第2章 万维网网页信息的表达及解析

2.1 万维网架构及网页表达

2.2 网页信息抽取

2.3 搜索引擎与分析系统

2.4 搜索引擎索引系统

2.5 搜索引擎查询系统5

《走进搜索引擎》 潘雪峰 花贵春 梁斌编著
电子工业出版社 2011年5月第2版

第2章 万维网网页信息的表达及解析

2.3 搜索引擎与分析系统

2.3.1 搜索引擎概述1

2.3.2 搜索引擎分析系统3.6

2.3.3 中文分词3.4

2.3.4 PageRank3.5

2.3.1 搜索引擎概述

- 搜索引擎：
  - 一种用来在计算机网络，特别是在万维网上检索各种文件的计算机程序。
  - 一个用户定义的信息聚合系统。通过用户输入的查询关键词，推测用户的查询意图，然后快速地返回相关的查询结果，供用户选择。
- 公认的搜索引擎有如下3
  - 目录式搜索引擎
  - 全文搜索引擎
  - 元搜索引擎（Meta Search Engine）

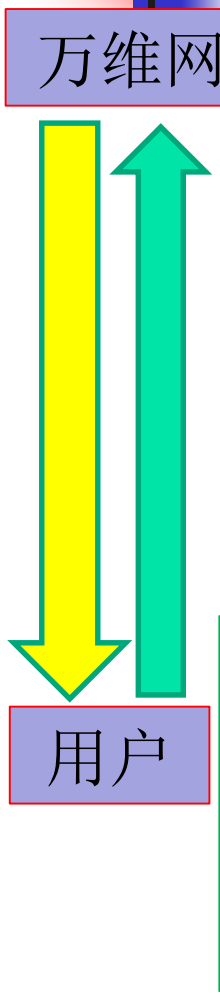
- 人工甄别、分类（Yahoo，Sohu）
- 检索的目标结果是网站
- 数据量有限、更新不及时维护成本较高

- 针对万维网所有网页的搜索引擎(谷歌、百度)
- 面向网页的全文检索服务
- 信息量大、更新及时，无须人工干预；
- 返回信息过多，用户必须自己筛选

- 又称多搜索引擎，通过一个统一的用户界面帮助用户在多个搜索引擎中选择和利用合适的（甚至是同时利用若干个）搜索引擎来实现检索操作，是对分布于网络的多种检索工具的全局控制机制
- 用户需要做更多的筛选

2.3.1 搜索引擎概述

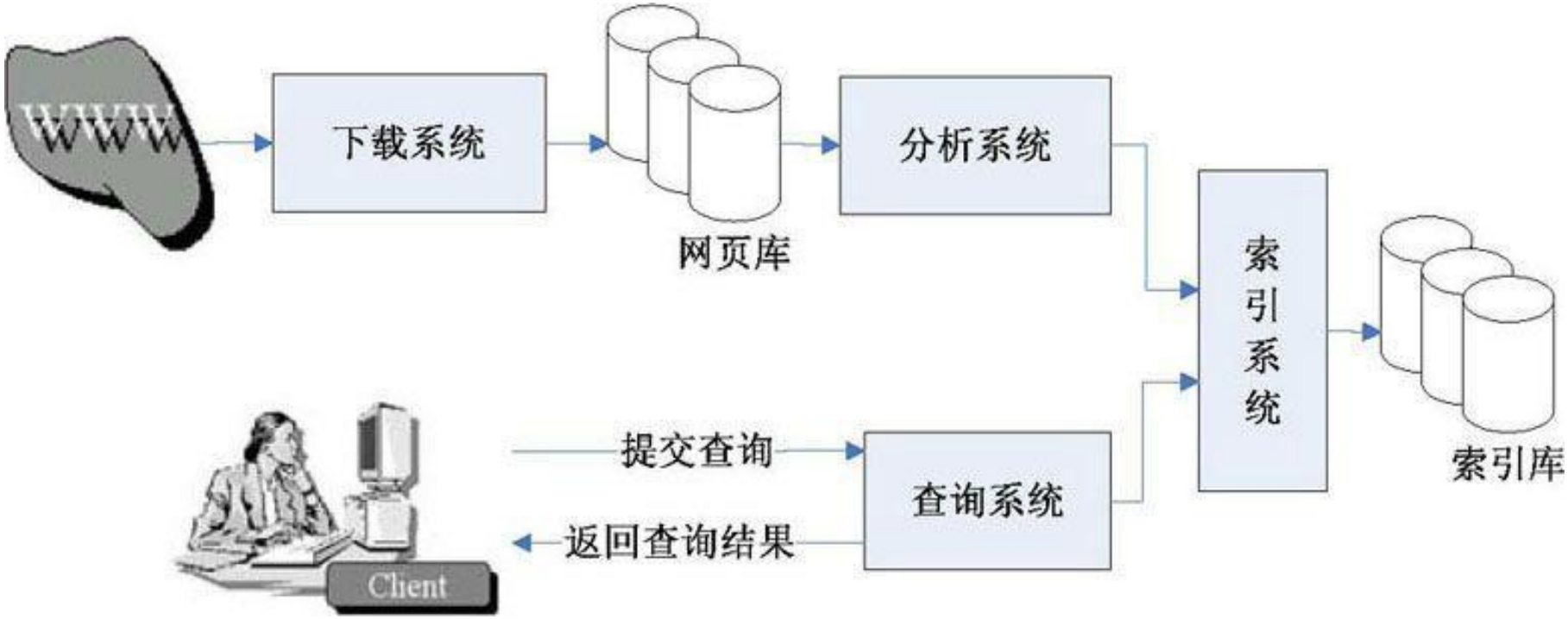
- （全文）搜索引擎的体系结构:



