



电子科技大学
University of Electronic Science and Technology of China

Lecture 5 数据分析算法 (I)

数据关系



教学目标

- 认识常用的数据关系、数据关联规则、Web结点关联算法的原理，并比较不同的数据关系之间的区别。
- 掌握自然语言处理中的向量空间模型、词频计算、Jaccard相似系数、余弦相似度、Apriori关联规则、PageRank排名等各种方法的原理，并能够选择适当的方法解决数据科学中的问题。



内容概述

- **数据分析**是从海量数据中提取信息的过程，以机器学习算法为基础，通过模拟人类的学习行为，**获取新的知识或技能**，不断改善分析的过程。
- 机器学习从很**多学科**中吸收了重要的成果
统计学、人工智能、信息论、认知科学、计算复杂性和控制等。



电子科技大学
University of Electronic Science and Technology of China

内容概述

- 主要来源：**机器学习十大经典算法**，ICDM，2006
- 刘凡平，《大数据时代的算法—机器学习、人工智能及其典型实例》



内容概述

数据分析算法

数据关系: TF-IDF, 余弦相似, Apriori, PageRank

分类与聚类: Bayes, AdaBoost, SVM, KNN, K-Means, EM

决策: ID3, C4.5, CART



第5讲 数据关系

网页编号	网页标题	网页正文
1	教育部：支援中西部高考招生 不影响江苏湖北录取率
2	三亚天价打印一张A4图片50 元 官方已介入调查
3	7.5万平方米 这道玻璃幕墙太 嗨了

- 防止重复（或被复制、转载）的网页被搜索到。
- 网页价值分析，越是被转载或复制的网页，其重要性越高。



5.1 TF-IDF算法

- 自然语言处理中一个典型的应用是**在一堆文档中选择属于每个文本最具有代表性的词汇**。
 - 关键词、摘要
- 该算法的名称为**TF-IDF** (Term Frequency-Inverse Document Frequency) , 它是一种常用于检索系统的加权技术。



文档	文档中对应词语集合
A	努力 向前 奋斗 奋斗 使得 未来 更好
B	创新 创新 万众 智慧
C	奋斗 是 人生 的 一部分

- 什么是关键词？
 - 对文档的高度概括
 - 区分不同的文档



词袋模型

- 文本进入计算的**第一步**：转换为数值向量，转换为能作为计算的数量，即**把文本转换到数量空间**。
- 把文本映射到向量表示的空间中，这称为**向量空间模型**。
 - 词袋模型，是一种广泛用于自然语言处理和信息检索的词语模型。
 - 将若干词语直接放到一个“袋子”中，而**不考虑词语间的语法和相互顺序**。



词袋模型

文本	语句
A	小张 喜欢 打 篮球 和 打 羽毛球
B	小李 喜欢 打 羽毛球
词袋	小张、喜欢、打、篮球、和、羽毛球、小李

转换规则：向量值即为词在文档中出现的次数（词频）



词袋模型

文本	语句
A	小张(1)、喜欢(1)、打(2)、篮球(1)、和(1)、羽毛球(1)、小李(0)
A向量	1 1 2 1 1 1 0
B	小张(0)、喜欢(1)、打(1)、篮球(0)、和(0)、羽毛球(1)、小李(1)
B向量	0 1 1 0 0 1 1

词袋模型简单、易于理解，但是其假设句子和语法与词序无关，不符合自然语言的实际分布规则和含义，因此不能进行更深层次的语义处理。因此，它擅长的是与词频相关、忽略词序和语法的文本信息处理。



TF-IDF算法

- 基本思想：文档中每个词的重要性与它在当前文档中出现的次数成正比，但是与它在其他文件中出现的次数成反比。
 - 推论：倘若一个词语在某一文档中出现的频率很高，并且在其他文档集合中出现的频率很低，那么则认为该词语对文件A有一定的代表性，能够通过该词与其他文档形成较好的内容区分能力。



5.1 TF-IDF算法

所以如果特征空间坐标系取TF词频作为测度，就可以体现同类文本的特点。另外考虑到单词区别不同类别的能力，TF-IDF法认为一个单词出现的文本频数越小，它区别不同类别文本的能力就越大，

因此引入了逆文本频度IDF的概念。以TF和IDF的乘积作为特征空间坐标系的取值测度，并用它完成对权值TF的调整，调整权值的目的在于突出重要单词，抑制次要单词。



TF-IDF算法

$$\text{TF-IDF}_{i,j} = \text{TF}_{i,j} \times \text{IDF}_i = \frac{n_{i,j}}{\sum_{j=1} n_{i,j}} \times \log \frac{|D|}{|\{i: t_i \in d_i\}|}$$

文档	文档中对应词语集合
A	努力 向前 奋斗 奋斗 使得 未来 更好
B	创新 创新 万众 智慧
C	奋斗 是 人生 的 一部分

$$\text{TF}_{i,j} = 2 / (7+4+5) = 1/8, \quad \text{IDF}_{i,j} = \log (3/2)$$

$$\text{TF-IDF}_{i,j} = 1/8 * \log 1.5 = 0.22$$



TF-IDF的缺点

- 对短文本的处理和过长文本的处理不是很好。
- 忽视了文档中语义和语法的表达。
- 词语之间必须完全匹配，对相似词语或者词语的子词语不能进行有效的匹配。



5.2 余弦相似性

- 问题：怎么判断或度量两个数据（如文本）的相似性？
- 例如：

数据价值是一种数据艺术
算法价值是一种算法艺术



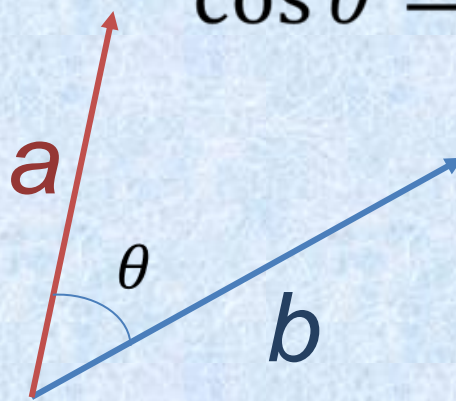
- 一个简单的指标是Jaccard系数，用于个体的特征属性通过符号度量或者布尔值标识，适合集合的计算。
- 对于文本，直观的想法是：两篇文章越相似，则它们词语的交集越多。

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

$$A \cup B$$



- Jaccard的理论基础支持不够，因为仅依靠是否出现去判定两者的相似度不够精准。
- 引入余弦相似性，通过余弦的方式计算相似度，将词语是否出现变更为词语在文本中的权重。


$$\cos \theta = \frac{a \times b}{\|a\| \times \|b\|} = \frac{\sum_{i=1}^n (x_i \times y_i)}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$



