可以不断迭代
- 强化学习推荐框架：
  - 用户收到推荐列表，选择点击或忽略（反馈）
  - 收到反馈，根据当前状态或通过训练更新模型
  - 重复以上步骤，以此类推



Zheng G, Zhang F, Zheng Z, et al. DRN: A deep reinforcement learning framework for news recommendation, WWW 2018

本节目录

- 基于内容的推荐
- **基于协同过滤的推荐**
  - 基于内存（Memory-based）
  - 基于模型（Model-based）
  - **应用示例：Netflix Prize**

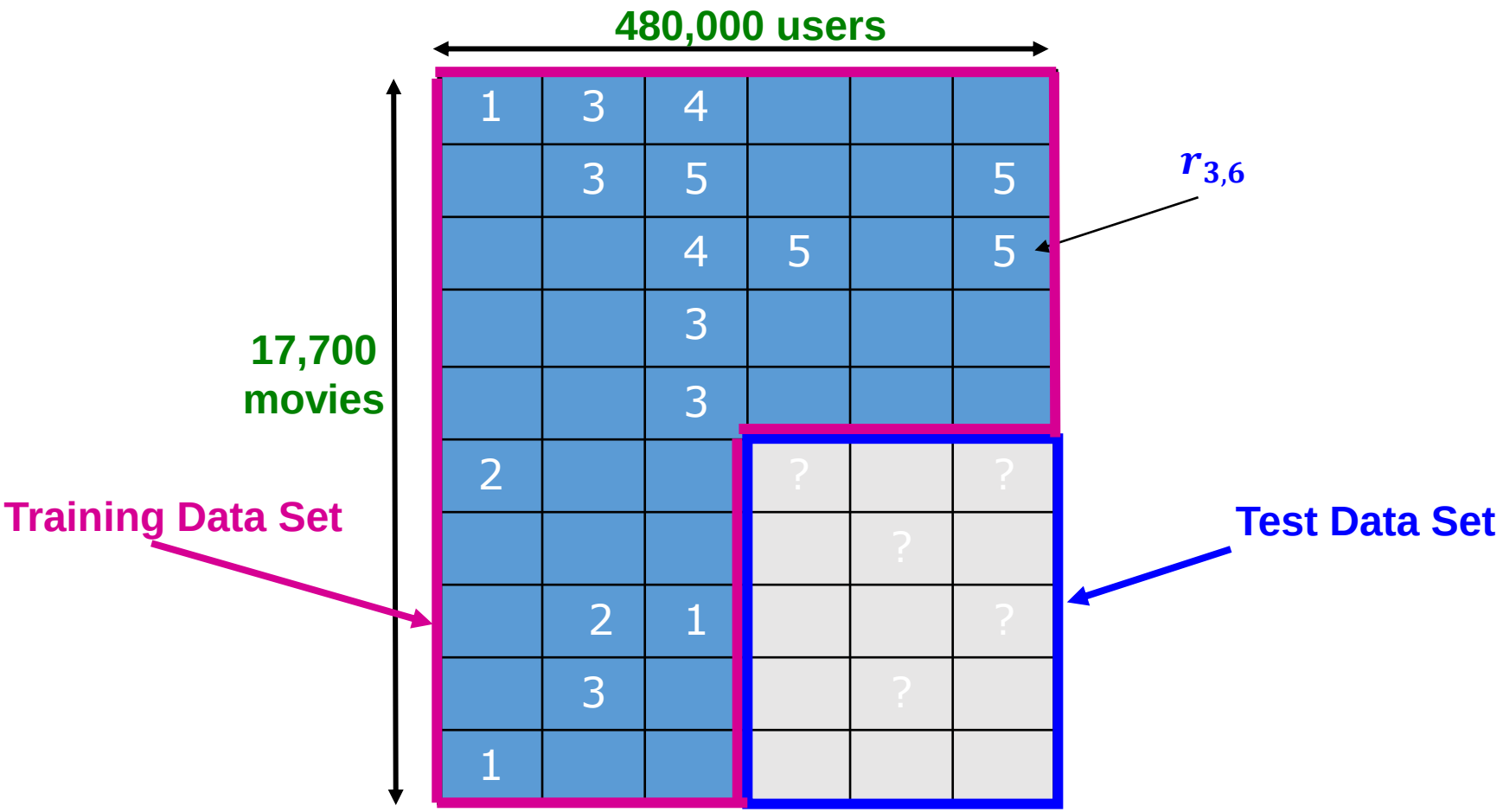
- **推荐系统典型案例：Netflix Prize**

- Netflix，流媒体巨头、世界最大的收费视频网站
  - 背景：用户与评分记录的快速增长
    - 2009年，该公司可提供多达10万部DVD电影，并有1千万的订户
    - 用户评分数量急剧增长，且用户评分标准随着用户扩大而偏移
  - 如何理解用户评分依据并实现合理推荐?
    - 2006年，Netflix发起了Netflix Prize竞赛
      - https://www.netflixprize.com/index.html（已失效）

- **推荐系统典型案例：Netflix Prize**

- 竞赛数据规模：
  - 来自于2000-2005年间的 1 亿条评分数据
  - 包含约48万个用户，约1.77万部电影
- 竞赛评价标准：采用RMSE指标，对280万条测试数据进行评分预测
  - 提升超过10%的可获得百万美元大奖（Netflix官方指标：0.9514）
- 超过2700支参赛队参数，毋庸置疑的数据挖掘领域的盛会