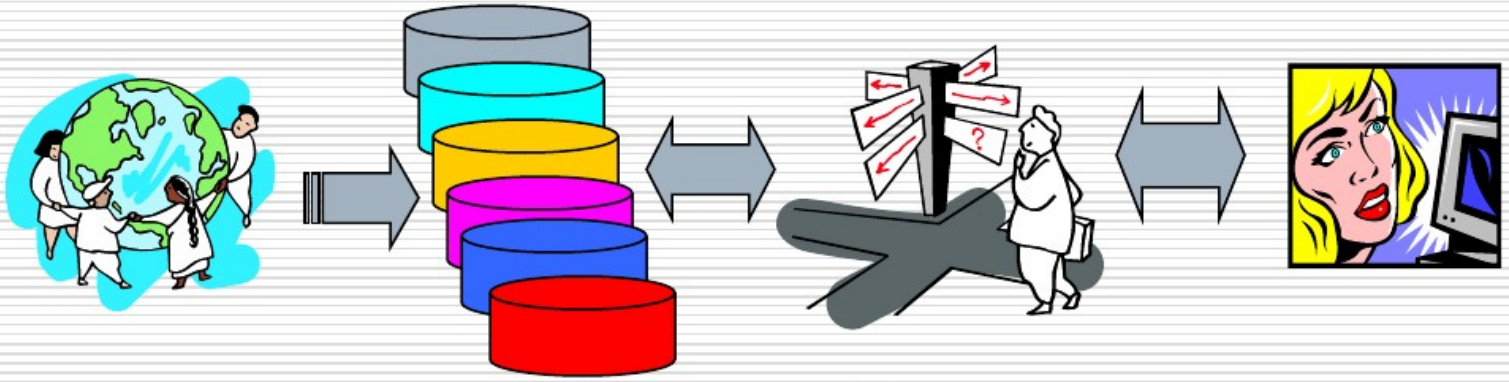
- 下载系统：
  - 网页下载，负责从万维网上自动下载各种类型的网页，并且保持对万维网变化的同步
- 分析系统：
  - 信息抽取,负责抽取下载系统得到的网页数据，并进行PageRank(网页排序)和分词计算，发现信息
- 索引系统：
  - 负责将分析系统处理后的网页对象建立索引，入库
- 查询系统：
  - 负责分析用户提交的查询请求，然后从索引库中检索出相关网页并将网页排序后，以查询结果的形式返回给用户

2.3.1 搜索引擎概述

- 搜索引擎的体系结构图:



搜索引擎的工作原理



搜索器

在因特网中发现和搜集信息

索引器

理解搜索器所搜索的信息，从中抽取出索引项，用于表示文档以及生成文档库的索引表，建立起自己的物理索引数据库。

检索器

根据用户的查询在索引库中快速检出文档，进行文档与查询的相关度评价，对将要输出的结果进行排序，并实现某种用户相关性反馈机制。

用户界面

输入用户查询、显示查询结果，提供用户相关性反馈机制。

下载系统
分析系统

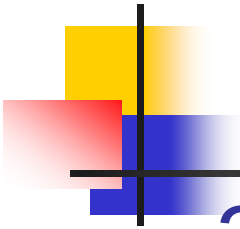
索引系统

查询系统

有代表性的中英文搜索引擎

- Google
- Lycos
- Infoseek
- Excite
- Ask Jeeves
- Inktomi
- Northern Light
- Wisenut
- AOL
- Alltheweb

