Web信息处理与应用

第七节

网页排序（下）

徐童　2024.10.24

上节回顾

- **一个好的检索应该是怎样?**

- 如何做出选择?
  - B医生，既治牙病，又治眼病，从医二十年
  - C医生，专治牙病，只有五年从医经验
- 从择医的角度考虑，需要同时考虑专长与医术
- 对于**网页排序**而言，也是如此
  - 网页内容匹配程度 → 医生的专长
  - 网页内容的质量 → 医生的经验与水平

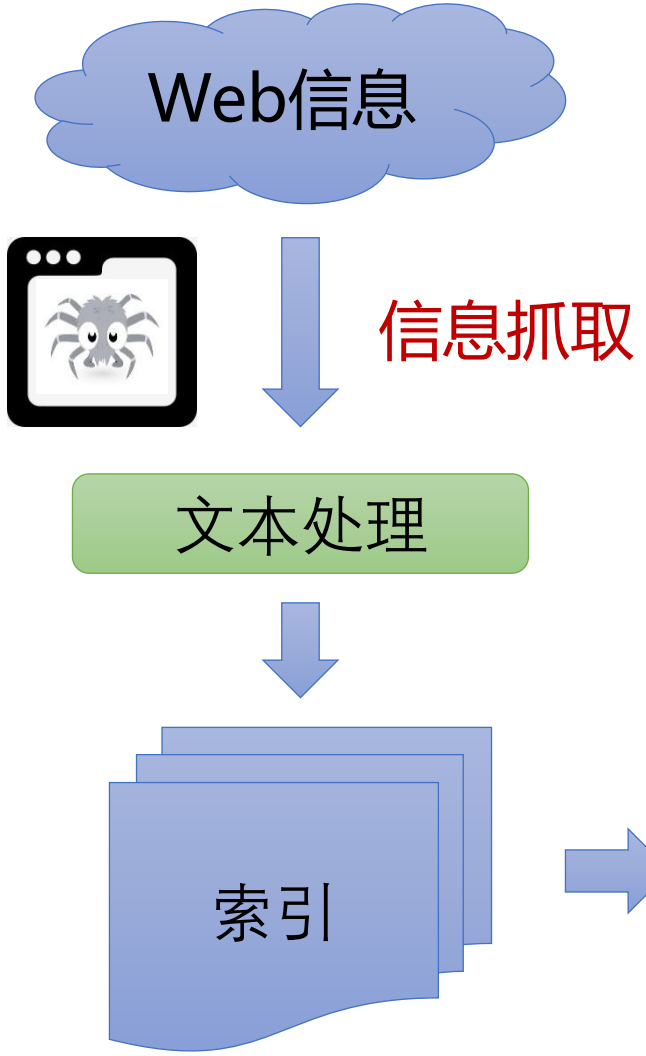
上节回顾

- **信息检索模型中的相关性**

- 相关性可通过以下两个维度进行衡量
  - 主题相关：文档与查询在主题上的一致性
    - 某种程度可体现在字面意义上的匹配性
  - 用户相关：文档在多大程度上满足用户需求
    - 可通过用户反馈判定，也可结合用户意图进行判定
- 相应的，可以通过二元（相关/不相关）或排序的方式进行评判
- 不同的维度有着不同的评价体系！

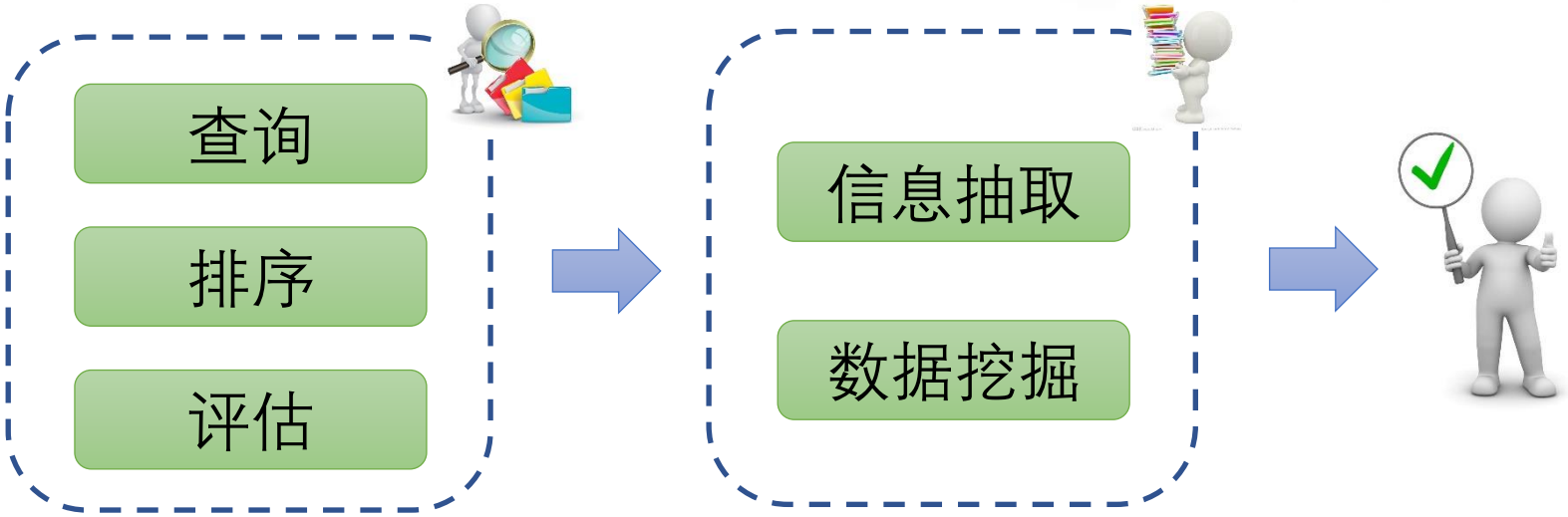
本节主题

- 本课程所要解决的问题



第七个问题：

如何对网页进行排序，在保障语义匹配的同时确保质量？



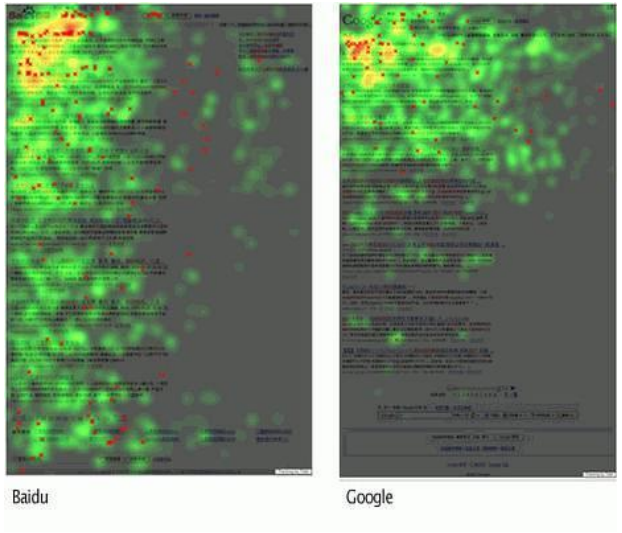
本节目录

- **排序学习算法**
  - Pointwise 算法
  - Pairwise 算法
  - Listwise 算法
- 网页权威性计算
  - PageRank及其衍生模型
  - HITS算法与网页角色区分

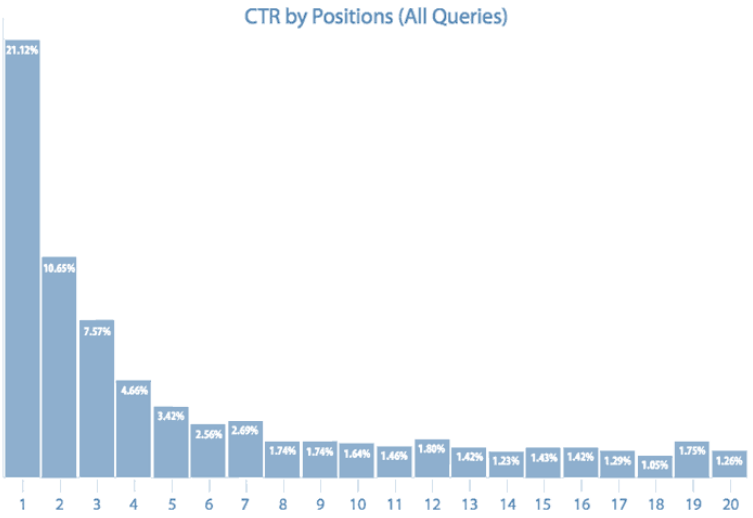
本节主题

- **网页排序的本质**

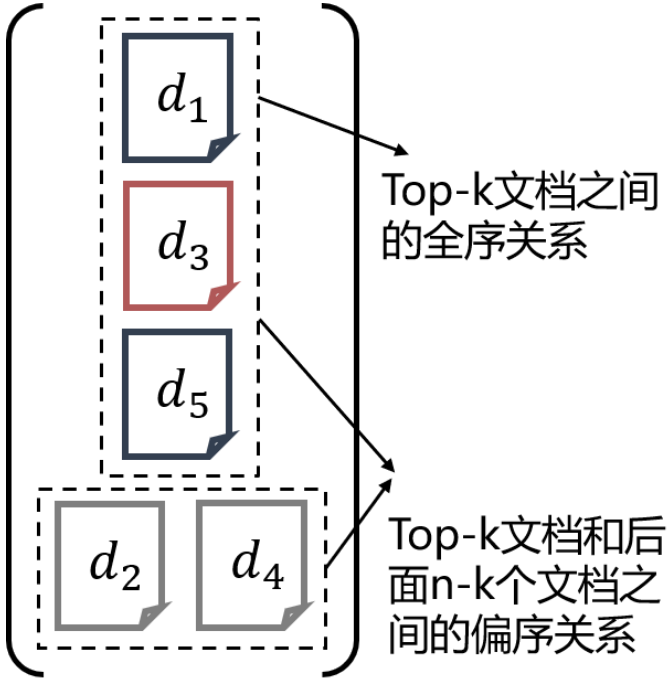
- 对于网页内容的排序问题，本质上是一个学习排序器（Ranker）的问题
  - 根据用户的需求 / 偏好，把更符合需求的网页文档尽可能往前排