文本	A	B
内容	数据价值是一种数据艺术	算法价值是一种算法艺术
分词结果	数据 价值 是 一种 数据 艺术	算法 价值 是 一种 算法 艺术
向量集	数据 算法 价值 是 一种 艺术	
词频计算	数据(2)算法(0)价值(1)是(1)一种(1)艺术(1)	数据(0)算法(2)价值(1)是(1)一种(1)艺术(1)
特征向量	2 0 1 1 1 1	0 2 1 1 1 1
Jaccard相似度	$J(A, B) = \frac{ A \cap B }{ A \cup B } = \frac{ A \cap B }{ A + B - A \cap B } = \frac{4}{6} \approx 0.667$	
余弦相似度	$\cos \theta = \frac{a \times b}{\ a\ \times \ b\ } = \frac{\sum_{i=1}^n (x_i \times y_i)}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}} = 0.5$	



- **Jaccard相似性**只关心个体间特征属性是否相同，反映了样本交集与并集（总集）的差异
- A和B文本，相同词语较多，但两句的意思迥异，**余弦相似度**更符合实际情况。
- **优点**：简单而有效。
- **缺点**：数据维数高，**计算复杂度也越高**，不适应当前的（**数亿级别**）大数据。



5.4 Apriori算法

- 数据的**关联规则**用于从看似无关的海量历史数据中，挖掘出可能具有的价值信息，在商业活动中会利用数据之间的关系产生较大的商业价值。
- 购物篮分析
 - 典型案例：啤酒与尿布
- 关联规则反映的是两个或多个事物相互之间的**依存性和关联性**。



- **问题**：在超市中，如何根据客户的购买历史清单来优化货物的摆放？

交易序号	购买商品列表
1	牛奶、纸巾、矿泉水
2	饼干、纸巾、口香糖
3	牛奶、饼干、纸巾、口香糖
4	饼干、口香糖



- Apriori算法
 - 频繁项集算法
 - 应用广泛，超市商品关联分析、消费习惯分析……
 - 利用频繁项集的先验知识，不断地按照层次进行迭代，计算数据集中的所有可能的频繁项集



概念

- **项集**：即项的集合。 $\{\text{牛奶}, \text{面包}\}$ ，其中牛奶和面包为项， $\{\text{牛奶}, \text{面包}\}$ 为2项集。
- **关联规则**：形如 $X \rightarrow Y$ 的蕴涵表达式 (If...Then...)，其中 X 和 Y 是不相交的项集。
- **支持度**：项集 X 、 Y 同时发生的概率称之为关联规则的支持度， $s(X \rightarrow Y) = \frac{|X \cup Y|}{N}$
- **置信度**：项集 X 发生的情况下，则项集 Y 发生的概率， $c(X \rightarrow Y) = \frac{|X \cup Y|}{|X|}$



概念

- **最小支持度**：人为按照实际意义规定的阈值，表示项集在统计意义上的最低重要性。
- **最小置信度**：人为按照实际意义规定的阈值，表示关联规则最低可靠性。
 - 如果支持度与置信度同时达到最小支持度与最小置信度，则此关联规则为**强规则**。
- **频繁项集**：满足最小支持度的所有项集，称作频繁项集。



案例

项集 $N = \{\text{牛奶, 纸巾, 矿泉水, 饼干, 口香糖}\}$

数据集 $T = \{T_1, T_2, T_3, T_4\}$

数据项 $T_1 = (\text{牛奶, 纸巾, 矿泉水})$

$T_2 = (\text{饼干, 纸巾, 口香糖})$

$T_3 = (\text{牛奶, 饼干, 纸巾, 口香糖})$

$T_4 = (\text{饼干, 口香糖})$

目标：1) 如何构建频繁项集？

2) 如何从频繁项集中提取高置信度的规则？



案例

项集 $N = \{\text{牛奶}, \text{纸巾}, \text{矿泉水}, \text{饼干}, \text{口香糖}\}$

关联规则: 买了饼干又买了纸巾, 饼干 \rightarrow 纸巾

支持度 $s(\text{饼干} \rightarrow \text{纸巾}) = 2 / 4 = 0.50$

置信度 $c(\text{饼干} \rightarrow \text{纸巾}) = 2 / 3 = 0.67$

如果设定最小支持度 **0.50**

如果设定最小置信度 **0.65**

则频繁项集 $= \{T2, T3\}$

关联规则 (饼干 \rightarrow 纸巾) 同时满足最小支持度和最小置信度,
所以是强规则



关联规则挖掘算法

大多数关联规则挖掘算法通常采用的策略，将关联规则挖掘任务分解为如下两个主要的子任务：

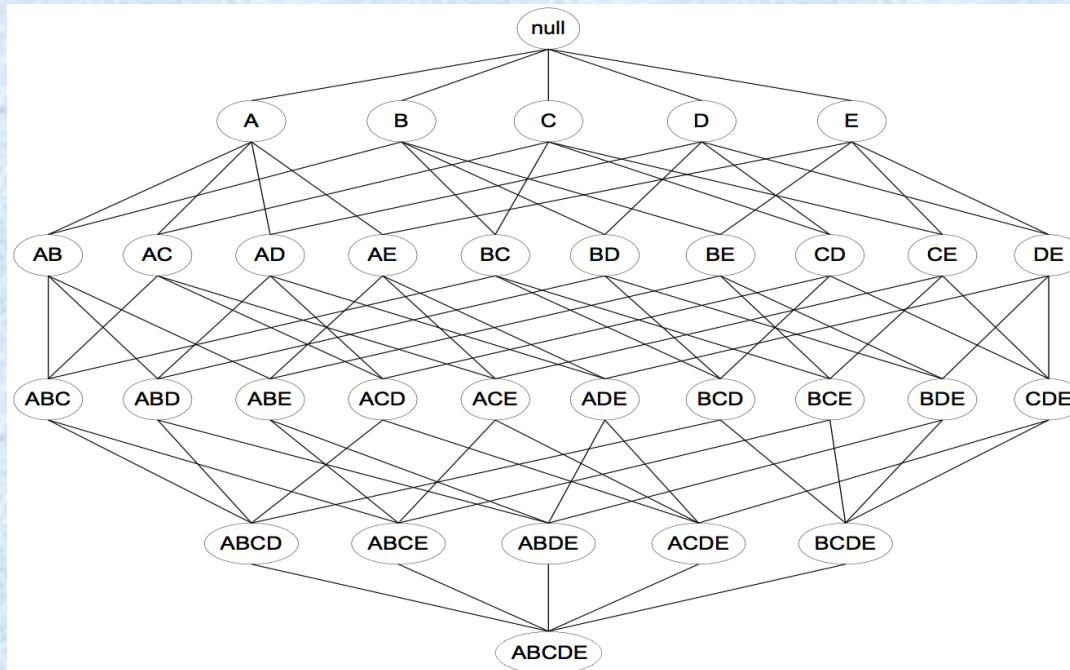
- **频繁项集构建**：其目标是寻找满足最小支持度阈值的所有项集，这个集称作频繁项集 (frequent itemset)
- **规则的产生**：其目标是从上一步发现的频繁项集中提取所有高置信度的规则，这些规则称作强规则 (strong rule)

