第12章 链接分析算法

《人工智能算法》

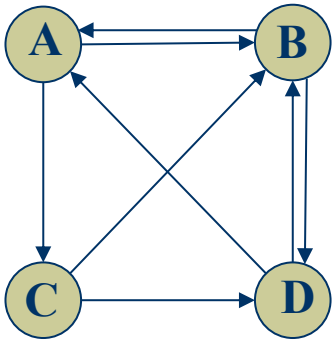
清华大学出版社
2022年7月

提纲

- 引例
- 链接分析概述
- PageRank算法
- 基于MapReduce的PageRank算法
- 总结

引例 (1)

- **有向图表示网页之间的联系，如何衡量各网页的重要程度？**



基于以下两个假设：

✓ 数量： 入链数量越多，页面越重要。

✓ 质量：质量越高的页面指向某一页面，则该页面越重要。

引例 (2)

- **步骤1：** $n \times n$矩阵**M**表示每个页面的入链与出链值，并初始化各页面的初始分布$\mathbf{R}_0$。

$$\mathbf{M} = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \end{bmatrix}, \mathbf{R_0} = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

- **步骤2：** 每一轮迭代将会更新每个页面的权重值，迭代若干轮后，当 $\lim_{t \to \infty} \mathbf{M}^t \mathbf{R_0}$ 收敛，则该极限就表示为各网页的重要程度。

引例 (3)

- **步骤3：**计算各时刻访问各网页的概率分布

$$\mathbf{R_0}=\begin{bmatrix}0.25\\0.25\\0.25\\0.25\end{bmatrix},\ \mathbf{MR}_0=\begin{bmatrix}0.25\\0.375\\0.125\\0.25\end{bmatrix},\ \mathbf{M}^2\mathbf{R}_0=\begin{bmatrix}0.3125\\0.3125\\0.125\\0.25\end{bmatrix},\ \mathbf{M}^3\mathbf{R}_0=\begin{bmatrix}0.2813\\0.3438\\0.1563\\0.2188\end{bmatrix}$$

$$\mathbf{M}^4\mathbf{R}_0=\begin{bmatrix}0.2813\\0.3281\\0.1406\\0.25\end{bmatrix},\ \mathbf{M}^5\mathbf{R}_0=\begin{bmatrix}0.2891\\0.3359\\0.1406\\0.2344\end{bmatrix},\ \mathbf{M}^6\mathbf{R}_0=\begin{bmatrix}0.2852\\0.3320\\0.1445\\0.2383\end{bmatrix},\ \mathbf{M}^7\mathbf{R}_0=\begin{bmatrix}0.2852\\0.3340\\0.1426\\0.2383\end{bmatrix}$$

T轮的时间复杂度：$O(Tn^2)$

引例 (4)

- **步骤4：**收敛后的向量为

$$\mathbf{R} = \begin{bmatrix} 0.2852 \\ 0.3340 \\ 0.1426 \\ 0.2383 \end{bmatrix}$$