搜索结果的眼动研究



不同位置的点击率统计

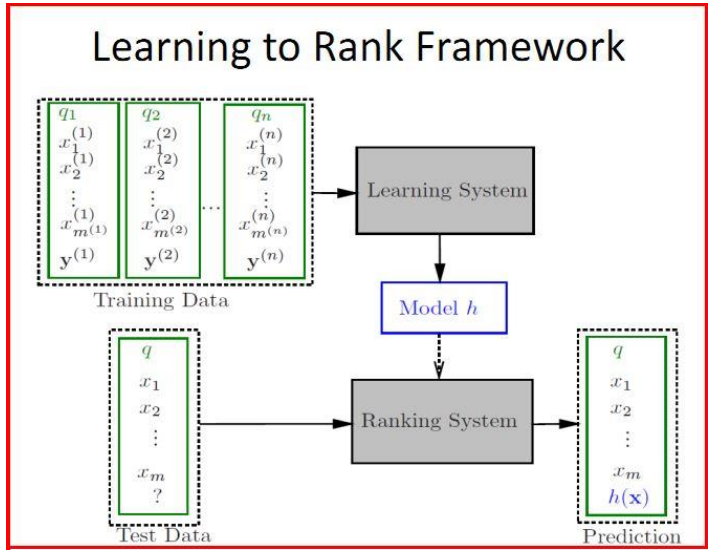


- **从网页排序到排序学习**

- 构想一个场景：如何基于用户的相关性反馈/评级，对网页进行排序?
  - 两种模式：已知排序依据与未知排序依据
    - ➢ 已知排序依据：根据已知信息和排序方法对文档进行排序
      - ◻ 如：基于查询条件 + 文档表示学习的向量空间模型
    - ➢ 未知排序依据：不知道用户喜欢什么样的文档，只能根据历史记录推测
      - ◻ 如：借助排序学习（Learning to Rank）推测用户查询意图和偏好

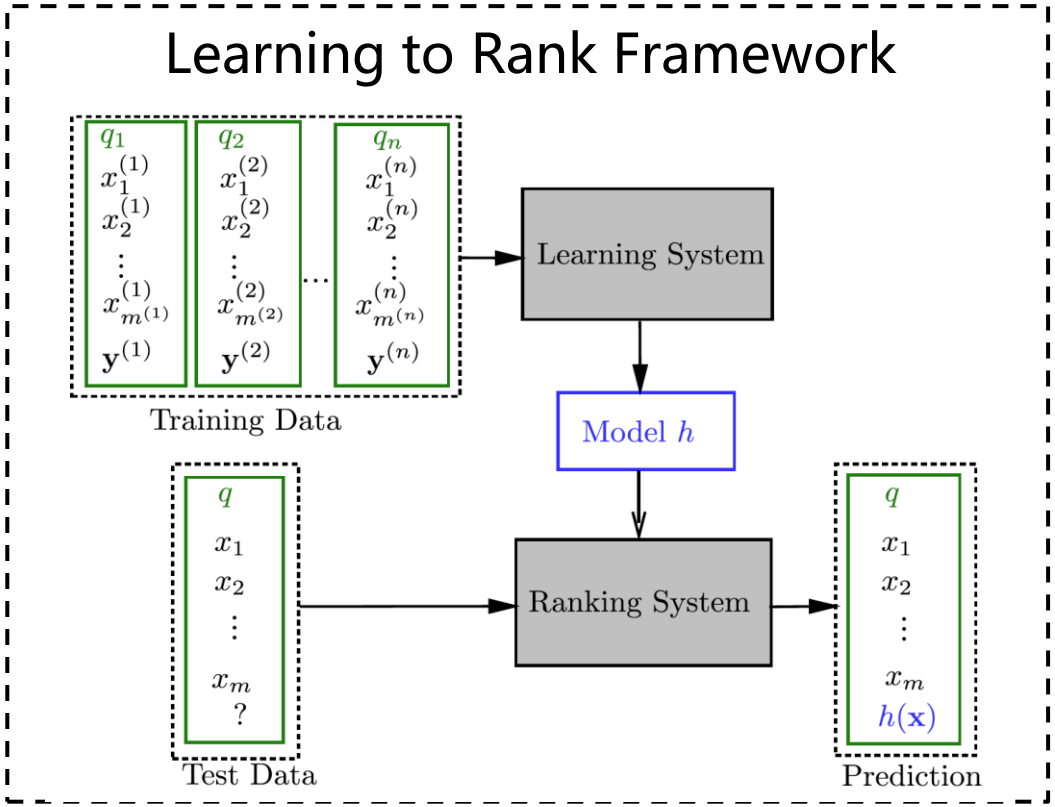
- **从网页排序到排序学习**

- 排序学习（Learning to Rank）需要做些什么?
  - 通过用户相关性反馈，训练一个学习排序器（Ranker）来解决该问题
  - 输入网页（文档）内容及查询，输出该网页合适的（相对）排序位置
  - 排序学习是个有监督学习问题
    - 基于已知的排序（如用户反馈）
    - 学习一个表示用户偏好的Ranker
    - 基于Ranker为新网页-查询对给出排序



- **排序学习的独特之处**

- 排序学习与传统有监督学习的区别
  - 排序学习需要同时考虑文档内容和查询信息（成对输入）
  - 相较于文档的精确得分，排序学习更关心文档之间的相对顺序



- **排序学习的训练数据问题**

→

1：相关
0：不相关

两级标注

或者

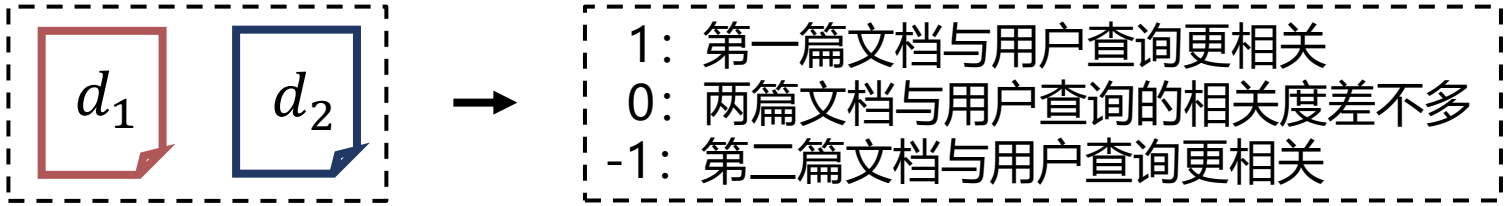
2：很相关
1：部分相关
0：不相关

多级标注

- 一般训练数据：用户提供网页文档的相关性判别 / 等级
  - 相关度级别的选择较为困难
  - 人工标注的代价高
  - 主观性强，标注结果存在高度分歧

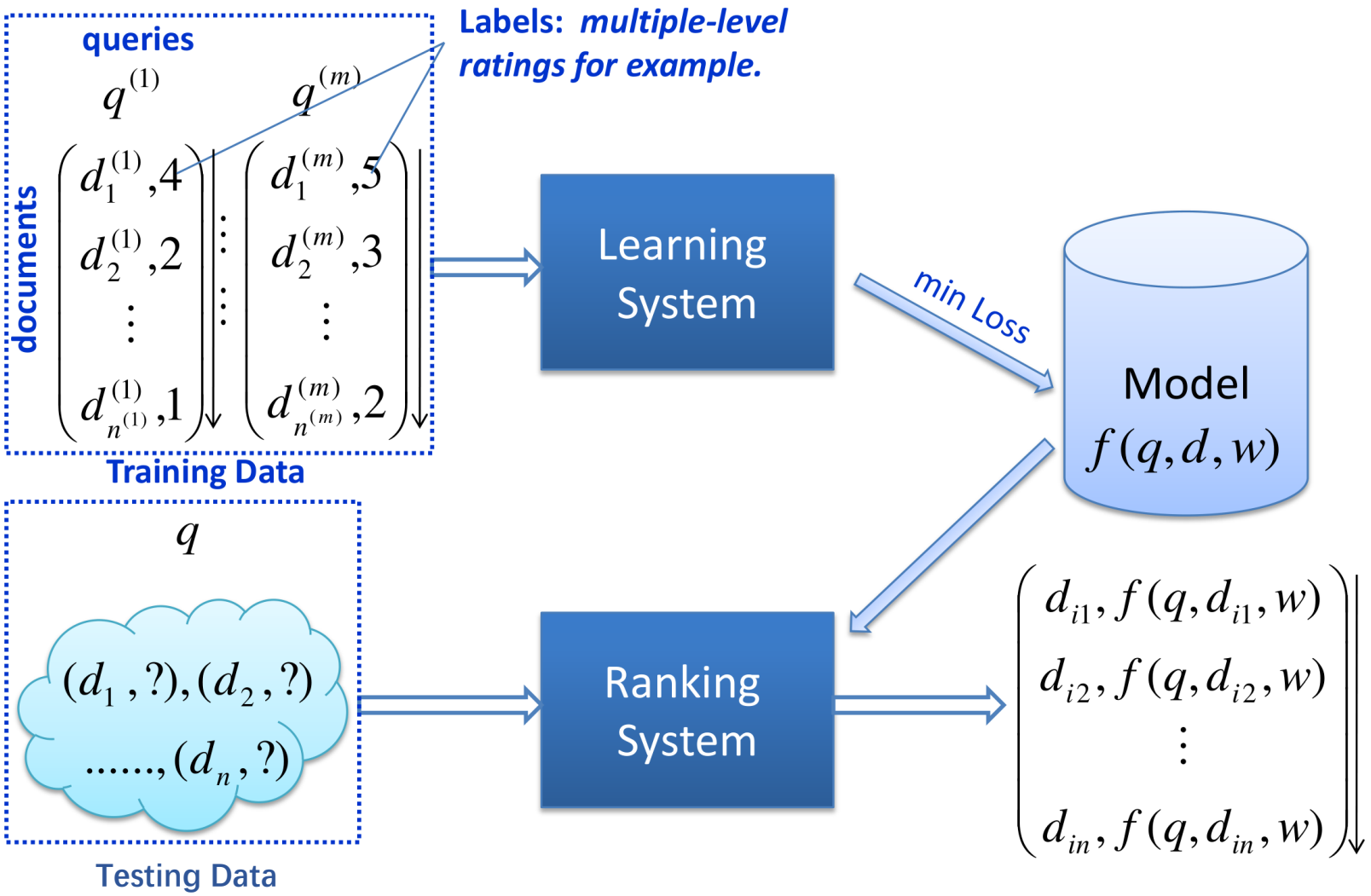
如何高效地获取L2R的高质量标注数据?