通常，构建频繁项集所需的计算开销远大于产生规则所需的计算开销。



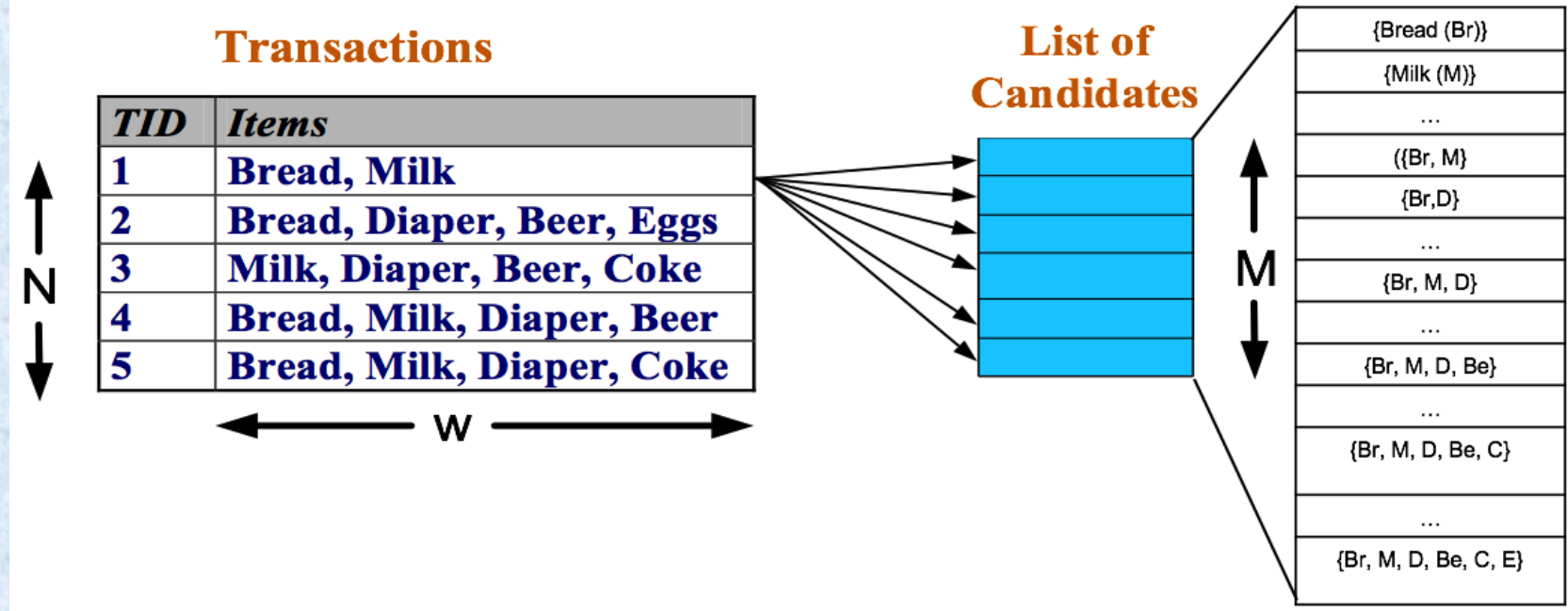
暴力搜索(Brute-force)方法

- 1) 列出所有的可能的规则组合
 - 2) 计算每一个组合的支持度和置信度
 - 3) 去除那些不满足最小支持度和最小置信度的规则
- 剩下的就是寻找到的关联规则。





暴力搜索方法成本



- 列出所有的可能的规则组合 (list of candidates)
- 将每一个组合去遍历Transactions集，计算该组合的支持度和置信度
- 计算成本为 $O(NMw)$, $M = 2^k - 1$