即为各网页的重要程度，可得出各网页重要性排序为：

$$\mathrm{B} > A > D > C$$

提纲

- 引例
- 链接分析概述
- PageRank算法
- 基于MapReduce的PageRank算法
- 总结

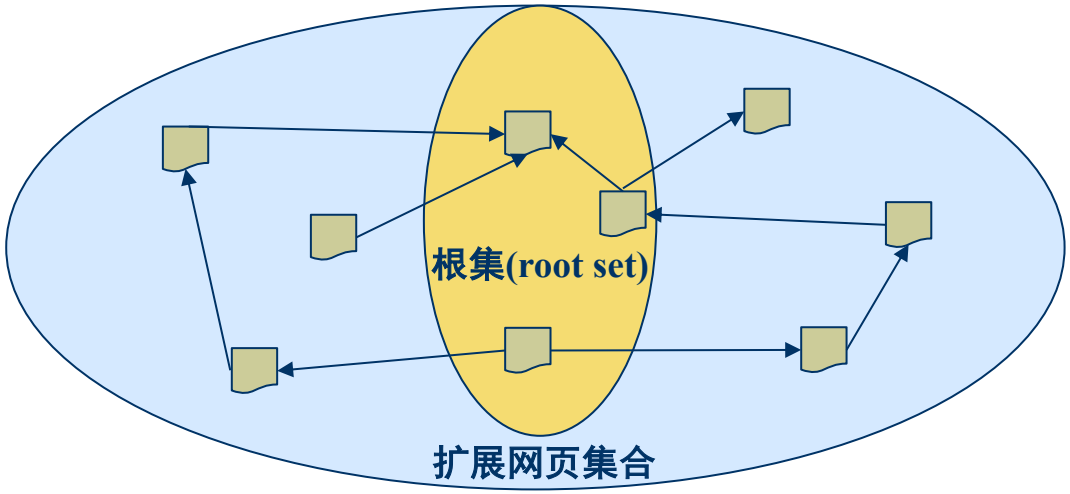
链接分析概述

- **链接分析：**运用数学分析和情报学等方法对图结构中的网络链接进行分析，以揭示图中节点的重要性，节点之间的关联信息及规律。
- **分类与应用**

分类	应用领域
情报链接分析（ISLA）	基于文献计量学的引文分析
计算机科学链接分析（CSLA）	网络动力学、链接与内容的关系、链接与信息检索、网络挖掘
社会科学链接分析（SSLA）	网络空间分析与超链接网络分析

主要的链接分析模型

- **随机游走（Random Walk）：**以PageRank及其改进算法为代表，对节点之间直接链接与远程链接两种方式进行分析。
- **子集传播（Subset Propagation）：**以Hilltop和HITS算法为代表，将图结构划分为子集并对特殊子集初始化，利用特殊子集与其他节点的链接关系将权值传递到其他节点。



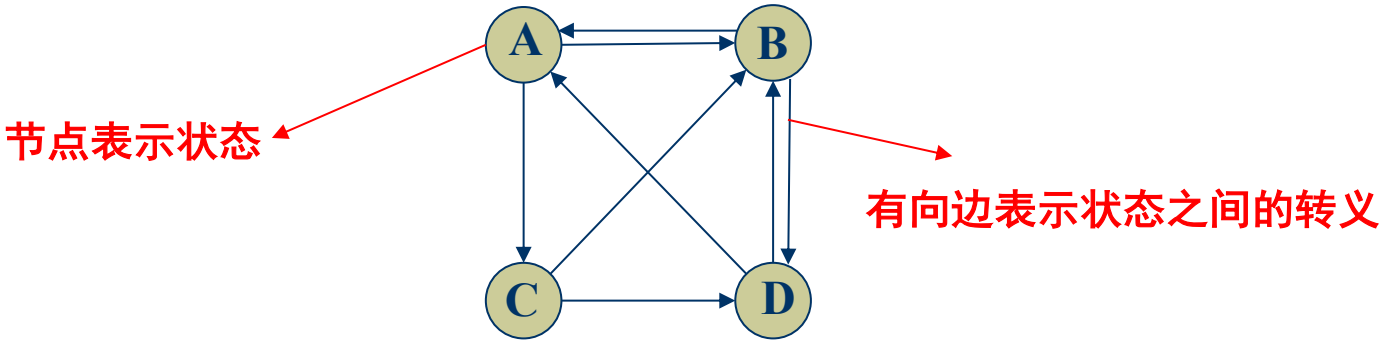
HITS算法

提纲

- 引例
- 链接分析概述
- PageRank算法
- 基于MapReduce的PageRank算法
- 总结

PageRank算法 (1)

- 以网页排序为例，PageRank算法可对所有网页进行重要性排序。
- 随机游走模型就是针对浏览网页的用户行为建立的抽象概念模型。



- **PageRank基本思想：**在给定有向图上初始化PageRank值及转移矩阵，定义随机游走模型，迭代更新节点状态值直至趋于稳定，每个节点的值为最终的PageRank值。