- 百度
- 天网搜索
- 中国搜搜
- 爱问
- 搜狗



第2章 万维网网页信息的表达及解析

2.3 搜索引擎与分析系统

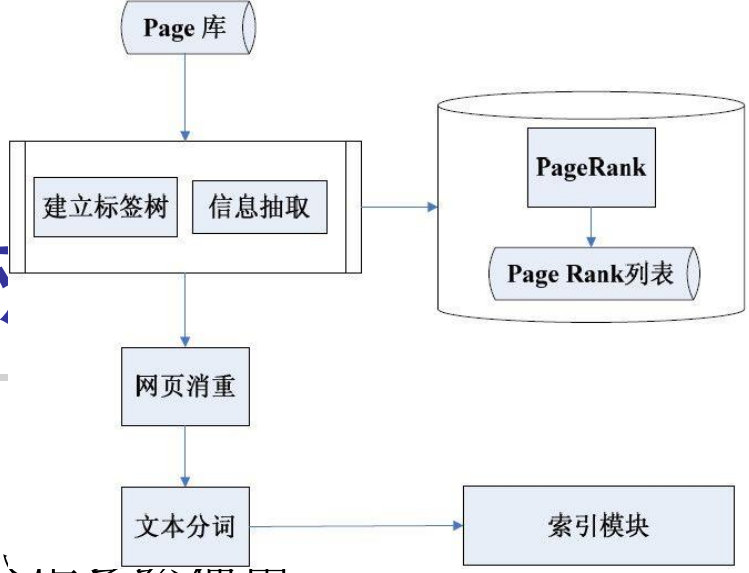
2.3.1 搜索引擎概述1

2.3.2 搜索引擎分析系统3.6

2.3.3 中文分词3.4

2.3.4 PageRank3.5

2.3.2 搜索引擎分析



- 分析系统功能：
  - Page库
    - 下载系统下载到的原始网页，供分析系统使用。
  - 网页结构化：
    - 建立标签树，并从网页中抽取有价值的属性→网页对象
  - 网页消重：
    - 丢弃冗余页面，相似或相同的网页仅保留一个传给分词模块。
  - 文本分词：
    - 将正文切分成以词汇为单位的集合。
  - PageRank：
    - 网页排序确定网页价值
  - 索引模块：
    - 将分析的结果发往索引模块，进行索引入库。

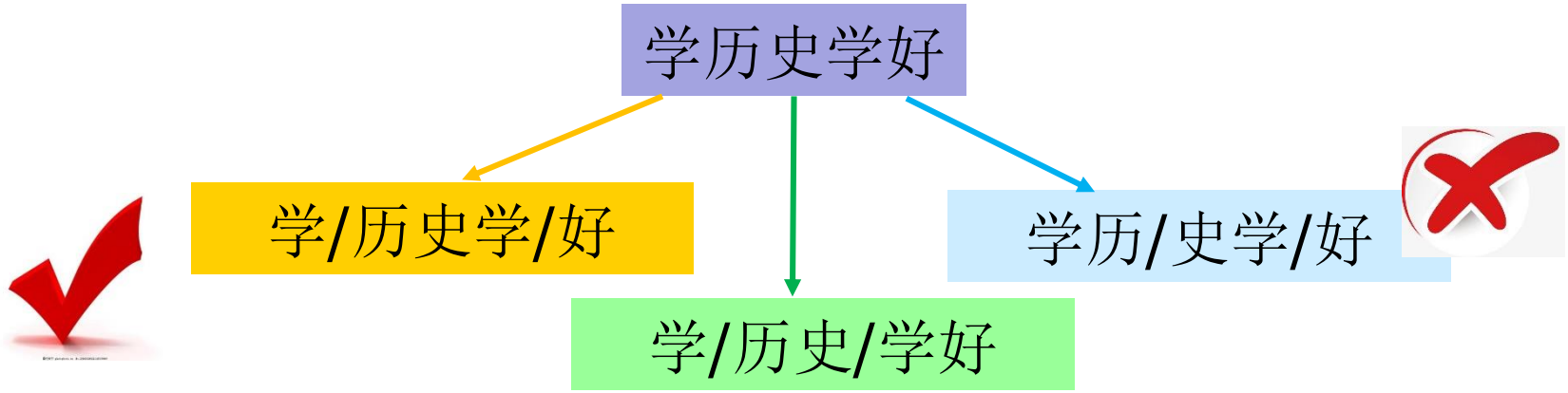
第2章 万维网网页信息的表达及解析

2.3 搜索引擎与分析系统

- 2.3.1 搜索引擎概述1
- 2.3.2 搜索引擎分析系统3.6
- 2.3.3 中文分词3.4
- 2.3.4 PageRank3.5

2.3.3 中文分词

- 任何文档都可以看做是一些连续的词的集合，然而中文并没有明显的词间分隔（比英文复杂）
- 在中文语法中，词汇是由两个或两个以上汉字组成的，并且句子是连续书写的，句子间由标点分隔。
- 中文分词
  - 将整句中文切割成小的词汇单元