- **排序学习的训练数据问题**



- 更方便的数据标注：用户提供两篇文档的相对优劣关系
- 优点
  - 无需选择相关度的级别，更容易进行人工标注
  - 标注质量有一定的提高
- 缺点
  - 标注的数量急剧增大（从$O(n)$到$O(n^2)$）

• 排序学习的一般流程



← 引自Tutorial @ WWW 2008

- **三类常见的排序学习算法**

- Pointwise：将排序退化为分类或回归问题
  - 输出：网页对应的分类（有序）、回归值或有序回归值
- Pairwise：比较一对网页之间的相关度，e.g., 相关 > 不相关
  - 输出：网页对之间的偏序关系
- Listwise：对整个网页集合进行排序
  - 输出：整个集合的完整排序，往往依赖特定排序指标

本节目录

- **排序学习算法**
  - **Pointwise 算法**
  - Pairwise 算法
  - Listwise 算法
- 网页权威性计算
  - PageRank及其衍生模型
  - HITS算法与网页角色区分

Pointwise

- **Pointwise类排序算法**

- 基本假设：训练样本中的任何一个查询-文档对，都可以映射到一个分值或一个有序的类别（如优良中差）
- 相应的，给定一个查询-文档对，Pointwise LTR将试图预测其分值/类别。
- 常见的模型类别包括：
  - 回归，将查询-文档对映射到具体分数
  - 分类，将排序问题转化为一个面向有序类别的二分类/多分类问题
  - 有序回归，在映射到具体分数的同时保持样本之间的有序关系

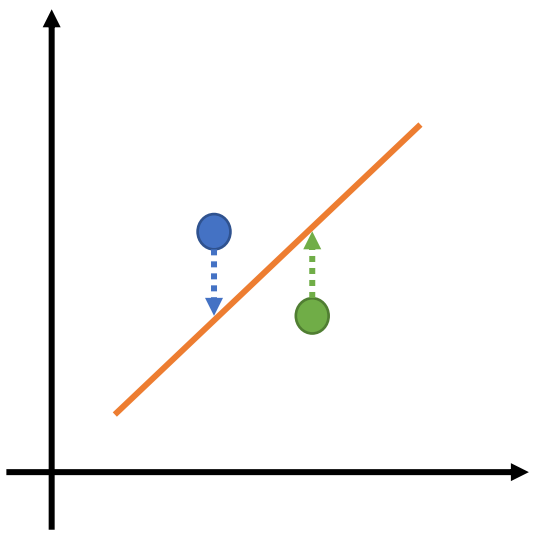
Pointwise

- **Pointwise类排序算法**

- 第一类方法：回归方法（Regression）
- 基于回归算法的一般损失函数如下所示：

$$L\left(f; x_j, y_j\right) = \left(y_j - f\left(x_j\right)\right)^2$$

- 该类方法实现较为简单，但缺陷也较为明显。
  - 最大的缺陷在于没有对文档间的排序添加约束。
  - 因此，可能为降低Loss所误导而错判文档间的顺序。



Pointwise

- **Pointwise类排序算法**

- 第二类方法：分类方法（Classification）
- 采用分类方法，将查询-文档对映射到二分类（相关/不相关）或多分类
  - 代表性算法：Discriminative Model for IR (R. Nallapati, SIGIR 2004)
  - 通过抽取特征表征文档，相关文档被视作正样本，不相关为负样本

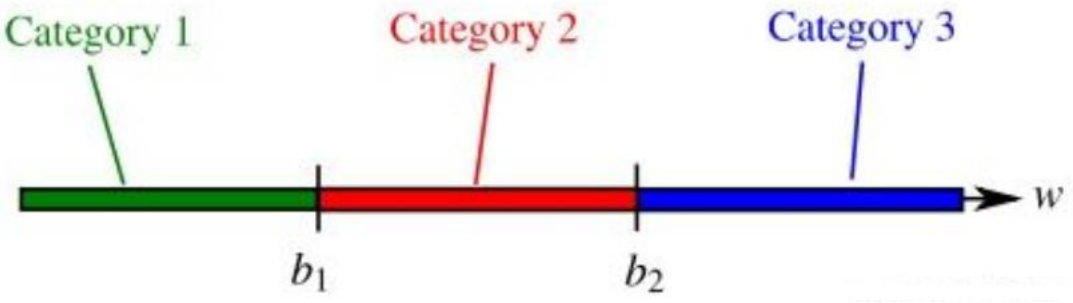
	Feature		Feature
1	$\sum_{q_i \in Q \cap D} log(c(q_i, D))$	4	$\sum_{q_i \in Q \cap D} (log(\frac{\lvert C \rvert}{c(q_i, C)}))$
2	$\sum_{i=1}^{n} log(1 + \frac{c(q_i, D)}{\lvert D \rvert})$	5	$\sum_{i=1}^{n} log(1 + \frac{c(q_i, D)}{\lvert D \rvert} idf(q_i))$
3	$\sum_{q_i \in Q \cap D} log(idf(q_i))$	6	$\sum_{i=1}^{n} log(1 + \frac{c(q_i, D)}{\lvert D \rvert} \frac{\lvert C \rvert}{c(q_i, C)})$

  - 有关分类的常用技术可参考校内人工智能/机器学习相关课程

Pointwise

- **Pointwise类排序算法**

- 第三类方法：有序回归方法（Ordinal Regression）
  - 某种意义上，相当于利用回归方法求解有序多分类问题。



  - 其中，文档的标签可通过如下公式推测：

$$\hat{y}_j = \arg\min_k \{w^T x_j - b_k < 0\}.$$

  - 投影后，位于数轴上的区间决定了文档的标签。

Pointwise

- **Pointwise类排序算法**

- Pointwise类排序算法可以简单且广泛地套用已有回归、分类算法
- 然而，其局限性也较为明显
  - 首先，Pointwise类方法往往更为注重文档的相关度得分，而并不注重文档之间的相关性排序
    - Pairwise类方法的出现，为解决这一问题提供了新的手段
  - 其次，不同查询所对应的文档，尤其相关文档数量不同，对损失函数的贡献也各不相同，一定程度上影响效果

本节目录

- **排序学习算法**
  - Pointwise 算法
  - **Pairwise 算法**
  - Listwise 算法
- 网页权威性计算
  - PageRank及其衍生模型
  - HITS算法与网页角色区分

Pairwise

- **Pairwise类排序算法**

- 基本假设：同样是将排序问题转化为分类问题（二分类或三分类）
  - 每次比较一个查询与两个文档，衡量两个文档的偏序（Partial Order）
  - 分类器的目的在于判断哪个文档应该排在前面
    - 对应的标签为{1, -1}或{1, 0, -1}（0表示两个文档可以并列）

$$q^{(i)} \begin{pmatrix} d_1^{(i)}, 5 \\ d_2^{(i)}, 3 \\ \vdots \\ d_{n^{(i)}}^{(i)}, 2 \end{pmatrix} \xrightarrow{\text{Transform}} q^{(i)} \left\{ \left(d_1^{(i)}, d_2^{(i)}\right), \left(d_1^{(i)}, d_{n^{(i)}}^{(i)}\right), \ldots, \left(d_2^{(i)}, d_{n^{(i)}}^{(i)}\right) \right\}$$

$$5 > 3 \qquad 5 > 2 \qquad 3 > 2$$

- **Pairwise类排序算法**

- 代表性方法：Ranking SVM (R. Herbrich, et al. ICANN 1999)
  - 转化为分类问题后，该类算法同样可以借鉴常用的分类方法
    - 例如，通过改进经典的SVM算法实现二分类（如下，当D1优于D2）

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\xi_i$$

$$z_i\left\langle w, \boxed{x_i^{(1)} - x_i^{(2)}}\right\rangle \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

$x^{(1)}$-$x^{(2)}$ as positive instance of learning

Use SVM to perform binary classification on these instances, to learn model parameter *w*

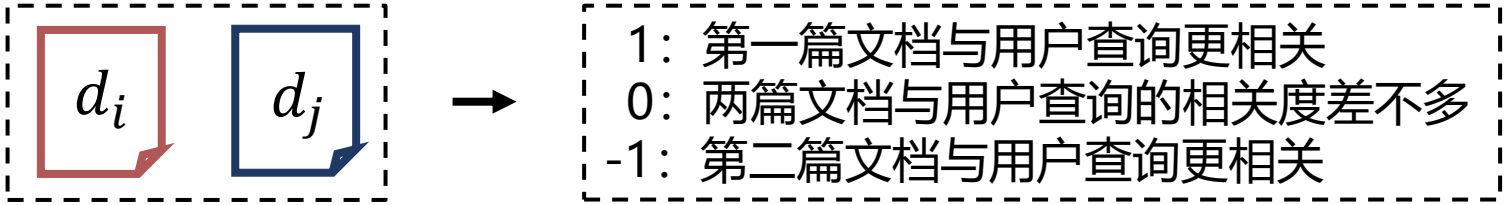
$$f(x;\hat{w}) = \boxed{<\hat{w}, x>}$$

Use *w* for testing

Pairwise

- **Pairwise类排序算法**

- 代表性方法：RankNet (Burges, Chris, et al. ICML 2005)
  - 训练过程
    - 构造特定于同一查询的文档对 $pair(d_i,\ d_j)$