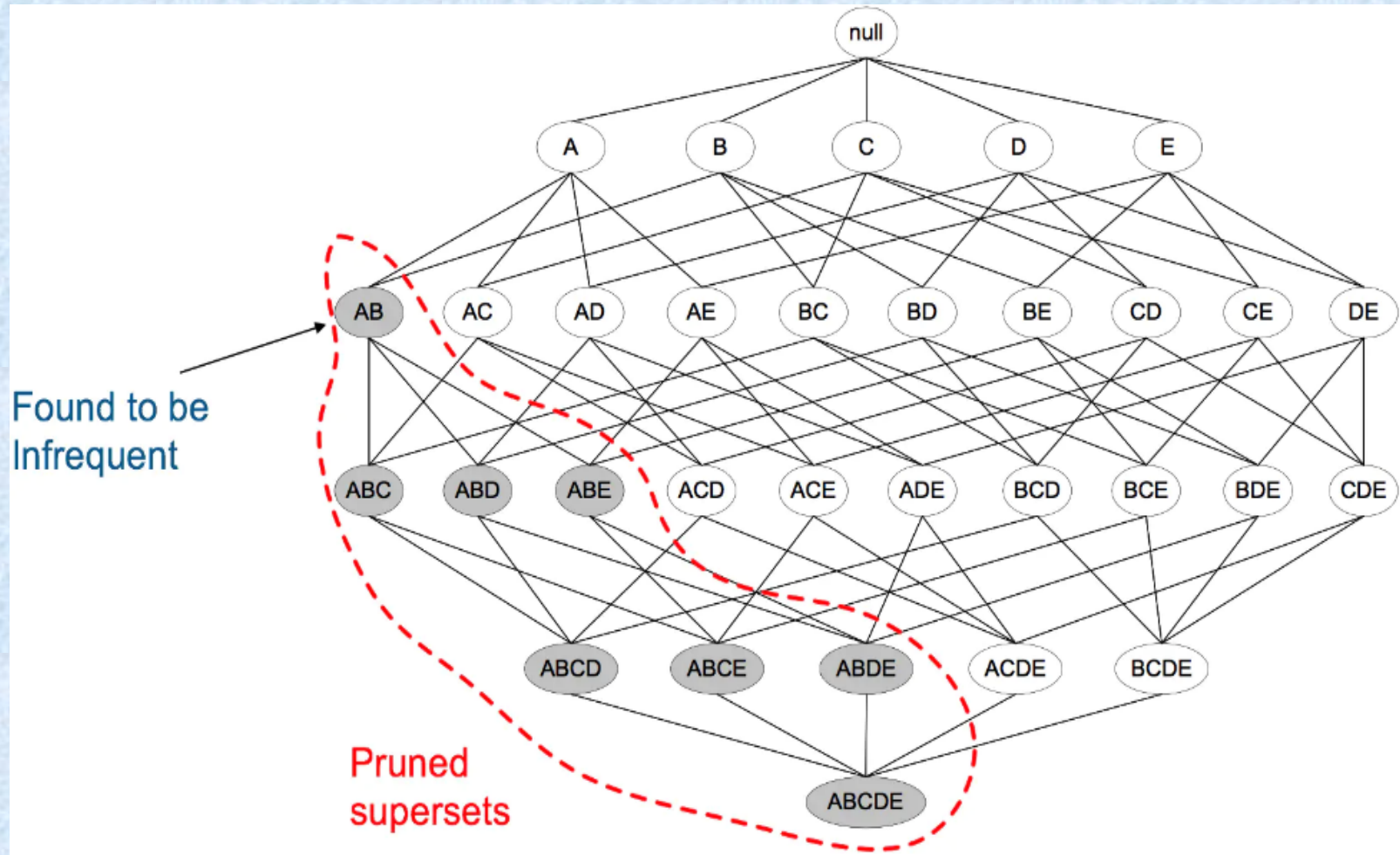


Apriori两大定理

- 定理1：如果一个集合是频繁项集，那么它的所有**子集**都是频繁项集合。
 - 例如，集合{苹果、梨子}是频繁项集，则子集{苹果}、{梨子}均属于频繁项集。
- 定理2：如果一个集合它不是频繁项集合，那么它的所有**超集**都不是频繁项集。
 - 例如，集合{苹果}不属于频繁项集，那么它的超集{苹果、梨子}也不属于频繁项集。



Apriori算法



上图表示当我们发现 $\{A, B\}$ 是非频繁集时，就代表所有包含它的超集也是非频繁集，即可以将它们都剪除。



Apriori算法

- 扫描历史数据，并对每项数据进行频率次数统计。
- 构建候选项集 C_1 ，并计算其支持度。
- 对候选项集的支持度进行筛选，从而形成频繁项集 L_1 。
- 对频繁项集 L_1 进行连接生成候选项集 C_2 。
- 重复上述步骤，最终形成频繁 K 项集或者最大频繁项集。



Apriori算例

T

序号	购买商品列表
1	牛奶、纸巾、矿泉水
2	饼干、纸巾、口香糖
3	牛奶、饼干、纸巾、口香糖
4	饼干、口香糖

扫描事务集

统计每个候选项的支持度

C_1

候选项集	支持度
牛奶	0.5
饼干	0.75
纸巾	0.75
矿泉水	0.25
口香糖	0.75

消除低于最小支持度的候选项，生成频繁项集1

L_1

频繁项集	支持度
牛奶	0.5
饼干	0.75
纸巾	0.75
口香糖	0.75

设定最小支持度 $C_{min} = 0.5$

消除矿泉水项，产生频繁集 L_1



Apriori算例

L_1

频繁项集	支持度
牛奶	0.5
饼干	0.75
纸巾	0.75
口香糖	0.75

根据频繁项集 L_1 ,
生成候选项集 C_2
(一项到二项排列
组合)



C_2

候选项集
牛奶、饼干
牛奶、纸巾
牛奶、口香糖
饼干、纸巾
饼干、口香糖
纸巾、口香糖



Apriori算例

T

序号	购买商品列表
1	牛奶、纸巾、矿泉水
2	饼干、纸巾、口香糖
3	牛奶、饼干、纸巾、口香糖
4	饼干、口香糖

扫描事务集 T ，
统计候选项集 C_2
的支持度

C_2

候选项集	支持度
牛奶、饼干	0.25
牛奶、纸巾	0.50
牛奶、口香糖	0.25
饼干、纸巾	0.50
饼干、口香糖	0.75
纸巾、口香糖	0.50

设定最小支持度 $C_{\min} = 0.5$

生成频繁项集2

L_2

频繁项集	支持度
牛奶、纸巾	0.50
饼干、纸巾	0.50
饼干、口香糖	0.75
纸巾、口香糖	0.50

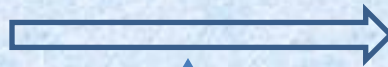


Apriori算例

L_2

频繁项集	支持度
牛奶、纸巾	0.50
饼干、纸巾	0.50
饼干、口香糖	0.75
纸巾、口香糖	0.50

根据频繁项集 L_2 ，
生成候选项集 C_3
(二项到三项排列
组合)