PageRank算法 (2)

- 含有n个节点的有向图，转移矩阵可表示为

$$\mathbf{M} = [m_{ij}]_{n\times n}$$

如果节点j有k个有向边链出，且节点i是其链出的一个节点，则第i行第j列的元素m_{ij}的值为$\frac{1}{\mathrm{k}}$，否则m_{ij}的值为0（$i, j = 1,2,\dots,n, i \neq j$）

- 转移矩阵**M**的性质：

$$m_{ij} \geq 0, \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij} = 1$$

PageRank算法 (3)

- 用n维列向量$\mathbf{R}_t$表示t时刻访问各节点的概率分布，则$t+1$时刻访问各节点的概率分布$\mathbf{R}_{t+1}$满足：

$$\mathbf{R}_{t+1} = \mathbf{M}\mathbf{R}_t$$

- 根据随机游走模型的思想，所有节点的PageRank值为：

$$\mathbf{R} = \left(d\mathbf{M} + \frac{1-d}{n}\mathbf{E}\right)\mathbf{R} = d\mathbf{M}\mathbf{R} + \frac{1-d}{n}\mathbf{Z}_n$$

转移矩阵访问各个节点的概率

完全随机访问各个节点的概率

d（$0 \le d \le 1$）为阻尼因子，一般取值0.85，Z_n为分量均为1的n为向量

增加d的动机？

PageRank算法 (4)

- 每个节点的PageRank值：

$$PR(v_i) = d\left(\sum_{v_j \in M(v_i)} \frac{PR(v_j)}{L(v_j)}\right) + \frac{1-d}{n} \quad (i = 1, 2, \ldots, n)$$

- ✓ $\mathbf{M}(v_i)$为指向v_i的节点集合
- ✓ $L(v_j)$为v_j的链出总数
- ✓ $PR(v_i) > 0$且满足$\sum_{i=1}^{n} PR(v_i) = 1$

PageRank算法 (5)

- **幂法求PageRank值的步骤**

$\mathbf{M}, d, \varepsilon, \mathbf{x}_0$ // 输入

$t \leftarrow 0$

Repeat

$\mathbf{A} \leftarrow d\mathbf{M} + \frac{1-d}{n}\mathbf{E}$ // 计算有向图的一般转移矩阵

$\boldsymbol{y}_{t+1} \leftarrow \mathbf{A}\mathbf{x}_t$

$\mathbf{x}_{t+1} \leftarrow \frac{\mathbf{y}_{t+1}}{||\mathbf{y}_{t+1}||}$ // 迭代计算，并将结果向量规范化

Until $||\mathbf{x}_{t+1} - \mathbf{x}_t|| < \varepsilon$

$\mathbf{R} \leftarrow \mathbf{x}_t$ // 停止迭代，得到各节点的概率分布

$\mathbf{R} \leftarrow \mathbf{R}/\text{sum}(\mathbf{R})$ // 归一化PageRank矩阵， sum(**R**)为矩阵**R**所有元素之和

Return **R**

时间复杂度为：$O(t(\varepsilon)n^2)$

PageRank算法 (6)

幂法求解网页的PageRank值示例

- **步骤1**：取阻尼系数d=0.85，t=0，初始化向量$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
- **步骤2**：计算有向图的一般转移矩阵**A**

$$\mathbf{A} = d\mathbf{M} + \frac{1-d}{n}\mathbf{E} = 0.85\times \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 1/2 \\ 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \end{bmatrix} + \frac{0.15}{4} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.0375 & 0.4625 & 0.0375 & 0.4625 \\ 0.4625 & 0.0375 & 0.4625 & 0.4625 \\ 0.4625 & 0.0375 & 0.0375 & 0.0375 \\ 0.0375 & 0.4625 & 0.4625 & 0.0375 \end{bmatrix}$$

PageRank算法 (7)