2.3.3 中文分词

- 目前分词的主要手段：
  - 字典分词法
    - 通过字典实现分词，也叫“机械分词法”，将分词的句子与词典中的词语进行匹配，匹配成功作为一个词，生成序列；
  - 统计学分词法
    - 把每个词看做是由字组成的
    - 统计词在整个语料库中的出现频率
    - 如果相连的字在不同文本中出现的次数越多，就证明这段相连的字很有可能就是一个词
- 没有一种分词方法能够解决全部的问题

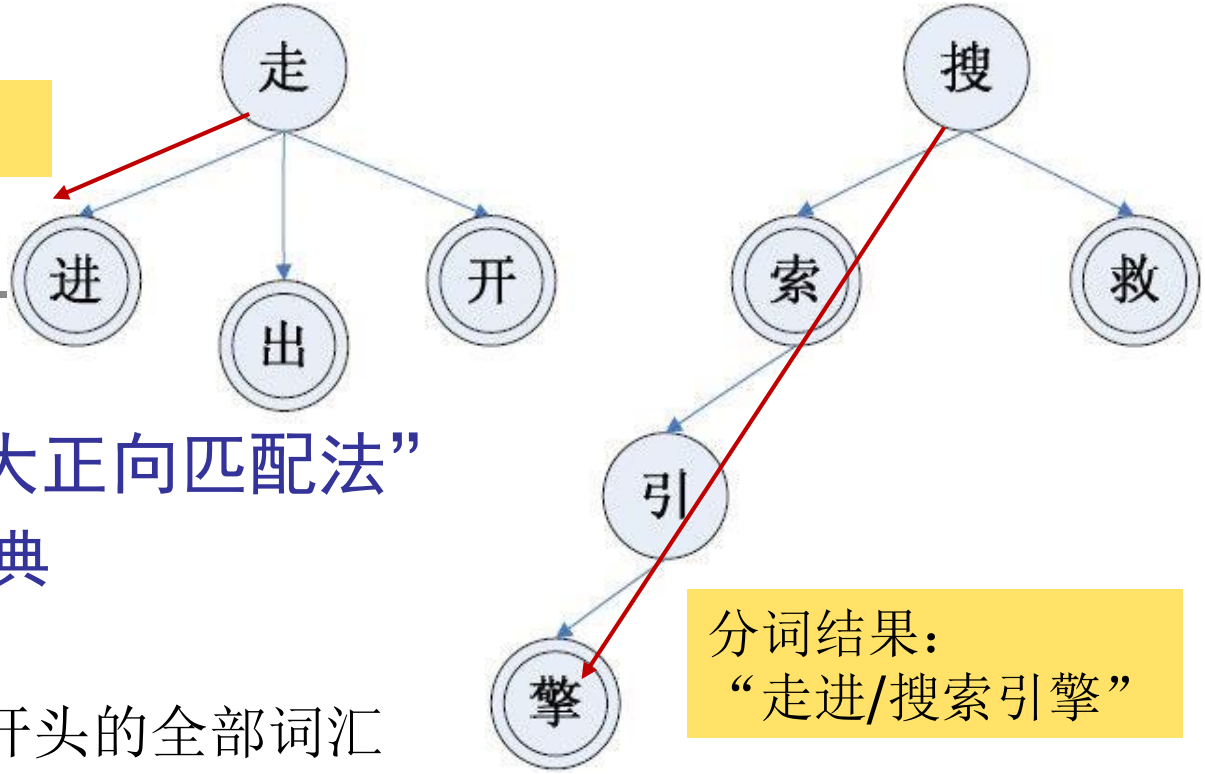
2.3.3 中文分词

1.字典分词法，常用方法：

- **MM法“最大正向匹配法”**(Maximum Matching Method)
  - 前缀树结构的词典
- **RMM法“逆向最大匹配法”**(Reverse Maximum Matching Method)
  - 后缀树结构的词典
- 最少切分法：使每一句中切出的词数最小，这也是基于词典分词的方法

- MM和RMM两种方法基本原理相同，词典构成与分词的扫描方向不同

“走进搜索引擎”分词：



分词结果：
“走进/搜索引擎”

- 1）MM法“最大正向匹配法”
- 前缀树结构的词典
  - 以一个字开头
  - 其后是以该字开头的全部词汇
  - 双线圈的词，认为是一次终结，表示它及其全部祖先可以构成字典中的一个词汇
- 最大正向匹配
  - 从词典根部向下读取，搜索最长匹配