C_3

候选项集
饼干、纸巾、口香糖

{牛奶, 纸巾, 饼干}，它的子集{牛奶, 饼干}
不属于频繁项集，因此超集也不属于频
繁项集



Apriori算例

T

序号	购买商品列表
1	牛奶、纸巾、矿泉水
2	饼干、纸巾、口香糖
3	牛奶、饼干、纸巾、口香糖
4	饼干、口香糖

扫描事务集，统计候选项集 C_3 的支持度

C_3

候选项集	支持度
饼干、纸巾、口香糖	0.50

生成频繁项集 L_3

L_3

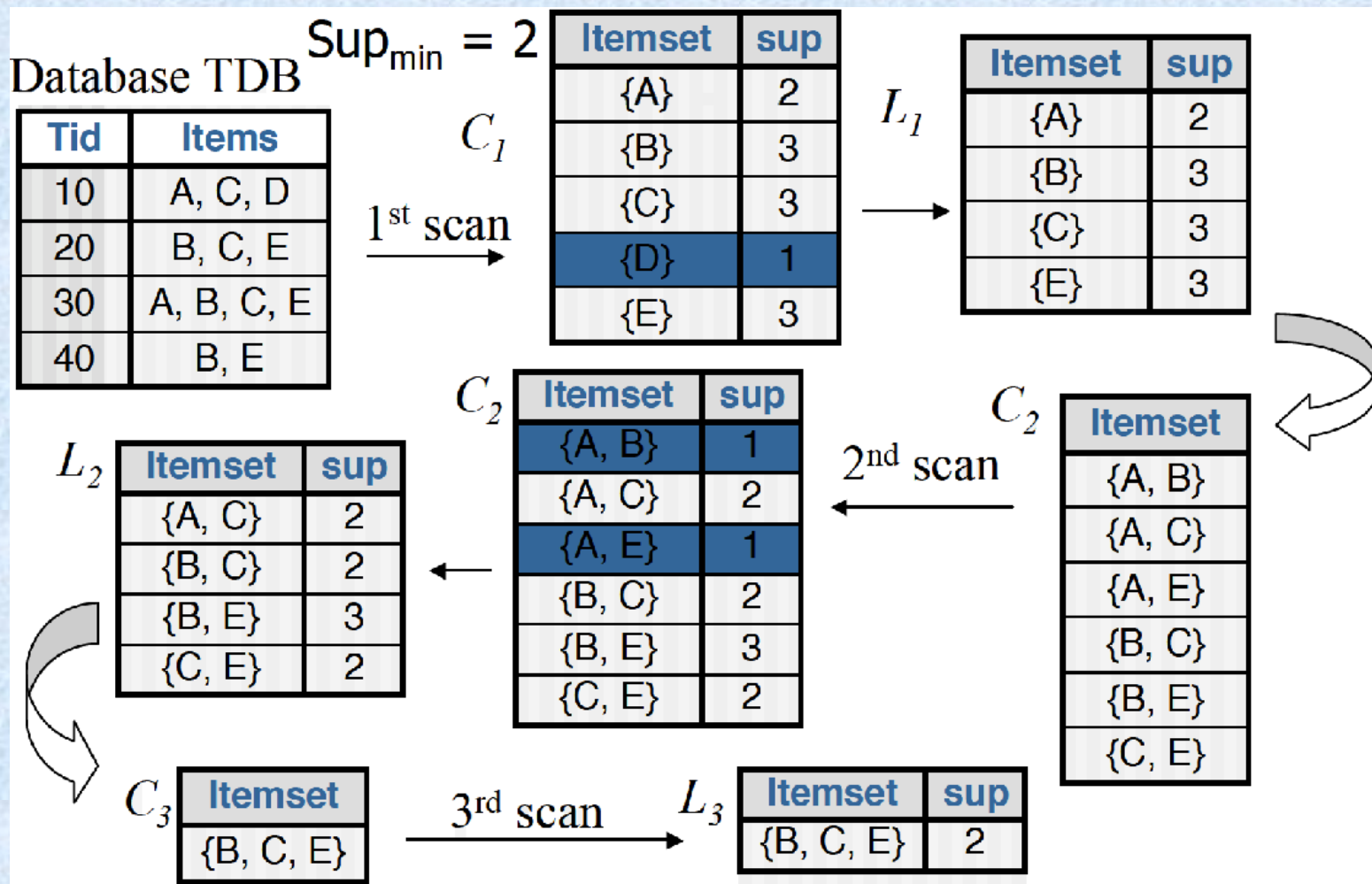
频繁项集	支持度
饼干、纸巾、口香糖	0.50

当得到了频繁项集 L_3 、 L_2 后，超市可以优先根据 L_3 ，将“饼干、纸巾、口香糖”放在同一货架或者连续货架中，然后再考虑根据 L_2 放置货架商品。



Apriori算例:自己算

算例: 最开始数据库里有4条交易项, {A、C、D}, {B、C、E}, {A、B、C、E}, {B、E}, 使用 $\text{min_support} = 2$ 作为最小支持度, 求最后筛选出来的频繁集。





Apriori算法缺点

- 产生候选项集时产生较多的组合，没有考虑将一些无关的元素排除后再进行组合。
- 每次计算项集的过程都会扫描原始的数据表，对于数据量较大的系统而言，重复扫描开销大。
- 解决途径：
 - 压缩数据表
 - 利用哈希表的快速查找特性对项集进行计数统计
 - 合理选样
 - FP-Growth算法



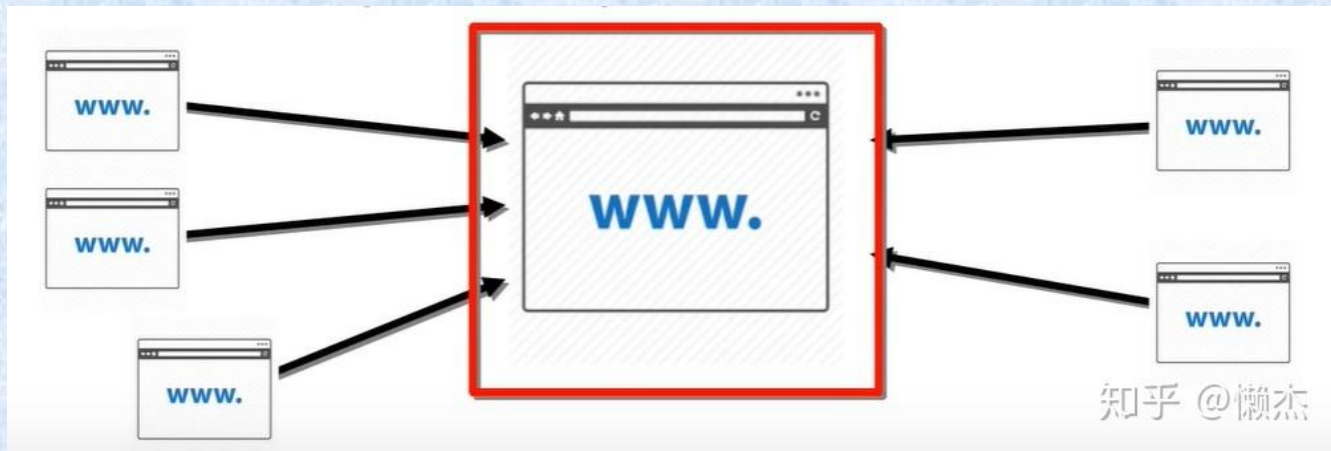
5.5 PageRank算法

- 数据关系还可以应用到搜索引擎和推荐系统
- PageRank的思想：看一个人怎样，看他有什么朋友就知道了
 - 被越多优质的网页所链接的网页，它是优质网页的概率就越大
- 网页PR值，概率上理解就是此网页被访问的概率，PR值越高其排名越高



5.5 PageRank算法

- 数量假设**：一个页面节点接收到的其他网页指向的入链数量越多，那么这个页面越重要。



- 质量假设**：越是质量高的页面指向页面A，则页面A越重要。





PageRank算法定义

出链：如果在网页A中包含了网页B的超链接，用户浏览网页A时可以点击该超链接然后进入网页B，这种情况即为A出链B。

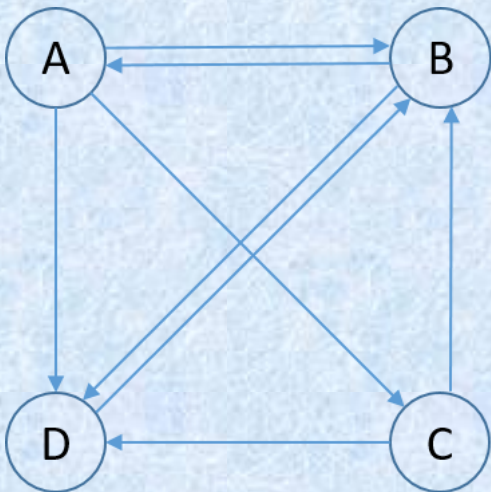
出度： $L(A)$ = total of 网页A包含的出链数目

PR值： $PR(A)$ 代表页A的PageRank值



PageRank算法步骤

- 给每个网页初始化PR值为 $\frac{1}{N}$ ，其中 N 为网页总数。
- 根据投票算法不断迭代， $PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$ ，直至达到平稳分布为止。
 - B_u 是所有链接到网页 u 的网页集合，网页 v 是 B_u 中的一个网页， $L(v)$ 是网页 v 的出度。