- **推荐系统典型案例：Netflix Prize**



- **推荐系统典型案例：Netflix Prize**

- 最终，BPC队以10.06%的提升夺得冠军

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

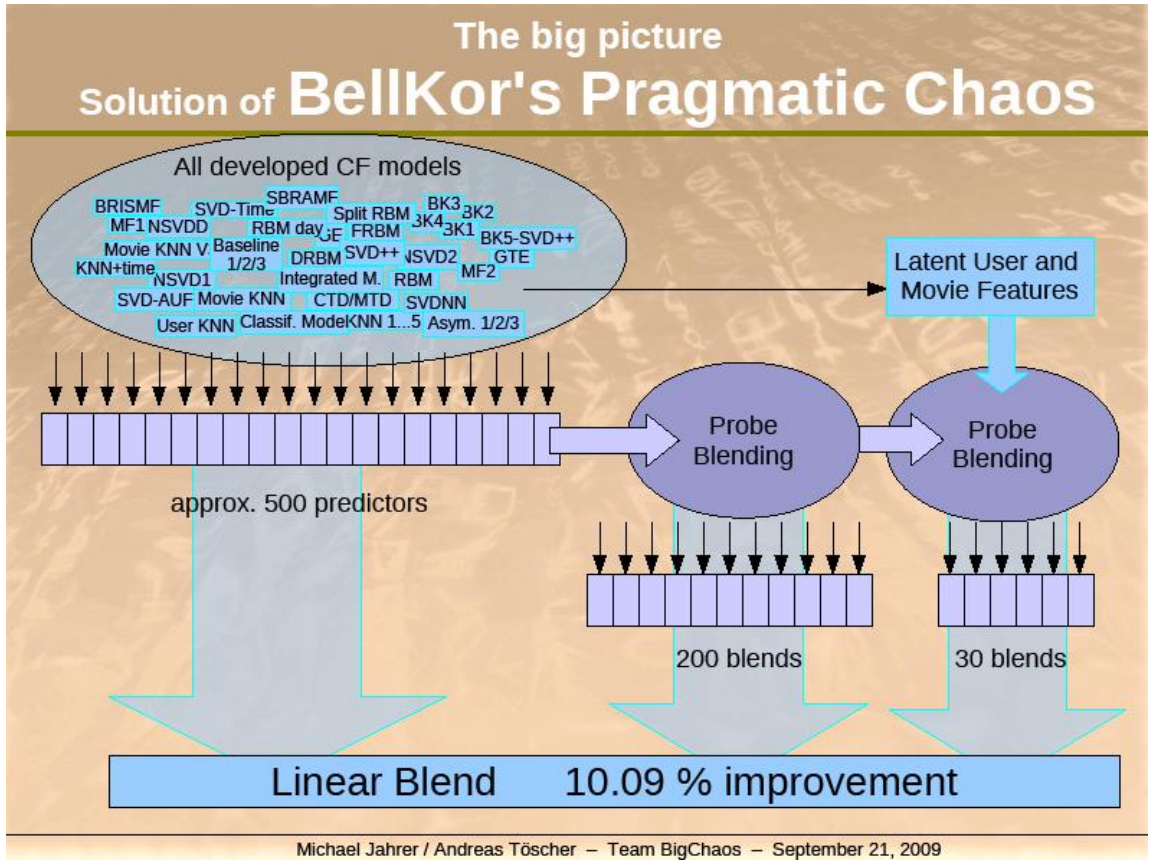
Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22
14	Gravity	0.8643	9.26	2009-04-22 18:31:32
15	Ces	0.8651	9.18	2009-06-21 19:24:53
16	Invisible Ideas	0.8653	9.15	2009-07-15 15:53:04
17	Just a guy in a garage	0.8662	9.06	2009-05-24 10:02:54
18	J Dennis Su	0.8666	9.02	2009-03-07 17:16:17
19	Craig Carmichael	0.8666	9.02	2009-07-25 16:00:54
20	acmehill	0.8668	9.00	2009-03-21 16:20:50
Progress Prize 2007 - RMSE = 0.8723 - Winning Team: KorBell				

- **推荐系统典型案例：Netflix Prize**

- BPC队的夺冠秘诀（1）集成学习
  - 采用多种不同的推荐技术，如矩阵分解、最近邻推荐
  - 采用线性集成方式，将各个推荐模型的结果进行加权整合得到最终结果



推荐看一看这篇文章：
https://blog.csdn.net/songzitea/article/details/42024399

- **推荐系统典型案例：Netflix Prize**

- BPC队的夺冠秘诀（2）全局与局部信息的融合
  - 每个人打分的习惯因人而异，有人偏高，有人偏低
  - 在这种情况下，可以根据个人打分习惯与总体趋势的偏离来估计
    - 例如，全局的电影平均分是3.7分
      - 《Sixth Sense》比全局平均高0.5
      - 某用户Joe，一般打分比平均分低0.2
      - 那么，Joe给《Sixth Sense》的打分?

- **推荐系统典型案例：Netflix Prize**

- BPC队的夺冠秘诀（3）捕捉情境化规律
  - 用户行为存在着情境化的规律，不同情境可能造成不同影响
    - 例如，用户周末情绪较好，打分偏高；而工作日情绪较差，打分偏低
    - 又如，用户不同时间的评分行为对当下呈现不同权重的影响
      - 旧的偏好会随着时间流逝而逐渐衰减

本章小结

推荐系统

- 推荐系统基本概念、关键问题、评估方法
- 基于内容的推荐方法
  - 项目与用户画像、冷启动问题、多样性问题
- 基于协同过滤的推荐方法
  - 基于内存的推荐：User-based / Item-based
  - 基于模型的推荐：矩阵分解及其衍生算法、基于深度学习的推荐方法
  - 推荐系统实例：Netflix Prize

tongxu@ustc.edu.cn