对应的标签 s_{ij}

    - 将 $pair(d_i,\ d_j)$ 输入 RankNet 模型
    - 得到预测值，计算损失函数
    - 迭代优化模型参数

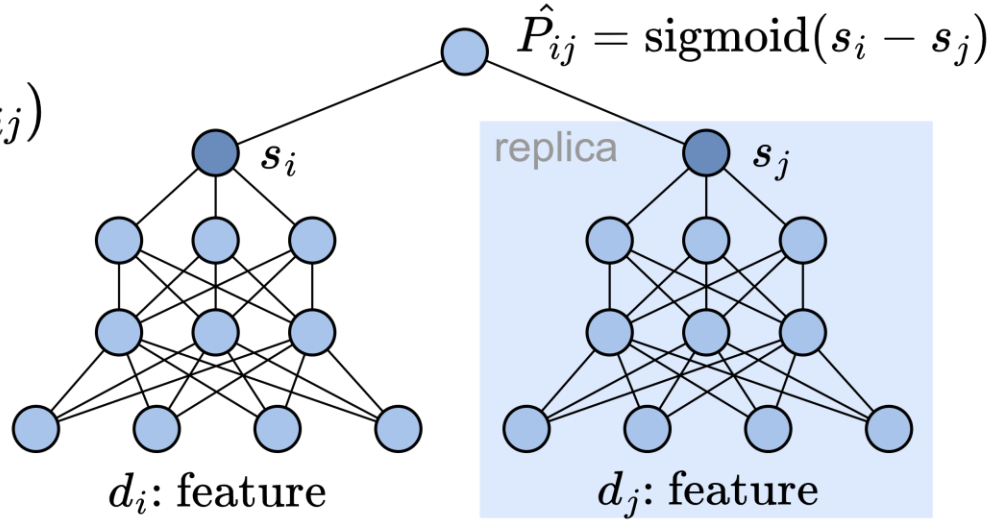
Pairwise

- **Pairwise类排序算法**

- 代表性方法：RankNet (Burges, Chris, et al. ICML 2005)
  - 训练过程
    - 真实标签 S_{ij} $\{1,\ 0,\ -1\}$ $\xrightarrow{P_{ij}=\frac{1}{2}(S_{ij}+1)}$ 概率值 P_{ij} $\{1,\ 0.5,\ 0\}$
    - 损失函数

$$C_{ij} = -P_{ij}\log\hat{P}_{ij} - (1-P_{ij})\log(1-\hat{P}_{ij})$$

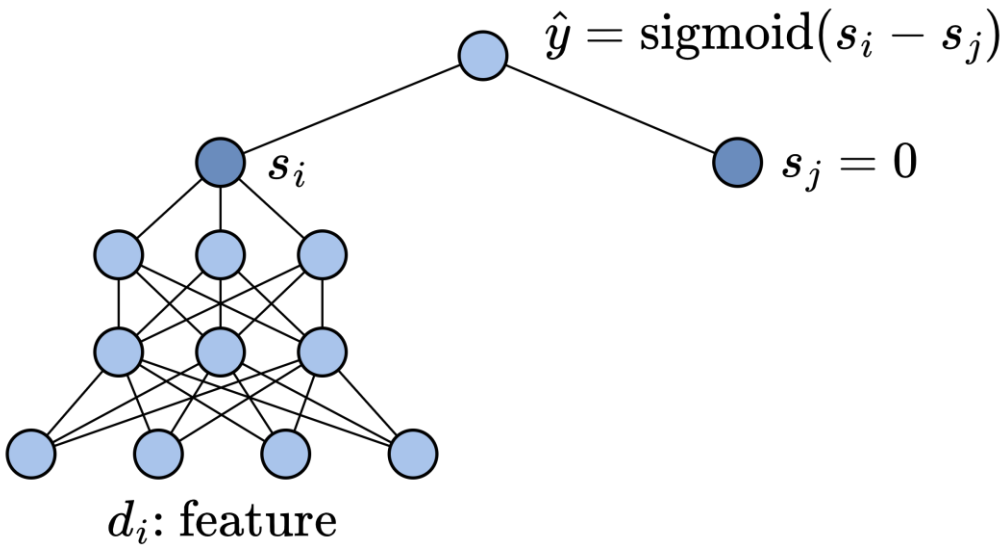
$$C = \sum_{(i,j)\in I} C_{ij}$$



Pairwise

- **Pairwise类排序算法**

- 代表性方法：RankNet (Burges, Chris, et al. ICML 2005)
  - 预测阶段
    - 将文档 i 的查询-文档对特征输入模型，得预测值为 $\hat{y} = f(d_i, 0)$
    - 对文档进行排序输出



Pairwise

- **Pairwise类排序算法**

- 相比于Pointwise类算法，Pairwise类排序算法通过衡量样本之间的偏序关系，实现了从绝对相关性（分值）到相对偏序的改进
- 然而，Pairwise类算法也具有自己的缺陷：
  - 首先，两两成对导致样本数大为提升，计算资源开支增加
  - 其次，Pairwise类算法仍然受样本不平衡问题的影响
  - 最后，Pairwise类算法无法体现全局排序的合理性
    - 由此引出了最后一类算法：Listwise类算法

Pairwise

- **Pairwise类排序算法**

- 局限性：样本不平衡性的影响
  - 如果不同Query的Doc数量相差很大，从偏序的对来说，Doc（对）数量较多的Query将掩盖其他的Query，从而产生干扰
  - 计算指标时可通过加权等方法加以缓解

		Case 1	Case 2
Document pairs of q_1	correctly ranked	770	780
	wrongly ranked	10	0
	Accuracy	98.72%	100%
Document pairs of q_2	correctly ranked	10	0
	wrongly ranked	0	10
	Accuracy	100%	0%
overall accuracy	document level	98.73%	98.73%
	query level	99.36%	50%

- **Pairwise类排序算法**

- 局限性：算法无法体现全局排序
  - 某个成对文档排序错误，发生在不同位置，在Pairwise类条件下是等价的
  - 例如：7篇文档，相关性从1到3，理想状态下的排序为3 2 2 1 1 1 1
    - 假如有如下两个排序：
    - Rank A：2 3 2 1 1 1 1　　Rank B：3 2 1 2 1 1 1
    - 这两个排序，在Pairwise类算法中是等价的，然而，后者显然更合理
      - 如前所述，用户往往更关心排在靠前位置的文档

本节目录

- **排序学习算法**
  - Pointwise 算法
  - Pairwise 算法
  - **Listwise 算法**
- 网页权威性计算
  - PageRank及其衍生模型
  - HITS算法与网页角色区分

- **Listwise类排序算法**

- 基本思路：直接面向整体排序结果进行优化
  - 直接将排序的完整队列作为学习的对象
- 通常情况下，解决这一问题采用以下两种思路
  - 直接采用某种IR指标对排序进行优化
  - 直接设计面向完整排序的损失函数

Listwise

- **Listwise类排序算法**

- 代表性思路（1）：采用某种IR指标对排序进行优化
  - 此类方法较为直观，且可以直接优化所获得排序结果的衡量指标
  - 然而，该类方法也面临明显的挑战：
    - 大多数排序指标，如NDCG等，因为与排序相关，属于非光滑、不可微的函数。
    - 因此，传统的优化方法很难直接应用于该问题。

- **Listwise类排序算法**

- 一种有趣的解决方案：AdaRank (J. Xu, et al. SIGIR 2007)
  - AdaBoost的变形，采用弱排序器的组合来实现全局排序。
    - AdaBoost是解决分类问题的经典算法，其思路大致如下：
      1. 生成一组弱分类器（如仅用一维特征进行分类）
      2. 根据结果更新分类器和样本的权重
         - 效果更好的分类器，权重更高；分类错误的样本，权重更高
      3. 面向权重更新后的样本，训练新的弱分类器
      4. 最终，按照权重整合所有的弱分类器，形成一个完整的强分类器

Listwise

- **Listwise类排序算法**

- 将AdaBoost的思路迁移至排序学习问题，即AdaRank
  - 借鉴了AdaBoost的设定及步骤。
  - 基于部分特征和指定指标（如NDCG），得到弱排序器。
  - 通过调节权重，重点修正那些在先前排序中排序错误的文档。
  - 回避了直接优化全局NDCG的问题。

Input: $S = \{(q_i, \mathbf{d}_i, \mathbf{y}_i)\}_{i=1}^m$, and parameters E and T
Initialize $P_1(i) = 1/m$.
For $t = 1, \cdots, T$

- Create weak ranker h_t with weighted distribution $\boldsymbol{P}_t$ on training data S.
- Choose α_t

$$\alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^m P_t(i)\{1 + E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}{\sum_{i=1}^m P_t(i)\{1 - E(\pi(q_i, \mathbf{d}_i, h_t), \mathbf{y}_i)\}}.$$

- Create f_t

$$f_t(\vec{x}) = \sum_{k=1}^{t} \alpha_k h_k(\vec{x}).$$

- Update $\boldsymbol{P}_{t+1}$

$$P_{t+1}(i) = \frac{\exp\{-E(\pi(q_i, \mathbf{d}_i, f_t), \mathbf{y}_i)\}}{\sum_{j=1}^m \exp\{-E(\pi(q_j, \mathbf{d}_j, f_t), \mathbf{y}_j)\}}.$$

End For
Output ranking model: $f(\vec{x}) = f_T(\vec{x})$.