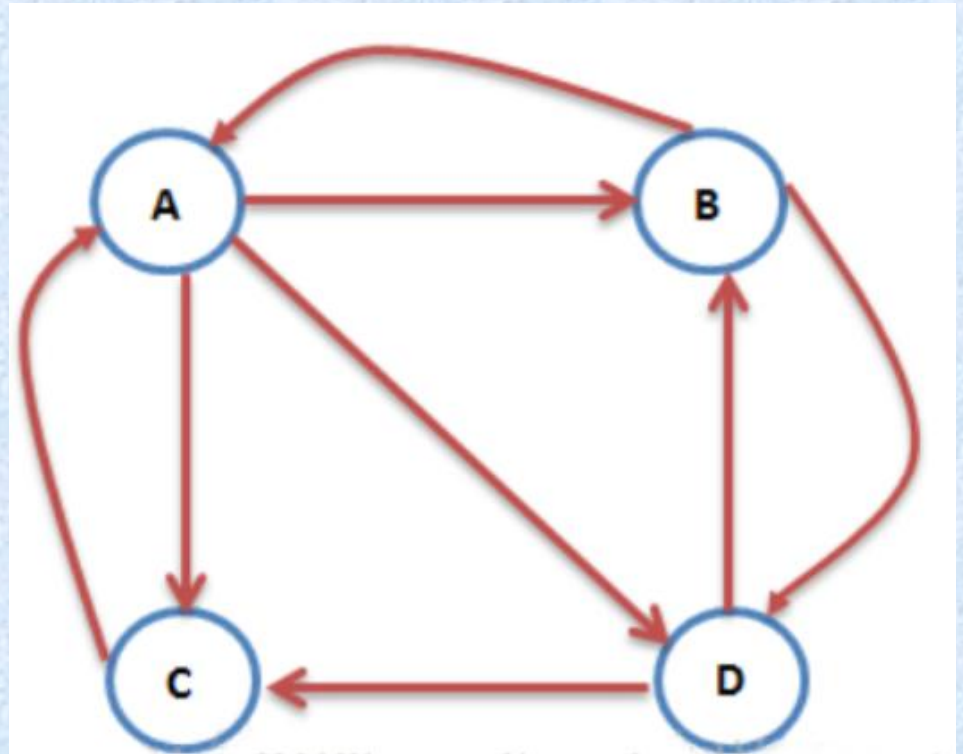
$$PR(D) = PR(A) / 3 + PR(B) / 2 + PR(C) / 2$$