有些解决不了，如“学历史学好” 分成：
“学历 / 史学 / 好”——错误结果

2）逆向最大匹配法

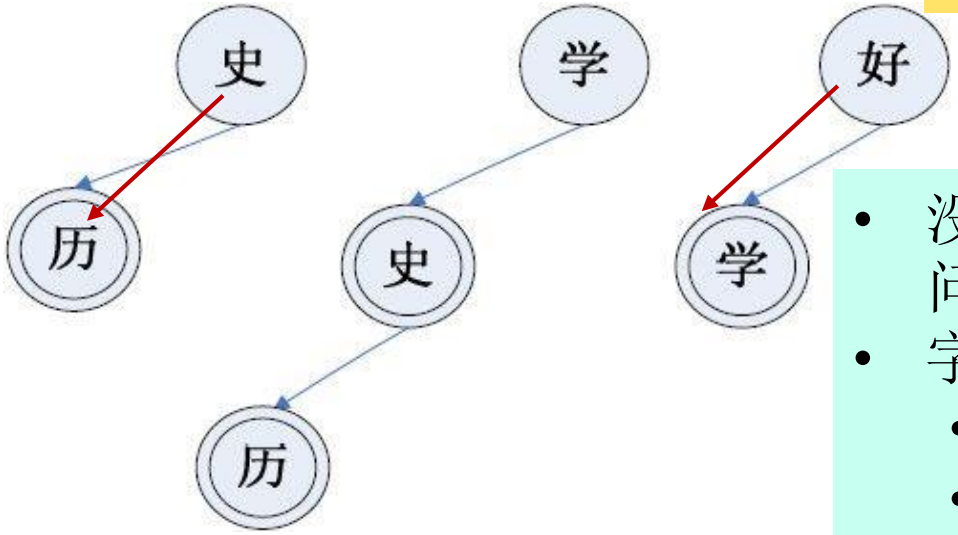
- 后缀树结构的词典
  - 树根表示一个词的结尾字，双圈字表示某个词的词头
- 逆向最大匹配法

例：“学历史学好”分词

从句子的右边向左边读，首先找到“好”这个分支。然后读入“学”，恰好存在这样一个词，因此抽取出它。

继续读入“史”，依次读入全部句子，直到切分结束

分词结果：“学 / 历史 / 学好”



- 没有一种分词方法能够解决全部的问题
- 字典分词法不足
  - 字典收录词数的限制
  - 词典发展滞后语言(新词)

2.3.3 中文

假如一个句子S有几种分词方法：

A1,A2,A3,…,Ak

B1,B2,B3,…,Bm

C1,C2,C3,…,Cn

如果：

P(A1,A2,A3,…,Ak)>P(B1,B2,B3,…,Bm)

P(A1,A2,A3,…,Ak)> P(C1,C2,C3,…,Cn)

那么A1,A2,A3,…,Ak就是最好的分词方法

2.统计分词法

- 基本原理
  - 根据字符串在语料库中出现的统计频率来决定其是否构成词
  - 词是字的组合，相邻的字同时出现的次数越多，就越有可能构成一个词
  - 因此字与字相邻共现的频率或概率能够较好的反映它们成为词的可信度。
- 统计分词一般做如下两步操作：
  - 1）建立统计语言模型（n-gram），对句子进行单词划分
  - 2）对划分结果做概率计算，获取概率最大的分词方式（评估语句是否合理）

2.3.3 中文分词

- 1）n-gram模型：https://blog.csdn.net/sdu_hao/article/details/86893580
  - 自然语言处理中常见的一种基于统计的语言模型
  - 基本思想：
    - 将文本里面的内容按照字进行大小为N的滑动窗口操作，形成长度为N的片段序列。
    - 每一个片段称为gram，在所给语句中对所有的gram出现的频数进行统计。
    - 再根据整体语料库中每个gram出现的频数进行比对可以得到所给语句中每个gram出现的概率。
    - 整句的概率就是各个词出现概率的乘积。
- 在判断句子合理性、句子相似度、分词等方面有突出的表现

2.3.3 中文分词

- 2）对划分结果做概率计算，获取概率最大的分词方式

s_1 = 我刚吃过晚饭

s_2 = 刚我过晚饭吃

对于中文来说，$P(s_1) > P(s_2)$

- the large green__. ('mountain' or 'tree'?)
- Kate swallowed the large green__.('pill' or 'broccoli'?)

前面的(历史)信息越多，对后面未知信息的约束就越强

先刷__　　早起先刷__

加了“早起”，好填得多

P(牙丨早起先刷)>P(牙丨先刷)

- 概率可以表示句子合理性
- 我们掌握的上下文越多，对句子的理解就会越准确

如果我们有一个由n个词组成的序列/句子s，即s = $w_1, w_2, \ldots, w_n$
该句子出现的概率：

$$P(S)=P(w_1w_2w_3\ldots w_n)=P(w_1)*P(w_2|w_1)*P(w_3|w_1w_2)*\ldots*P(w_t|w_1w_2w_3\ldots w_{t-1})$$