Listwise

- **Listwise类排序算法**

- 代表性思路（2）：直接设计面向完整排序的损失函数
- 代表性工作：RankCosine (T. Qin, T. Liu, et al. IP&M 2007)
  - 将排序器所得的排序向量与标准答案的排序向量之间的余弦相似度作为目标函数：

$$J(\mathbf{g}(q),\mathbf{H}(q)) = \tfrac{1}{2}\left(1-\cos(\mathbf{g}(q),\mathbf{H}(q))\right) = \tfrac{1}{2}\left(1-\frac{\mathbf{g}(q)^T\mathbf{H}(q)}{\|\mathbf{g}(q)\|\|\mathbf{H}(q)\|}\right)$$

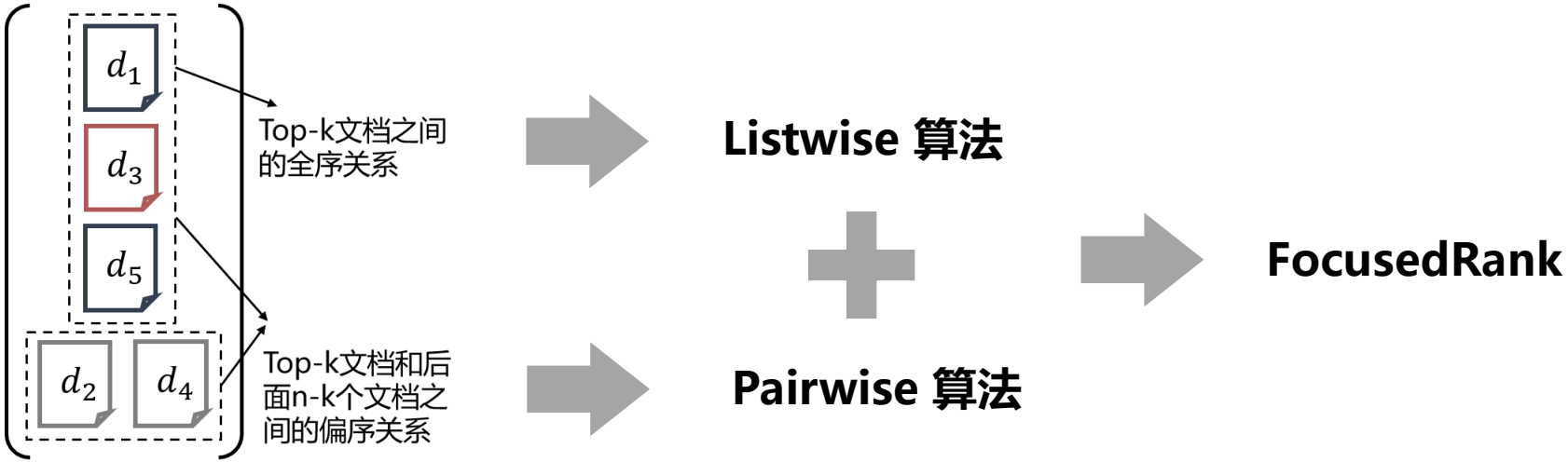
    - 其中，g(q)为标准答案，H(q)为待训练的排序器
      - H(q)同样由一组带权的弱排序器组合而成，权重由g(q)与H(q)的逐步迭代所得

- **Listwise类排序算法**

- 通常情况下，由于全局优化的作用，Listwise类排序算法可以取得相比于Pointwise和Pairwise类算法更好的效果。
- 然而，Listwise类算法也会面临一些小的挑战，例如两个网页并列的情况
  - 相当于Pairwise中文档对标签为0的情况

- **排序算法拓展**

- Listwise与Pairwise的融合：FocusedRank (Shuzi Niu , et al. SIGIR 2012)



$$L(f;q_i) = \beta \times L_{\text{list}}\left(f;T_i,\mathbf{y}_i\right) + (1-\beta) \times L_{\text{pair}}\left(f;P_i,\mathbf{y}_i\right)$$

工具包

- **Learning To Rank 工具包**

- 上古LTR工具包Ranklib：https://sourceforge.net/p/lemur/wiki/RankLib/
- TensorFlow：TF-Ranking https://github.com/tensorflow/ranking
- PyTorch：allRank https://github.com/allegro/allRank
- 提供不同损失函数，pointwise，pairwise，listwise
- 提供不同排序指标，MRR，NDCG 等

本节目录

- 排序学习算法
  - Pointwise 算法
  - Pairwise 算法
  - Listwise 算法
- **网页权威性计算**
  - PageRank及其衍生模型
  - HITS算法与网页角色区分

上古回顾

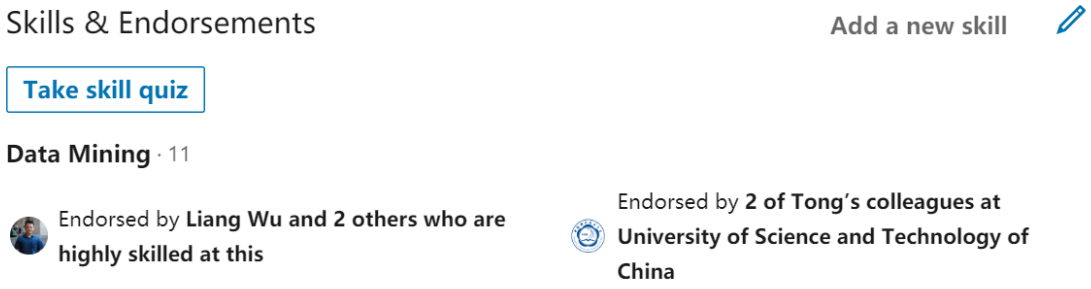
- **Web 2.0时代的隐患与网页权威性的意义**

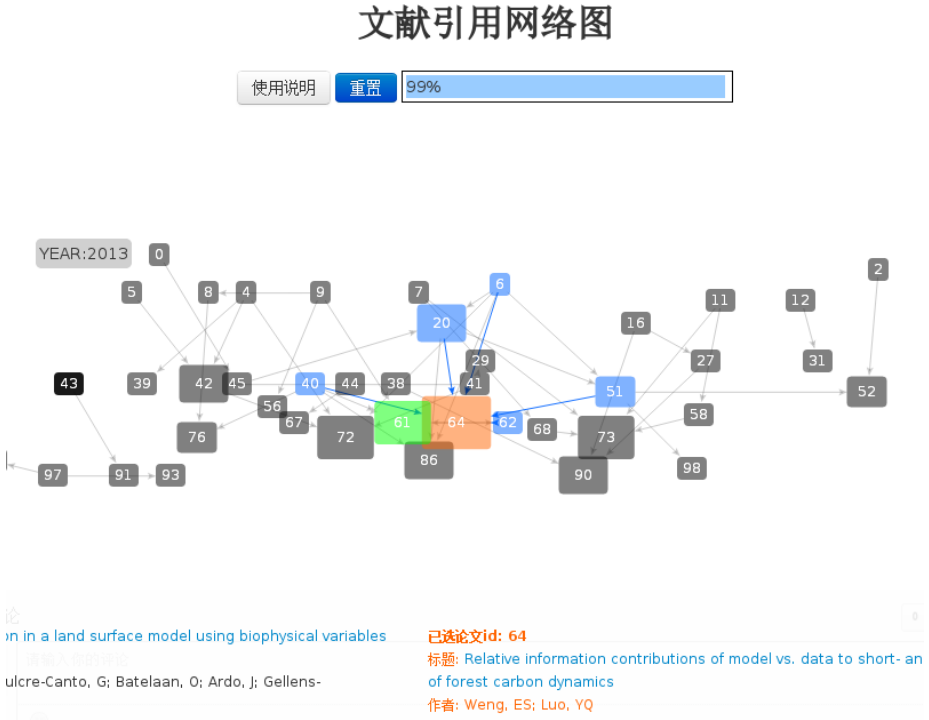
- 大众创造内容的时代后遗症：内容权威性的下降
  - 信息的错漏、谣言与虚假信息的兴起
  - 低质内容服务商大量涌现
- 如何在衡量相关性的同时，确保内容权威性？

PageRank

- **一个启发式的想法**

- 大家说好才是真的好：来自同行的endorsement
  - 区分“点赞”的人将获得更好区分效果





本节目录

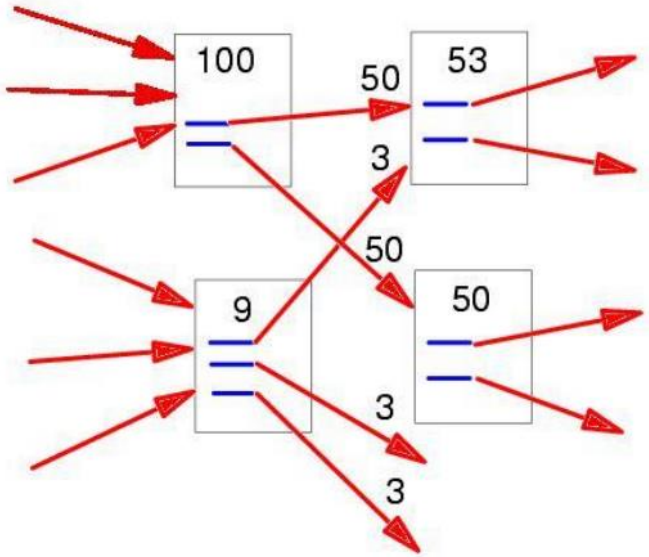
- 排序学习算法
  - Pointwise 算法
  - Pairwise 算法
  - Listwise 算法
- **网页权威性计算**
  - **PageRank及其衍生模型**
  - HITS算法与网页角色区分

• PageRank的历史

- Sergey Brin 和Lawrence Page 在1998年提出了PageRank 算法
  - Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*，Stanford InfoLab, 1999
    - 截止2024.10.24，引用19323次
    - http://www-db.stanford.edu/~backrub/pageranksub.ps
  - Sergey Brin, Lawrence Page, *The Anatomy of a Large-scale Hypertextual Web Search Engine*. WWW'98, Computer Networks 30(1-7): 107-117 (1998)
    - 截止2024.10.24，引用24490次
    - https://ai.google/research/pubs/pub334.pdf

• PageRank的设计思路

- 将网页视作一个点，网页间的超链接视作一条边，形成一个巨大的有向图
- PageRank背后的总体思路：优质网页引用或推荐的网页，一定还是优质网页
  - 三重衡量标准：
    - 链入链接数：单纯意义上的受欢迎程度
    - 链入链接本身是否推荐度高：欢迎本身是否具有权威性
    - 链入链接源页面的链接数：被选中的几率
      - 体现源网页是否滥发推荐