PR值计算

case1: 网页都有出入链

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1}$$



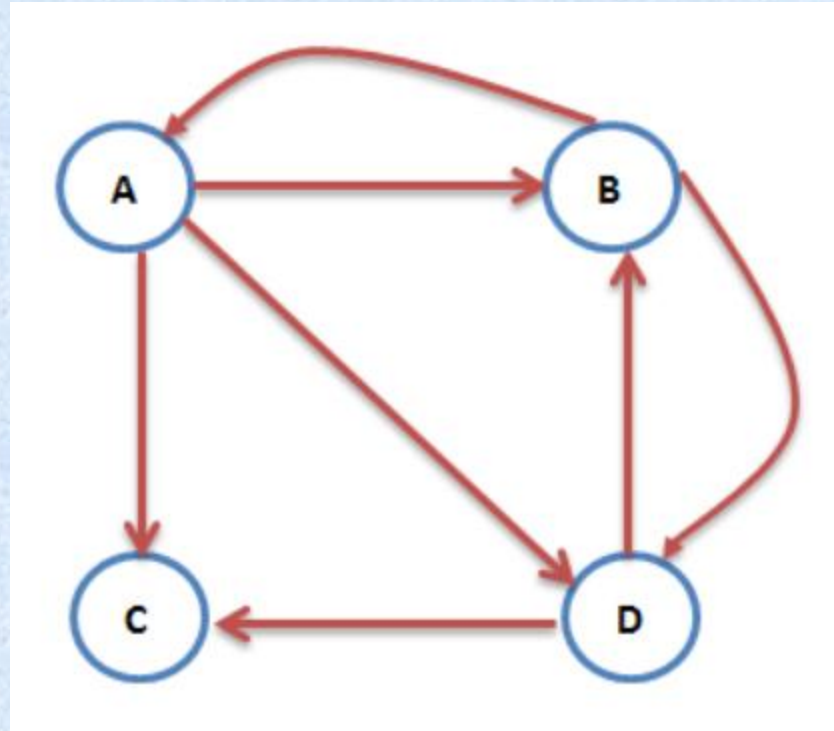


PR值计算

case2: 存在没有出链的网页

可以理解为：没有出链的网页，算法强制它对所有的网页都有出链，即让它对所有网页都有PR值贡献。

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{4}$$



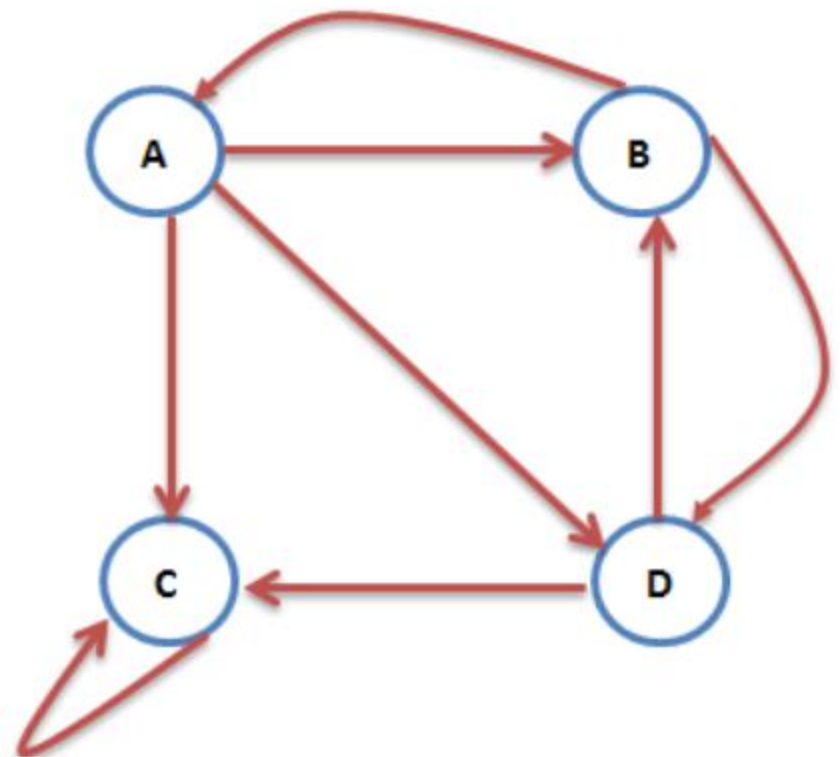


PR值计算

case3: 存在只对自己出链的网页

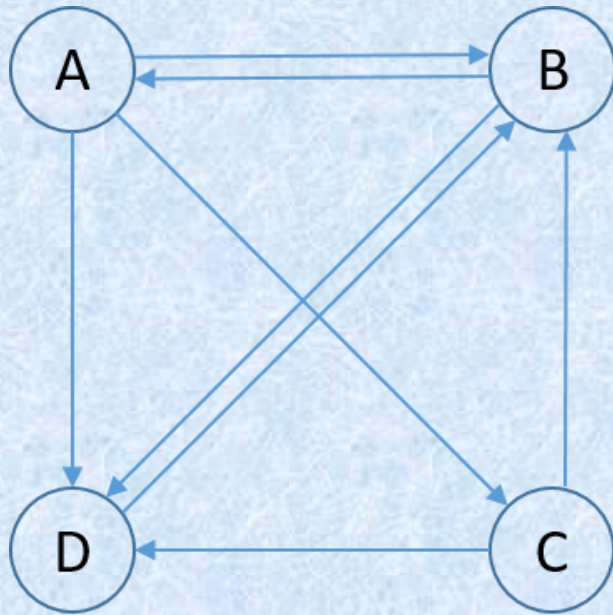
算法策略：设存在一定概率 α ，用户在地址栏输入A/B/C/D地址，然后从C跳转到A/B/C/D进行浏览。

$$PR(A) = \alpha \left(\frac{PR(B)}{2} \right) + \frac{(1-\alpha)}{4}, \text{ 一般取值 } \alpha=0.85$$





PageRank算法迭代



	$PR(A)$	$PR(B)$	$PR(C)$	$PR(D)$
初始值	0.25	0.25	0.25	0.25
第1次迭代	0.125	0.333	0.083	0.458
第2次迭代	0.1665	0.4997	0.0417	0.2912
.....
第n次迭代	0.1999	0.3999	0.0666	0.3333

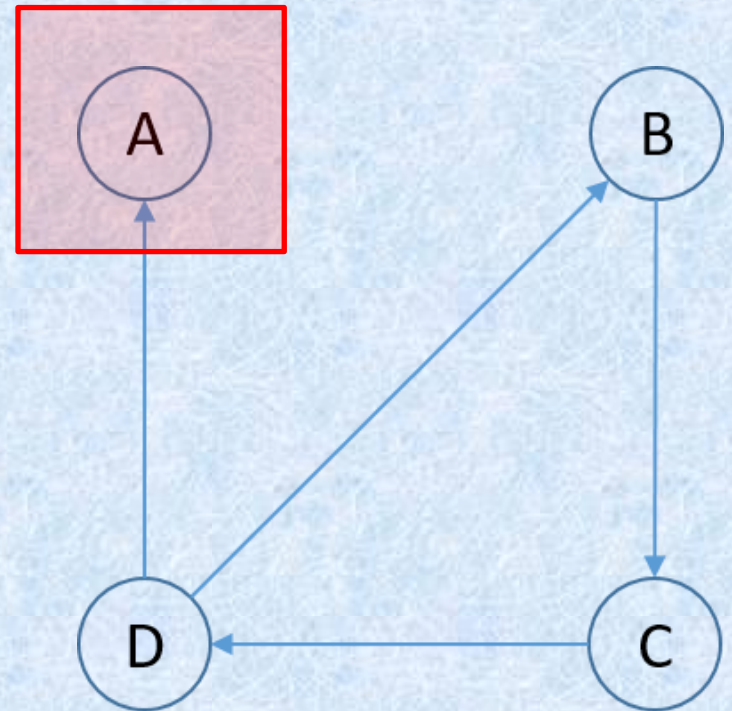


PageRank算法公式

- 网页没有出度—> **排名泄露**，所有网页的PR值都趋向于0

– 强制A对所有的网页包括自己都有出链

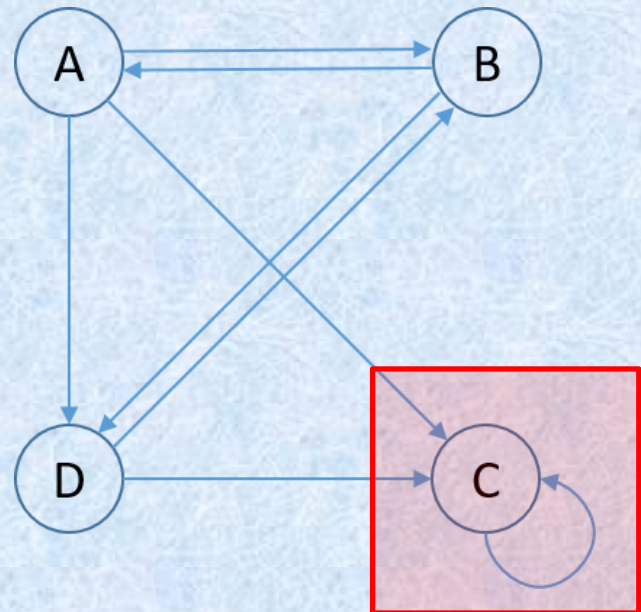
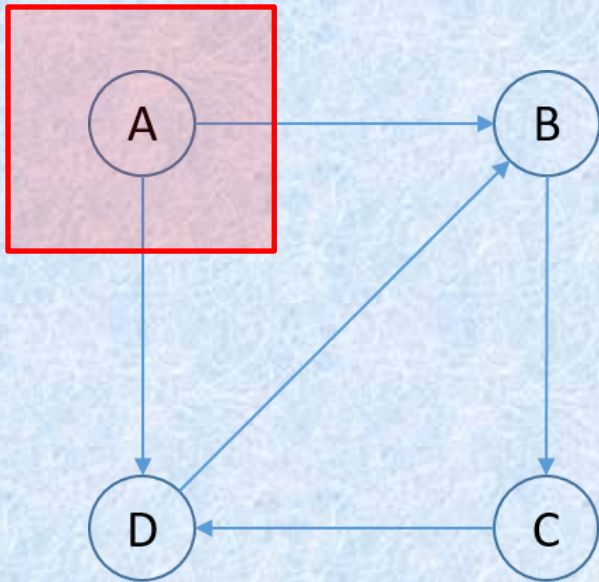
$$- PR(B) = \frac{P(A)}{4} + \frac{PR(D)}{2}$$