- **步骤3：**迭代计算并进行规范化处理

$$\mathbf{y}_1 = \mathbf{A}\mathbf{x}_0 = \begin{bmatrix} 1 \\ 1.425 \\ 0.575 \\ 1 \end{bmatrix}, \mathbf{x}_1 = \frac{1}{1.425}\begin{bmatrix} 1 \\ 1.425 \\ 0.575 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.7018 \\ 1 \\ 0.4035 \\ 0.7018 \end{bmatrix}$$

$$\mathbf{y}_2 = \mathbf{A}\mathbf{x}_1 = \begin{bmatrix} 0.8285 \\ 0.8732 \\ 0.4035 \\ 0.7018 \end{bmatrix}, \quad \mathbf{x}_2 = \frac{1}{0.8732}\begin{bmatrix} 0.8285 \\ 0.8732 \\ 0.4035 \\ 0.7018 \end{bmatrix} = \begin{bmatrix} 0.9488 \\ 1 \\ 0.4621 \\ 0.8037 \end{bmatrix}$$

PageRank算法 (8)

不断迭代并进行规范化处理，得到$\mathbf{x}_t(t = 0,1,\ldots,10)$的向量序列：

$$\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\begin{bmatrix}0.7018\\1\\0.4035\\0.7018\end{bmatrix},\begin{bmatrix}0.9488\\1\\0.4621\\0.8037\end{bmatrix},\cdots,\begin{bmatrix}0.2781\\0.3246\\0.1558\\0.2415\end{bmatrix},\begin{bmatrix}0.2781\\0.3246\\0.1557\\0.2416\end{bmatrix}$$

得到如下稳定分布：

$$\mathbf{R} = \begin{bmatrix}0.2781\\0.3245\\0.1557\\0.2416\end{bmatrix}$$

网页的排序结果为：$B > A > D > C$

PageRank算法 (9)

- **PageRank优点：**
  - ✓ 快速找出图中占主导地位的节点；
  - ✓ 计算过程可离线，提升检索效率，有利于快速响应。
- **PageRank缺点：**
  - ✓ 转移矩阵某一列为0时，排序失效；
  - ✓ 新加入节点的链接较少时，迭代过程中PageRank值会不断减少；
  - ✓ 对图中节点进行全局PageRank值计算，难以应对推荐系统类场景。