- **PageRank的计算方法**

- PageRank的核心公式如下：

$$\mathrm{PR}(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

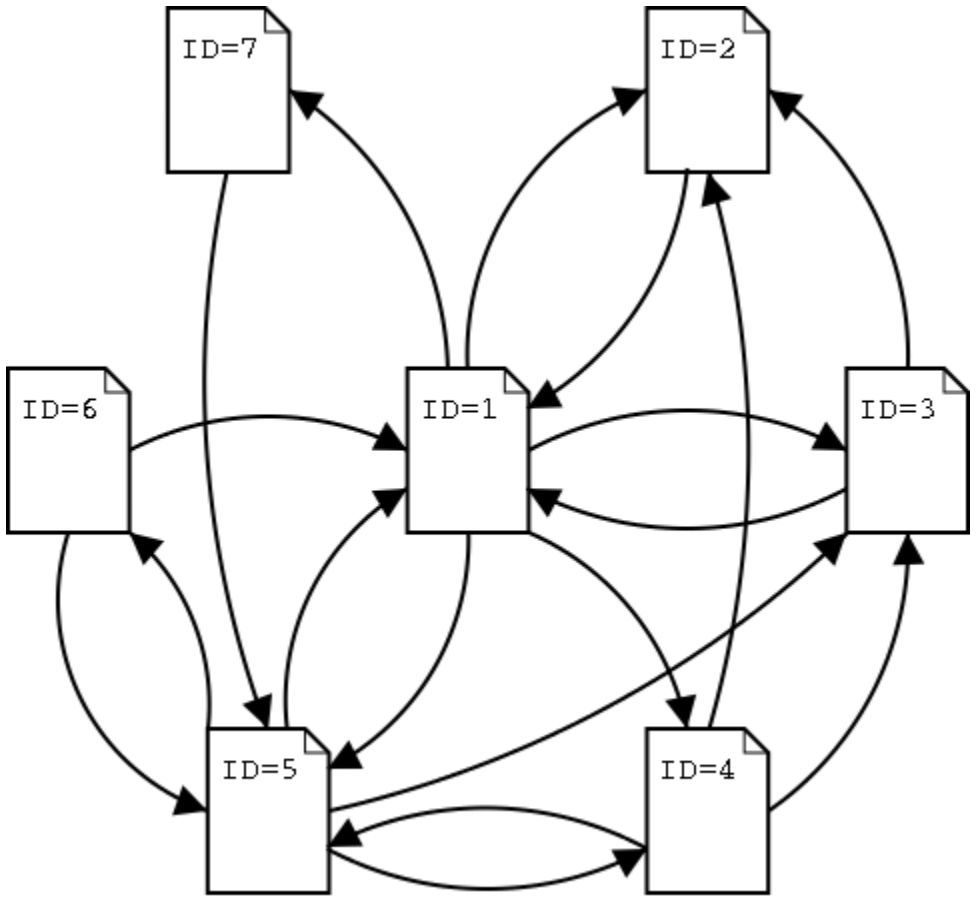
  - 其中：
    - PR(pi)为网页pi的PageRank值
    - PR(pj)为指向网页pi的某个网页pj的PageRank值
    - L(pj)为网页pj发出的链接数量
    - d为阻尼系数，取值在0-1之间
    - N为网页总数，M(pi)为链入pi的页面集合

- **PageRank的实现过程**

- PageRank的迭代计算过程：
  - 采用近似迭代算法计算PageRank值
    - 首先给每个网页赋予一个初值，例如1/N
    - 然后，利用之前的公式进行迭代有限次计算，得到近似结果
- 先前提到的论文显示，实际进行大约100次迭代才能得到整个网络的PageRank。
- 中等规模的网站（如论文中提到的26M个网站），需要数小时完成这一过程

- **PageRank的计算实例**

链接源I D	链接目标 ID
1	2,3 ,4,5, 7
2	1
3	1,2
4	2,3,5
5	1,3,4,6
6	1,5
7	5



- **PageRank的计算实例**

- 从邻接矩阵到跳转矩阵
  - 转置后，将各个数值除以所在列的非零元素数（即出边数量）

$$A=\begin{bmatrix} 0, & 1, & 1, & 1, & 1, & 0, & 1; \\ 1, & 0, & 0, & 0, & 0, & 0, & 0; \\ 1, & 1, & 0, & 0, & 0, & 0, & 0; \\ 0, & 1, & 1, & 0, & 1, & 0, & 0; \\ 1, & 0, & 1, & 1, & 0, & 1, & 0; \\ 1, & 0, & 0, & 0, & 1, & 0, & 0; \\ 0, & 0, & 0, & 0, & 1, & 0, & 0; \end{bmatrix} \Rightarrow R=\begin{bmatrix} 0, & 1, & 1/2, & 0, & 1/4, & 1/2, & 0; \\ 1/5, & 0, & 1/2, & 1/3, & 0, & 0, & 0; \\ 1/5, & 0, & 0, & 1/3, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 0, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 1/3, & 0, & 1/2, & 1; \\ 0, & 0, & 0, & 0, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 0, & 0, & 0, & 0; \end{bmatrix}$$

- **PageRank的计算实例**

- 开始计算，假设所有PageRank值初始均为1/N

$$R=\begin{bmatrix} 0, & 1, & 1/2, & 0, & 1/4, & 1/2, & 0; \\ 1/5, & 0, & 1/2, & 1/3, & 0, & 0, & 0; \\ 1/5, & 0, & 0, & 1/3, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 0, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 1/3, & 0, & 1/2, & 1; \\ 0, & 0, & 0, & 0, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 0, & 0, & 0, & 0; \end{bmatrix} + M=\begin{bmatrix} 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \end{bmatrix}$$

以3号节点为例，从先前的图可以看到，3号节点有三条入边，分别来自1、4、5号

同时，1、4、5号节点又各自有5、3、4条出边

根据之前的迭代式我们知道，**假设d=1时**，有R3 = R1 / 5 + R4 / 3 + R5 / 4

这个公式恰好等于上述R矩阵第三行 与 M向量 的内积

- **PageRank的计算实例**

- 因此，我们采用 R·M 来进行迭代计算每个节点的PageRank值

$$R=\begin{bmatrix} 0, & 1, & 1/2, & 0, & 1/4, & 1/2, & 0; \\ 1/5, & 0, & 1/2, & 1/3, & 0, & 0, & 0; \\ 1/5, & 0, & 0, & 1/3, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 0, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 1/3, & 0, & 1/2, & 1; \\ 0, & 0, & 0, & 0, & 1/4, & 0, & 0; \\ 1/5, & 0, & 0, & 0, & 0, & 0, & 0; \end{bmatrix} + M=\begin{bmatrix} 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \\ 1/7 \end{bmatrix}$$

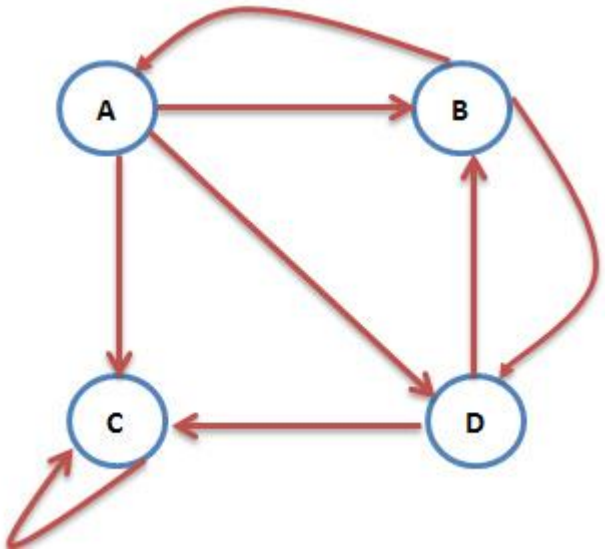
某轮迭代后，若M与M’对应维元素的差值高于阈值则继续迭代，直至收敛

当前M与M’差值过高，继续迭代

$$R \bullet M = \begin{bmatrix} 0.321 \\ 0.148 \\ 0.148 \\ 0.064 \\ 0.148 \\ 0.036 \\ 0.029 \end{bmatrix} = M'$$

- **PageRank中的两类特殊情况**

- PageRank的计算过程，实际上是一个马尔科夫过程
  - 如果计算中出现陷阱节点或终止节点，如何处理?
    - 陷阱节点：只有一条指向自己的边，没有其他出边
    - 终止节点：没有任何出边，如同黑洞
- 此外，孤立节点的存在也会产生一定影响
  - 没有任何入边，单纯使用马尔科夫链无法跳转
    - 仅有初始概率，不再更新，也不影响其他节点

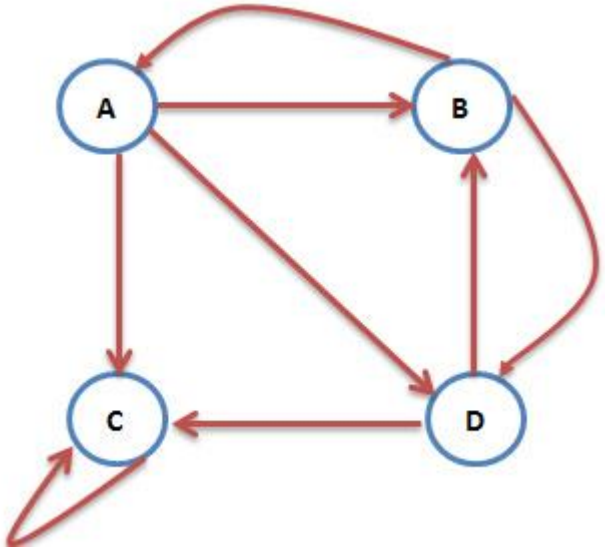


- **PageRank中的两类特殊情况**

- 如何解决这几类特殊情况：Restart机制
  - 回顾PageRank的公式

$$\mathrm{PR}(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

  - 其中的(1-d)/N的部分，相当于以一定概率重新选择起点
    - 此时，所有节点以一定等概率被选中作为新起点
    - 由此，跳出了陷阱和黑洞的干扰（所有节点全联通）
    - d一般选择为0.85左右（Google的选择）



- **PageRank中的收敛性问题**

- 如前所述，PageRank的计算过程，实际上是一个马尔科夫过程。

$$\mathrm{PR}(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

  - 取$A = dM + [(1-d)/N]ee^T$，其中M为跳转矩阵，e为所有元素都为1的列向量
  - 则有$P_{n+1} = AP_n$，形成马尔科夫过程

- 前述论文已从理论上证明，不论初始PageRank值如何选取，这种算法都保证了网页排名的估计值能收敛到他们的真实值。如何保证?