PageRank算法公式

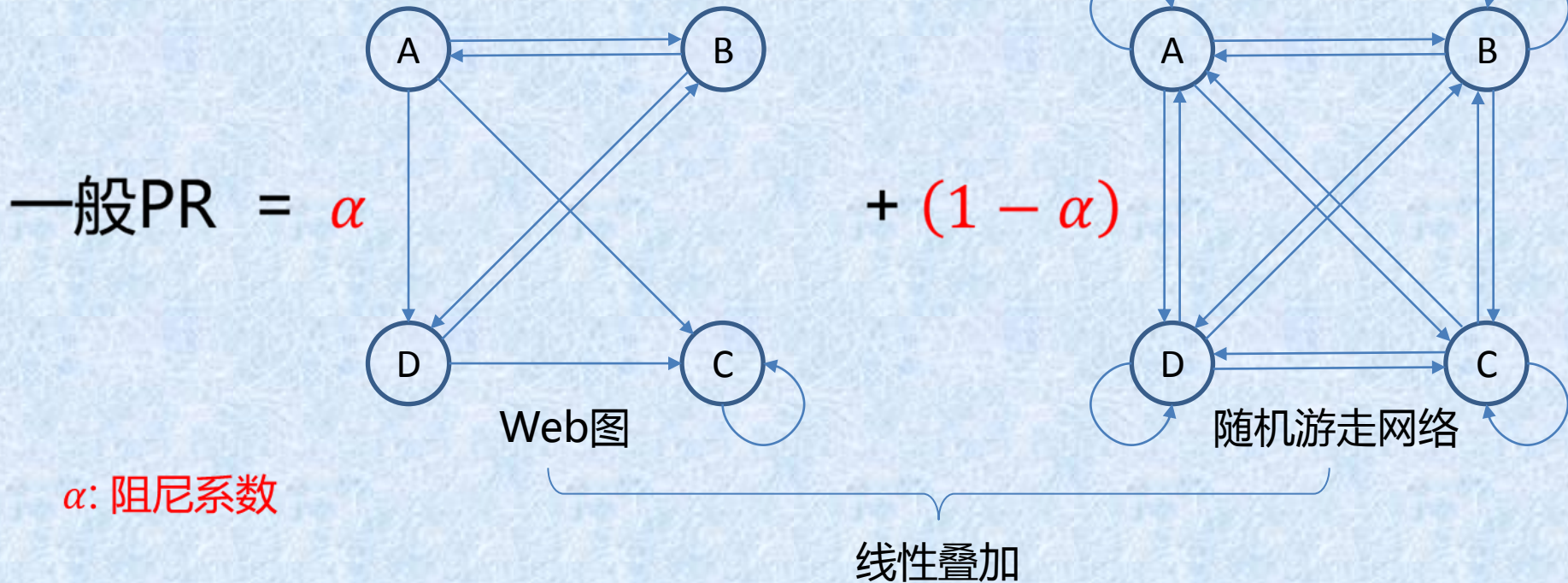
- 网页没有入度—> **排名下沉**，该网页PR值趋于0
- 网页只有对自己有出链，或者几个网页的出链形成封闭—> **排名上升**，这些网页的PR值只增不减





PageRank算法公式

- 网页没有入度—> **排名下沉**，该网页PR值趋于0
- 网页只有对自己有出链，或者几个网页的出链形成封闭—> **排名上升**，这些网页的PR值只增不减



$$PR(C) = \alpha \left(\frac{PR(A)}{3} + \frac{PR(D)}{2} \right) + (1 - \alpha) \frac{1}{4}$$

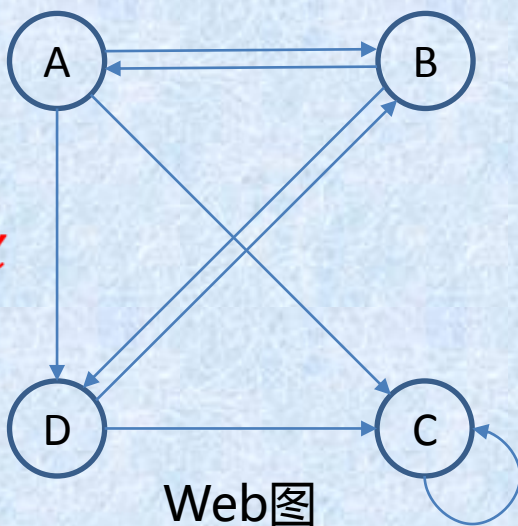


PageRank算法公式

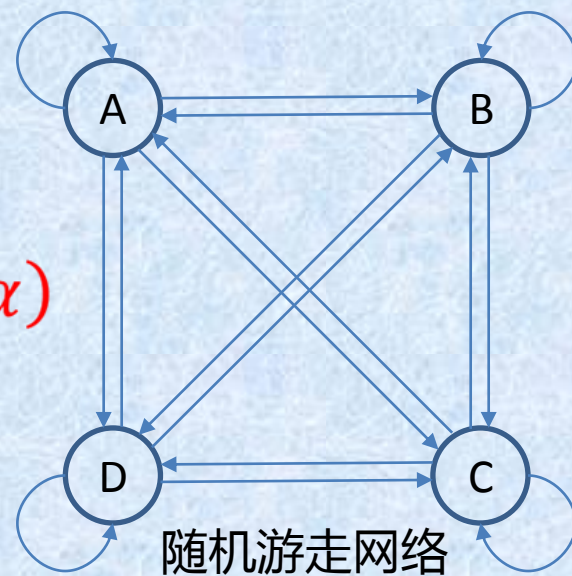
$$PR(u) = \alpha \sum_{v \in B_u} \frac{PR(v)}{L(v)} + \frac{(1 - \alpha)}{N}$$

一般PR = α

α : 阻尼系数



+ $(1 - \alpha)$



线性叠加



PageRank优缺点

- 优点：与查询无关的静态算法，所有网页的PageRank值通过**离线计算**获得；有效减少在线查询时的计算量，极大降低了查询响应时间。
- 缺点：**过分相信链接关系**，而一些权威网页往往是相互不链接的；忽视了**主题相关性**；**旧**的页面等级会比新页面高。
- Google现在使用的，不是简单的PageRank。



PageRank优缺点

可以通过“僵尸网站”或链接，人为刷PageRank值