引入马尔科夫假设：当前词只与最多前n-1个有限的词相关

- N-gram算法：

unigram: $P(S) = P(w_1) * P(w_2) * \ldots * P(w_t) = \prod_{i=1}^{t} P(w_i)$

bigram: $P(S) = P(w_1) * P(w_2|w_1) * \ldots * P(w_t|w_{t-1}) = P(w_1) * \prod_{i=2}^{t} P(w_i|w_{i-1})$

trigram:
$$P(S) = P(w_1) * P(w_2|w_1) * P(w_3|w_2w_1) \ldots * P(w_t|w_{t-1}w_{t-2})$$
$$= P(w_1) * P(w_2|w_1) \prod_{i=3}^{t} P(w_i|w_{i-1}w_{i-2})$$

- 训练语料库例https://blog.csdn.net/sdu_hao/article/details/86893580
  - Bigram统计模型分词
  - 假设语料库总词数为13,748

I want to eat Chinese food lunch.

各词出现的次数

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

n=2词序列频度

I-I频次：5

I-want频次：827

I-food频次：0

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0			

问题：数据稀疏

2.3.3 中文分词

Chinese？

Chinese？

- 计算“s1=i want english food”和“s2=want i english food”哪个句子更合理
  - 从表1中知道 “i” 一共出现了2533次，而其后出现 “want” 的情况一共有827次，所以P(want|i)=827/2533≈0.33
  - 同理：p(eat|i)=0.0036这个概率值讲解，从表一得出“i”一共出现了2533次，而其后出现eat的次数一共有9次，p(eat|i)=p(eat,i)/p(i)=count(i,eat)/count(i)=9/2533 = 0.0036

P(s1)=P(i|<s>)P(want|i)P(english|want)P(food|english)P(</s>|food)

=0.25×0.33×0.0011×0.5×0.68=0.000031

s1=“i want english food</s>”更像人话

P(s2)=P(want|<s>)P(i|want)P(english|want)P(food|english)P(</s>|food)

=0.25*0.0022*0.0011*0.5*0.68 = 0.00000002057

2.3.3 中文分词

- 如何选择依赖词的个数n呢？从前人的经验来看：
- 1.经验上，trigram用的最多。尽管如此，原则上，能用bigram解决，绝不使用trigram。n取≥4的情况较少。
- 2.当 n 更大时：对下一个词出现的约束信息更多，具有更大的辨别力；
- 3. 当 n 更小时：在训练语料库中出现的次数更多，具有更可靠的统计信息，具有更高的可靠性、实用性。

- 常用1-gram(unigram)、2-gram(bigram)、3-gram(trigram)

扩展-N-Gram语言模型

https://blog.csdn.net/ahmanz/article/details/51273500

- n-gram的数据格式（google books Ngram Viewer）：

```
1  circumvallate   1978   335    91
2  circumvallate   1979   261    91
```

一个1-gram的数据片段，“circumvallate”这个单词在1978年出现335次，存在91本书中

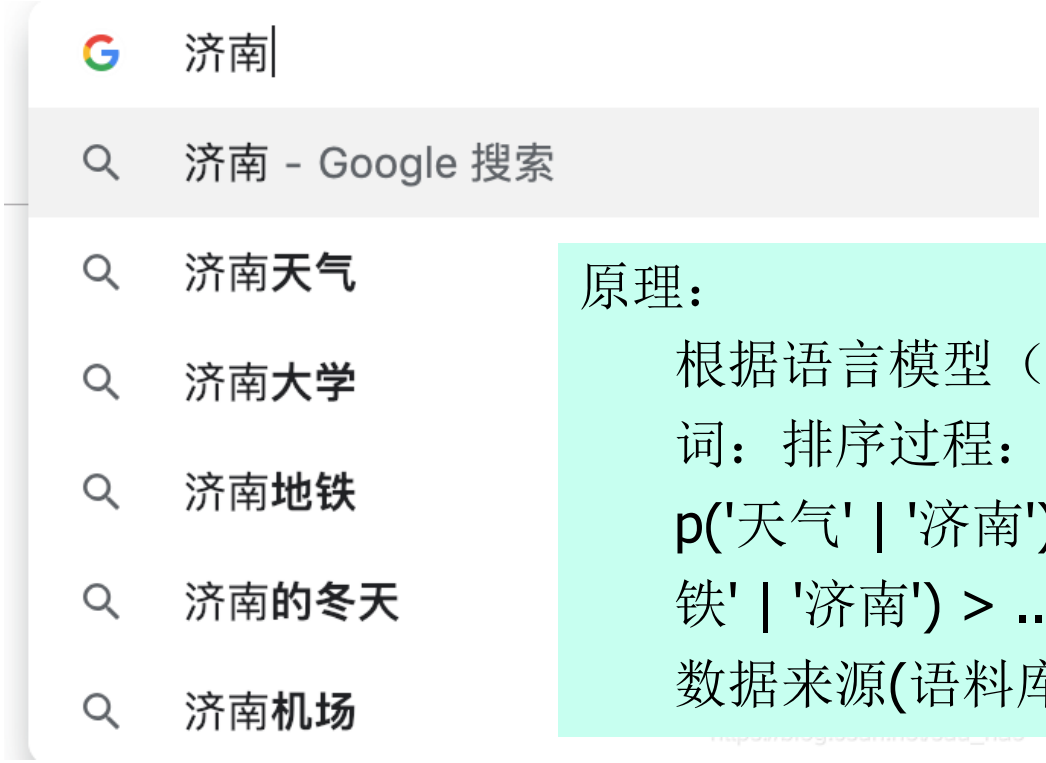
```
1  analysis is often described as  1991  1   1   1
```