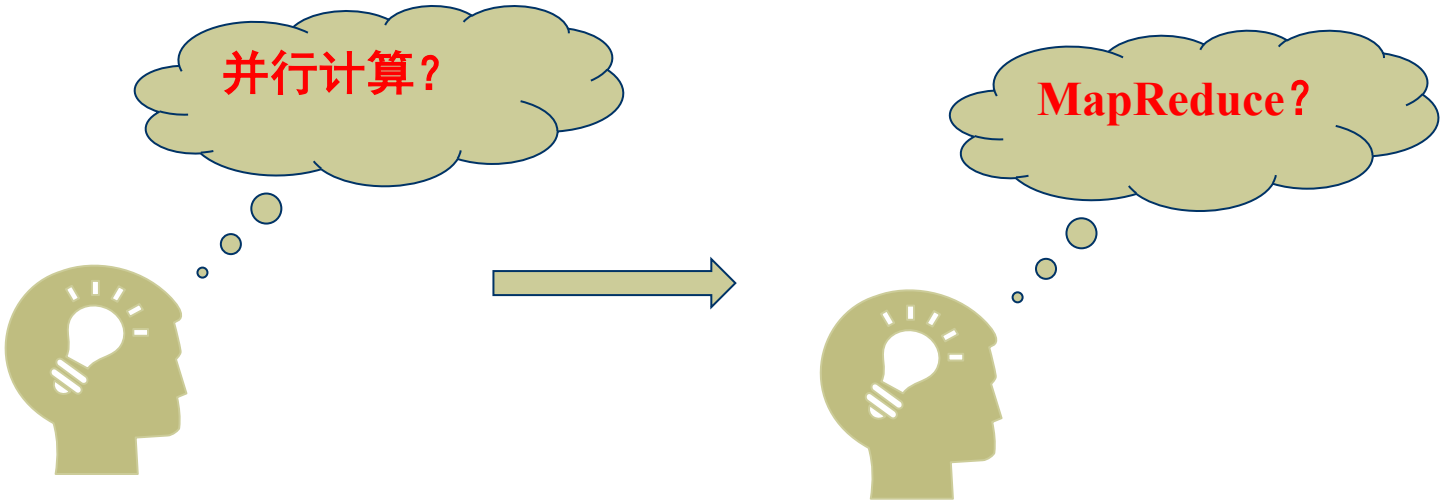
提纲

- 引例
- 链接分析概述
- PageRank算法
- 基于MapReduce的PageRank算法
- 总结

基于MapReduce的PageRank算法 (1)

问题：

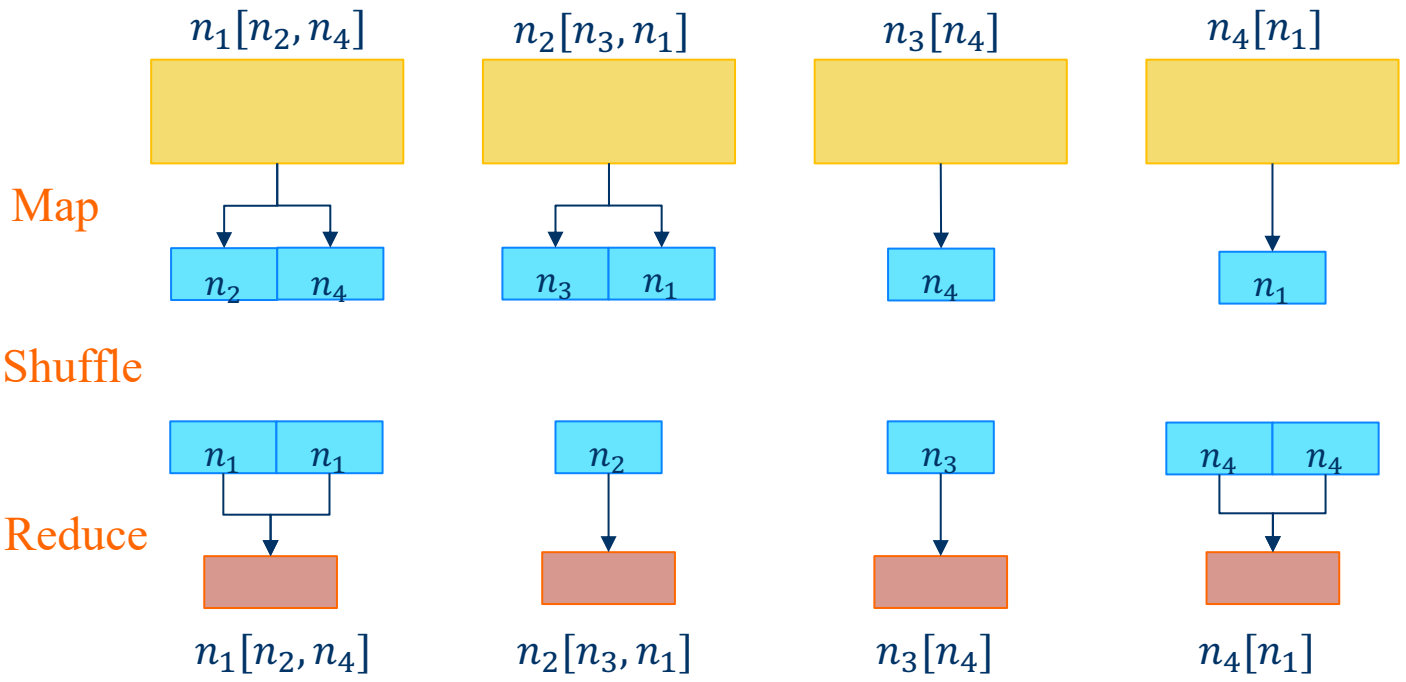
实际链接分析任务中，图中的节点规模往往比较大，如何高效的计算节点的PageRank值？