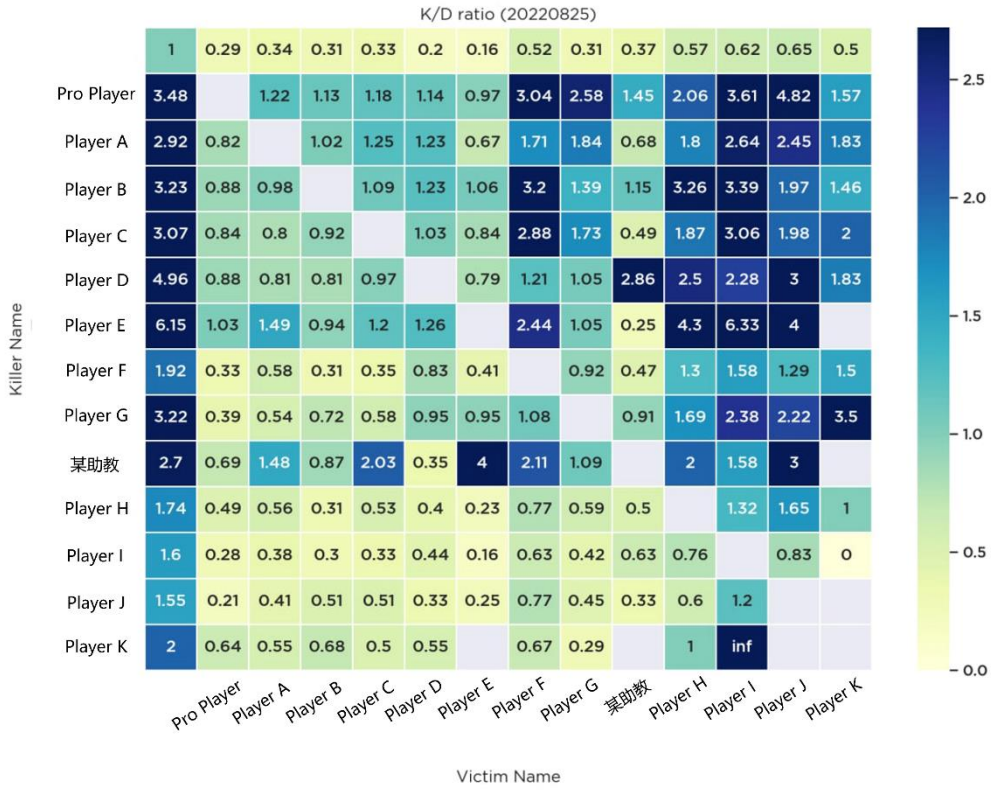
- **PageRank中的收敛性问题**

- 马尔科夫过程的三个收敛条件：
  - 转移矩阵A为马尔科夫矩阵
    - A矩阵所有元素都大于等于0，并且每一列的元素和都为1
  - 转移矩阵A为不可约的
    - 当图是强连通时，A为不可约，而Restart保障了这一条件
  - 转移矩阵A为非周期的
- 这三个条件，PageRank算法都满足，因此保障了其收敛性。

• PageRank的拓展应用

PageRank不止可以用来给网页排序。

例如，助教在和同学联机打FPS游戏时的发现：



	1	0.29	0.34	0.31	0.33	0.2	0.16	0.52	0.31	0.37	0.57	0.62	0.65	0.5
Pro Player	3.48		1.22	1.13	1.18	1.14	0.97	3.04	2.58	1.45	2.06	3.61	4.82	1.57
Player A	2.92	0.82		1.02	1.25	1.23	0.67	1.71	1.84	0.68	1.8	2.64	2.45	1.83
Player B	3.23	0.88	0.98		1.09	1.23	1.06	3.2	1.39	1.15	3.26	3.39	1.97	1.46
Player C	3.07	0.84	0.8	0.92		1.03	0.84	2.88	1.73	0.49	1.87	3.06	1.98	2
Player D	4.96	0.88	0.81	0.81	0.97		0.79	1.21	1.05	2.86	2.5	2.28	3	1.83
Player E	6.15	1.03	1.49	0.94	1.2	1.26		2.44	1.05	0.25	4.3	6.33	4	
Player F	1.92	0.33	0.58	0.31	0.35	0.83	0.41		0.92	0.47	1.3	1.58	1.29	1.5
Player G	3.22	0.39	0.54	0.72	0.58	0.95	0.95	1.08		0.91	1.69	2.38	2.22	3.5
某助教	2.7	0.69	1.48	0.87	2.03	0.35	4	2.11	1.09		2	1.58	3	
Player H	1.74	0.49	0.56	0.31	0.53	0.4	0.23	0.77	0.59	0.5		1.32	1.65	1
Player I	1.6	0.28	0.38	0.3	0.33	0.44	0.16	0.63	0.42	0.63	0.76		0.83	0
Player J	1.55	0.21	0.41	0.51	0.51	0.33	0.25	0.77	0.45	0.33	0.6	1.2		
Player K	2	0.64	0.55	0.68	0.5	0.55		0.67	0.29		1	inf		

PageRank →

Pro Player	0.105
助教	0.105
Player D	0.097
Player E	0.096
Player B	0.094
Player A	0.086
Player C	0.084
Player G	0.074
Player F	0.052
Player H	0.047
Player K	0.042
Player J	0.038
Player I	0.038
Player K(bot)	0.034

- **PageRank的拓展模型：个性化PageRank**

- PageRank在衡量网页的整体权威性方面具有显著的意义
- 然而，用户的个性化因素却没有在PageRank中得以体现
  - 每个用户有自己特定的偏好，绝大多数网页并不会被用户浏览
  - 一种解决方案是将用户偏好排序与PageRank值结合起来
    - 从用户喜爱的网页中挑选更权威的，较为简便，但可能效果一般
  - 另一种方案，模拟用户自己的网页浏览行为，从而计算个性化的权威性
    - 个性化PageRank（Personalized PageRank）

- **PageRank的拓展模型：个性化PageRank**

- 早在两位大神1999年的论文中，就提到了Personalized PageRank的思想
- 在个性化PageRank中，用户视作网络中的一个虚拟节点
  - 首先，任何跳转行为均从用户开始，即用户自身作为浏览行为的起点
  - 其次，从用户出发的跳转概率不一定视作均等（如1/N）
    - 可根据用户对不同网页的偏好决定其概率
    - 基本公式形如 $\boldsymbol{r} = (1-d)\boldsymbol{M}\boldsymbol{r} + d\boldsymbol{v}$ 的形式
      - v 体现了用户偏好，可视作不同网站在用户偏好下的点击概率

- **PageRank的拓展模型：主题敏感PageRank**

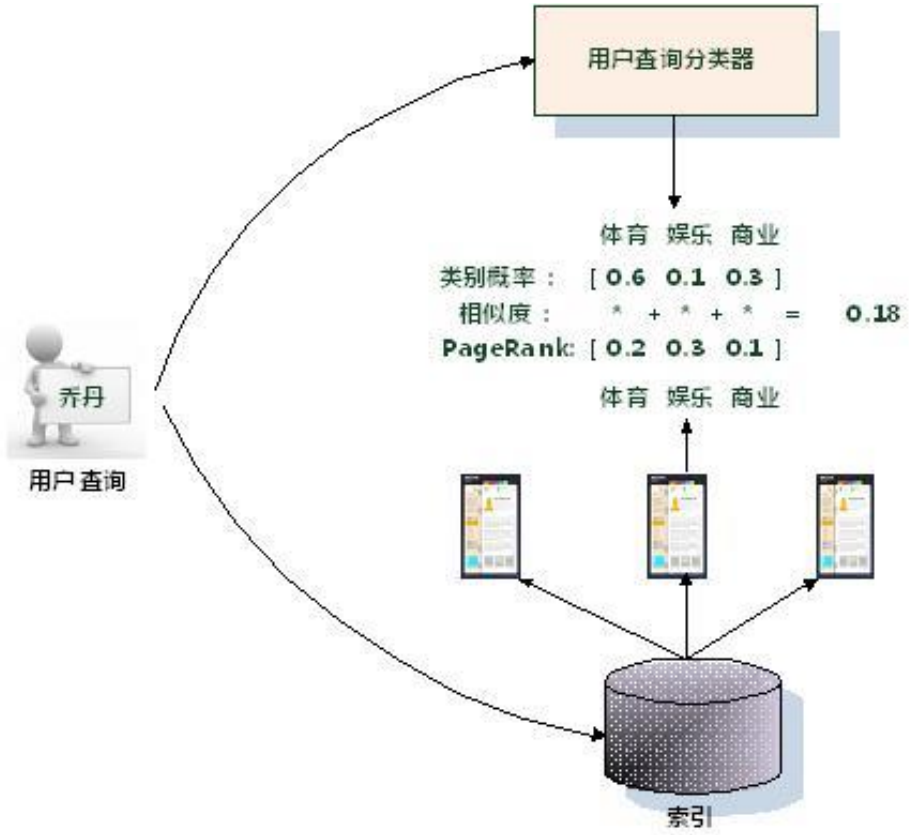
- 个性化PageRank固然体现了用户偏好，然而代价过于巨大
  - 为每一个用户单独计算其PageRank值事实上是个Mission Impossible
- 如何既在一定程度上考虑偏好因素，又减轻计算资源的负担？
  - 一个折中的方案是，以主题为中介，为特定的主题计算相应的PageRank
    - TH Haveliwala, Topic-sensitive pagerank, WWW‘02，已引用3703次
    - 某种意义上，这一模型体现了“仅有同行的评价才有价值”的思想
      - 隶属不同主题的网站，其相互引用和推荐的作用将被削弱

• PageRank的拓展模型：主题敏感PageRank

- Topic-sensitive PageRank与基本PageRank的区别
  - 首先，每个网页隶属于某个特定主题
    - 在论文中，作者参考Open Directory Project划分了16个类别
  - 其次，在计算部分，区别主要在于起始节点和Restart部分
    - 对于某个Topic来说，起点仅限于隶属于该Topic的网站
    - (1-d)e/N更改为了(1-d)s/|s|，s为一个N维向量
      - 如果某个网站隶属于该主题，则 s 中的该维为1，反之为0
      - |s| 表示 s 中为 1 的元素的个数。