5-gram数据片段

n-gram只是一个语言模型，只要把需要的信息存储下来，至于什么格式都是依据应用来定。
如：1-gram的数据片段中，除了频率335次是必须的，其他的元数据（例如，还有词性等）可以根据应用需求来定

扩展-N-Gram语言模型

- N-Gram语言模型的其他应用 https://blog.csdn.net/sdu_hao/article/details/86893580
- 搜索引擎(Google或百度)或输入法的猜想或提示



原理：

根据语言模型（2-gram语言）预测下一个单词：排序过程：

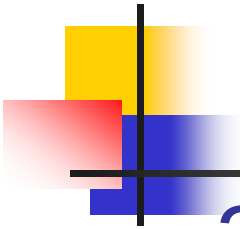
p('天气' | '济南') > p('大学' | '济南') > p('地铁' | '济南') > ...,

数据来源(语料库)可以是用户搜索的log。

扩展-N-Gram语言模型

https://blog.csdn.net/ahmanz/article/details/51273500

- n-gram应用——拼音输入法
- 可能的输出：



第2章 万维网网页信息的表达及解析

2.3 搜索引擎与分析系统

2.3.1 搜索引擎概述1

2.3.2 搜索引擎分析系统3.6

2.3.3 中文分词3.4

2.3.4 PageRank3.5

2.3.4 PageRank

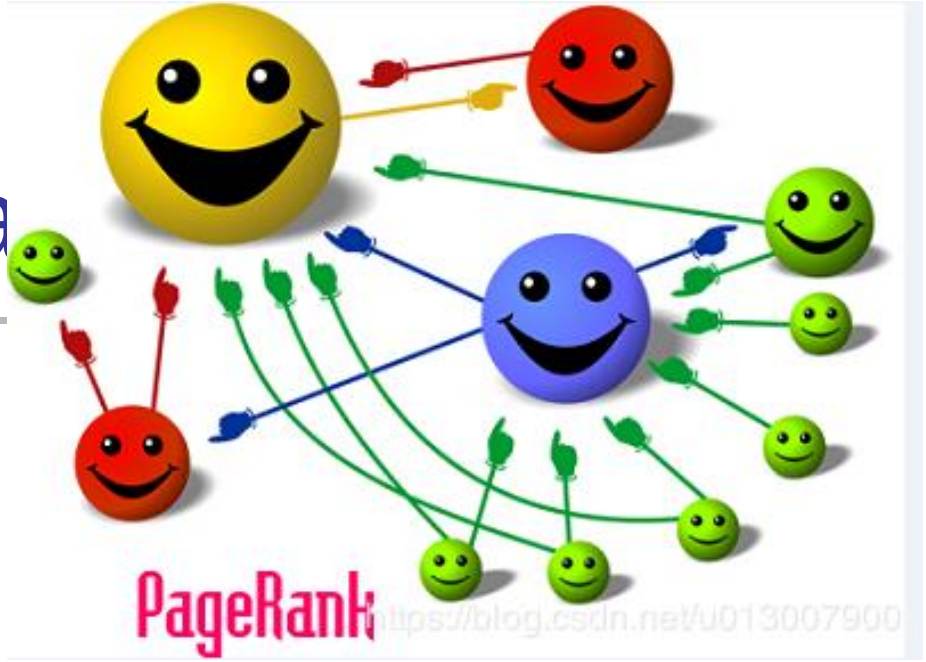
PageRank的核心思想就诞生了——网页排名

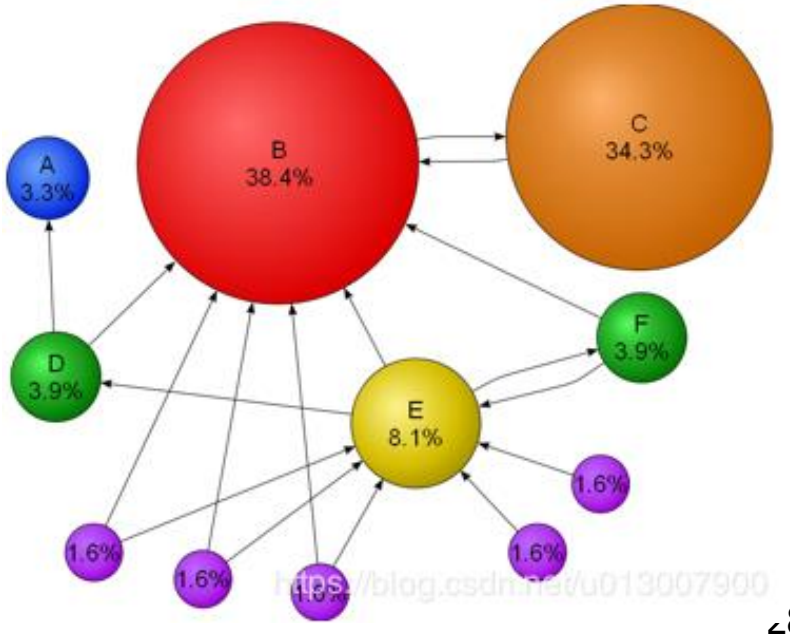
算法来源

- 网页搜索的本质是网页信息的聚合
- 网页被聚合后就自然会产生排序问题，需要通过科学有效的方法将“好”的搜索结果按照“好”的程度依次排列。
- 如何评价这个“好”成为搜索质量的关键。