基于MapReduce的PageRank算法 (2)

◆ **基本思想**

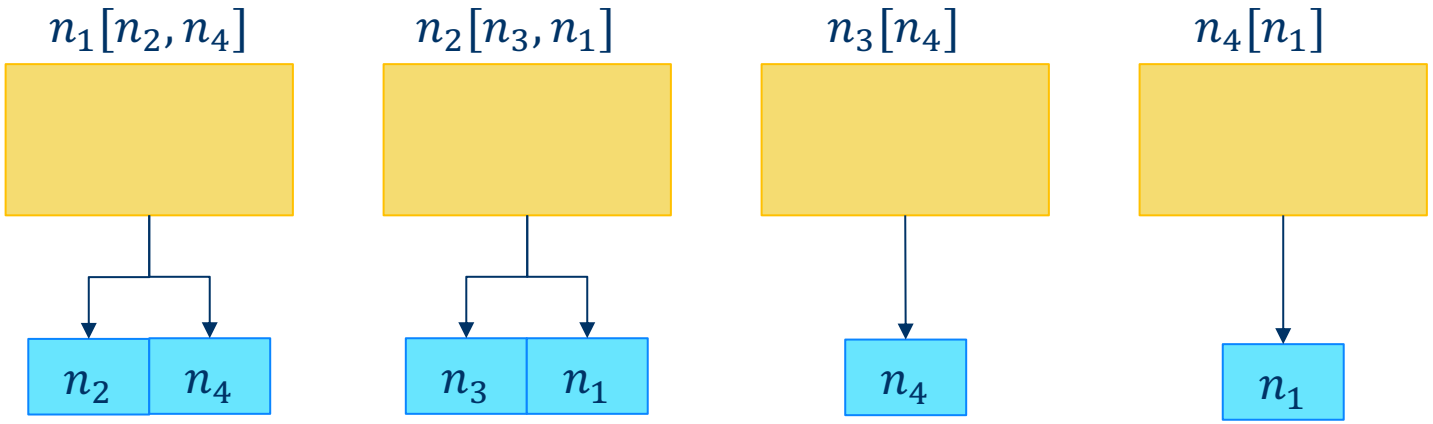
预先统计各网页之间的链接信息，并记录成<网页ID, (网页PageRank值, 网页链接列表)>，再通过Map和ReDuce函数进行迭代计算。



基于MapReduce的PageRank算法 (3)

◆ **Map阶段的任务与思想**

任务：根据形如<网页ID，（网页PageRank值，网页链出列表）>的输入信息产生两种键值对（<key, value>）



思想：计算每个链出网页被源网页“贡献”的PageRank值并生成<链出网页，PageRank贡献值>，由<网页，链出网页列表>和链出列表确定网页之间的链接情况。

基于MapReduce的PageRank算法 (4)

- **Map阶段算法步骤**

输入：

<*key*, **value**>，其中key为网页ID，**value**为（*PR*, **O**），PR为*key*对应网页的PageRank值，**O**为*key*对应网页的链出网页列表。