• PageRank的拓展模型：主题敏感PageRank

- Topic-sensitive PageRank的输出
  - 最终，每个网站将得到一个Topic-sensitive的向量。
    - 即使不属于某个Topic，网页也可以通过PageRank获得相应的数值。
  - 同样，用户的需求也将表示为Topic向量
  - 两个向量的内积可以反应这一网站在用户特定主题需求下的权威性。

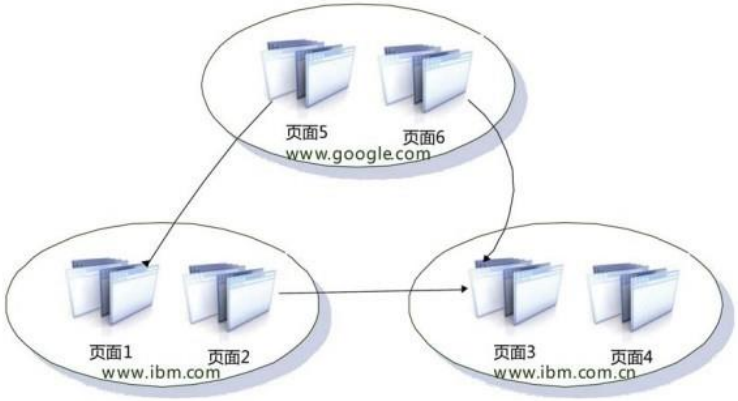


- **PageRank的拓展模型：Hilltop算法**

- 如前所述，主题敏感PageRank体现了“同行的评价更有价值”的思想
- 然而，即使在这种约束下，PageRank仍有被滥用的危险
  - 过分依赖PageRank衡量网页权威性，可能会鼓励那些试图通过增加大量无效链接而提高PageRank值的作弊手段。
- 如何保障这一指标的可靠性?
  - 一种启发式方法：预先选定一批“高质量网页”，然后通过这批网页发出的链接，来判断其他网页的价值
  - Hilltop算法，由Krishna Baharat 在2000年研究，于2001年申请专利

- **PageRank的拓展模型：Hilltop算法**

- Hilltop算法的基本概念之一：非从属组织网页
  - 满足以下两种情况，将被视作从属组织网页
    - 主机IP地址的前三个字段相同，如182.61.200.X （百度）
    - URL中的主域名段相同，如 XXX.ustc.edu.cn



- **PageRank的拓展模型：Hilltop算法**

- Hilltop算法的基本概念之二：专家页面
  - 所谓专家页面，即与某个主题相关的高质量页面
  - 专家页面一般通过如下启发式规则进行挑选
    - 至少包括K个出边，K由人为指定
    - K个出边指向的所有页面相互之间的关系都符合“非从属组织页面”
  - 专家页面之外的部分被称作“目标页面集合”
    - 即需要进行PageRank计算，判定其权威性的部分

- **PageRank的拓展模型：Hilltop算法**

- Hilltop算法的基本流程
  - 首先，根据用户的查询需求，选定相关的专家页面，并计算其相关性
  - 其次，遵循PageRank算法，对目标页面集合进行排序
    - 即将专家页面的相关性得分，传递到其他的目标页面上去
  - 最后，整合最相关的专家页面和得分较高的目标页面返回给用户
- 该方法能够一定程度上避免恶意添加链接的作弊行为，同时保证搜索与主题的相关性。但结果严重依赖专家页面的搜索和确定，存在一定偏见。

- **PageRank的总结**

- PageRank是一种基于链接分析的全局网页排序算法
- 优点
  - 对于网页给出全局的重要性排序，并且可以离线完成以保障效率
  - 基础的PageRank算法本身独立于检索主题，可以实现结果的通用
- 缺点
  - 主题无关性，同时对于恶意链接、广告链接等缺乏区分
  - 旧网页得分更高，因为新网页往往少有入链（论文同样如此！）
  - 一般不能单独用于排序，需要与相关性排序方法相结合
  - 效率问题（很多图问题的通病）

本节目录

- 排序学习算法
  - Pointwise 算法
  - Pairwise 算法
  - Listwise 算法
- **网页权威性计算**
  - PageRank及其衍生模型
  - **HITS算法与网页角色区分**

- **从PageRank到HITS算法**

- Hilltop算法尝试了网页角色的区分，但仍有进一步的空间
  - 即使通过个性化或主题敏感进行了修正，但对网页的功能仍不区分
    - 例如，专业信息站点与门户网站，在提供信息的种类上区别很大
    - 因此，将对外链接的输出和功能视作等价不符合实际情况

- **从PageRank到HITS算法**

- 在PageRank提出的同一年（1998），康奈尔大学的Jon Kleinberg（又一尊大神）提出了Hyperlink – Induced Topic Search（HITS）
  - Kleinberg博士认为，既然搜索是用户发起的提问，那么页面的重要性衡量就应该遵循用户的检索意图

Kleinberg J M. Authoritative sources in a hyperlinked environment, In Proceedings of the ACM-SIAM symposium on discrete algorithms (SODA 1998). 1998. （引用3308次）

Kleinberg J M. Authoritative sources in a hyperlinked environment[J]. Journal of the ACM (JACM), 1999, 46(5): 604-632. （引用11707次）

- **HITS算法的两个核心概念**

- 权威（Authority）网页与枢纽（Hub）网页的区分
- 权威网页：指某个领域或某个话题相关的高质量网页
  - 如科研领域的中科院之声，视频领域的~~优酷与爱奇艺~~Bilibili等
- 中心网页：类似中介，指向了很多高质量的权威网页
  - 如“hao123”，各个浏览器自带的首页（手动滑稽）
- HITS的目的即在海量网页中找到并区分这些与用户查询主题相关的高质量“Authority”与“Hub”网页，尤其是“Authority”（因为更能满足用户的信息需求）

- **HITS算法的两个基本假设**

- 基本假设与核心概念相互对应
  - 基本假设1：好的Authority会被很多好的Hub指向

$$\forall p, a(p) = \sum_{i=1}^{n} h(i)$$

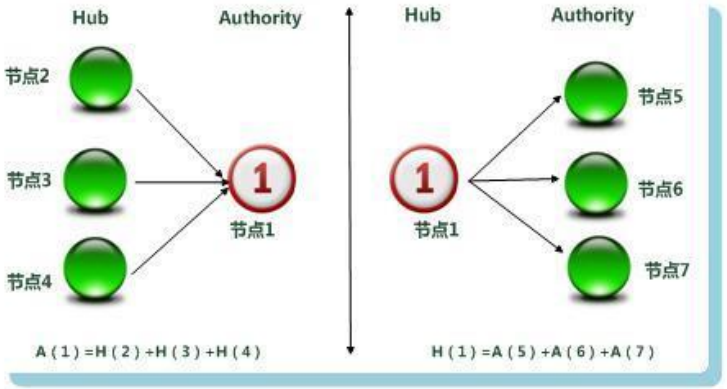
  - 基本假设2：好的Hub会指向很多个好的Authority

$$\forall p, h(p) = \sum_{i=1}^{n} a(i)$$

  - 因此，在HITS算法中，每个网页需要计算两个权威值

- **HITS算法的计算过程**

- 基于先前的基本假设，HITS的计算过程如下：
- 首先，根据关键字获取与查询最相关的少数页面，及与这些页面有链接关系的页面，作为待选集合
- 其次，对所有网页的 a(p) 与 h(p) 进行初始化，可都设为1
- 最后，迭代计算两个步骤，即基本假设所对应的两个公式
  - 重复这一步，直到最终收敛为止
    - 注意每一步中的归一化过程！
  - 输出Authority或Hub值较高的页面

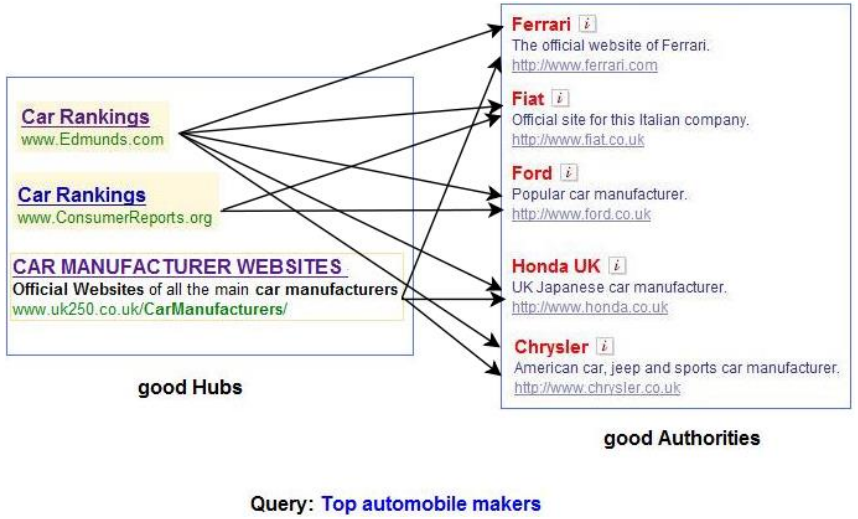


- **HITS算法的计算过程**

- 基于先前的基本假设， HITS的计算过程如下：
  - 假定邻接矩阵为M，Authority向量为a，Hub向量为h
  - 则有如下迭代式：
    - $a_{k+1} = M^T h_k$， $h_{k+1} = M a_{k+1}$
  - 或者，可采用如下迭代式：
    - $a_{k+1} = (M^T M)^k M^T a_0$， $h_{k+1} = (M M^T)^{k+1} h_0$
    - 其中，a_0, h_0为Authority/Hub向量的初始值，可设为全1向量

HITS算法的优缺点

- 相比于PageRank，HITS算法是一种能够区分网页功能的排序算法
- 优点
  - 更好地描述互联网组合特点
  - 主题相关，因此可以单独用于网页排序
- 缺点
  - 需要在线计算，时间代价较大
  - 对链接结构变化敏感，且依然可能受到“链接作弊”的影响



Query: Top automobile makers

本章小结

网页排序（下）

- 排序学习算法
  - Pointwise 算法
  - Pairwise 算法
  - Listwise 算法
- 网页权威性计算
  - PageRank及其衍生模型
  - HITS算法与网页角色区分