过程：

Emit(*key*, **O**) // 输出网页的链出列表

$O^{num} \leftarrow |\mathbf{O}|$ // O^{num}为**O**中网页链出的总数

For *i*=1 To O^{num} Do //第i个链出网页

Emit(*i*, PR/O^{num}) // 计算对每个链出网页的PageRank贡献值

End For

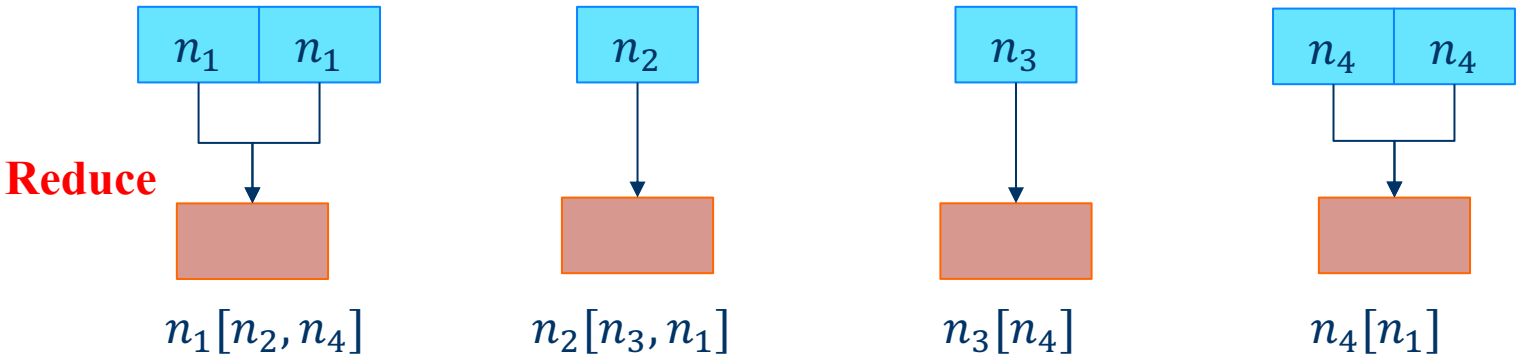
输出：

$\{key'_i, value'_i\}(i = 1, 2, \ldots, n)$；<key, **O**>

基于MapReduce的PageRank算法 (5)

◆ **Reduce阶段的任务与步骤**

任务：计算每个网页的PageRank值



思想：

由*key*从Map输出中找到所有对应的键值对并得到<*key*, **链出网页列表**>；
利用<*key*, *PageRank贡献值*>更新当前的PageRank值。

基于MapReduce的PageRank算法 (6)

- **Reduce阶段算法步骤**

输入：

$\{< key, value_j^r > (j = 1,2, \dots, M)\}$, M为网页的链入数, $< key, \mathbf{0} >$, d, n

过程：

$(key, \mathbf{values}) \leftarrow (key, \boldsymbol{collector}(\{< value_j^r > (j = 1,2, \dots, M)\}))$

$PR_{sum} \leftarrow 0$ // 初始化累计PageRank贡献值

For *value* In **values** Do // *value*为*key*对应网页的链入网页的PageRank贡献值

$PR_{sum} \leftarrow PR_{sum} + \text{value}$

End for

$PR^{new} \leftarrow d(PR_{sum}) + (1 - d)/n$ // 更新当前的PR值

$(key, value_r) \leftarrow (key, (PR^{new}, O))$

$Emit(key, value_r)$

输出：

$< \text{key}, value_r >$： $value_r$为(PR^{new}, O)， PR^{new}为新的PageRank值

基于MapReduce的PageRank算法 (7)

各阶段时间复杂度分析

- 假设有n个网页，平均每个网页的链出数和链入数分别为s和u，设MapReduce框架可同时处理m个Map或Reduce任务
- 每次迭代中的Map函数时间复杂度为：$O((n\times s)/m)$
- Reduce函数的时间复杂度为：$O((n\times u)/m)$
- 经t次迭代收敛，基于MapReduce的PageRank算法的时间复杂度为：

$$O(t\times n(s+u)/m)$$

提纲

- 引例
- 链接分析概述
- PageRank算法
- 基于MapReduce的PageRank算法
- 总结

总结

- 分析对象和潜在链接关系的多样化，人们对链接分析算法的效率、精度和鲁棒性等指标提出更高的要求。
- 针对不同的链接分析任务和性能要求，人们基于不同模型和架构对PageRank进行改进和扩展。例如，基于Spark框架的PageRank算法，从内存管理、减少内存I/O开销的角度进一步提高了计算效率。
- PageRank算法为基于链接关系的权重计算提供了可供参考的思路，具有通用性和普遍性，在学界和业界都得到广泛的应用。

结语

谢谢！