- Google发明了PageRank：
  - 谷歌的两位创始人，当时还是美国斯坦福大学(Stanford University) 研究生的佩奇 (Larry Page) 和布林 (Sergey Brin) 开始了对网页排序问题的研究——用网页的重要性来评价

2.3.4 PageRa

- PageRank的核心思想
  - 数量假设：一个页面越被其他页面链接，说明他越重要（ps：难怪好多技术博客的都互相链接）
  - 质量假设：越是被高质量页面链接，说明该页面越重要（ps：最好能被大博主推荐一波，粉丝蹭蹭蹭往上涨）





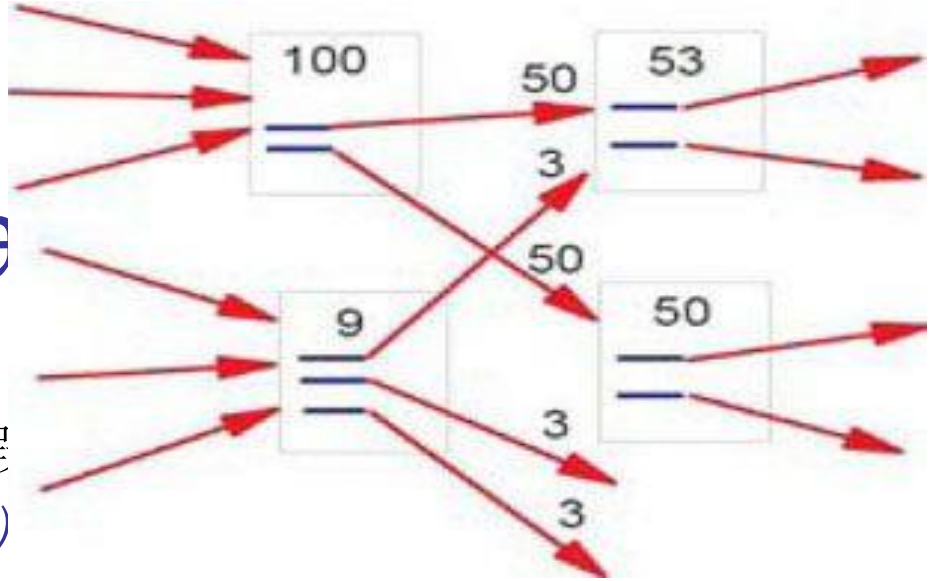
2.3.4 PageRank

- PageRank也称为“链接分析”（link analyze）
  - 如果网页A链接到B
    - 网页A的编写者对网页B的一种认可
    - 网页A为网页B投了一票，网页B的重要性被网页A认可。
  - 具体：
    - （1）反向链接（backlink）越多的网页越重要。
    - （2）反向链接的源网页质量越高，被链接指向的网页越重要。
    - （3）（正向）链接数越少的网页越重要。

2.3.4 PageRank

- 例.假定在一次比赛中一个网球选手A被另一个网球选手B击败，定义为A→B（表示A输给B，或者说A认可了B的厉害）
- 相应结论：
  - 如果B的反向链接越多（认可B厉害的人越多），B的排名越高
  - 如果输给B的选手的排名越高（认可B厉害的人也是厉害的人），那么B的排名越高。
  - 如果B输给的选手越少（B认可的厉害的人越少），那么B的排名越高。
- 综上所述，赢的次数多、赢得对手质量高且输的少的选手的排名高是很自然的，PageRank的算法的基本思想也来自于此。

2.3.4 PageRa



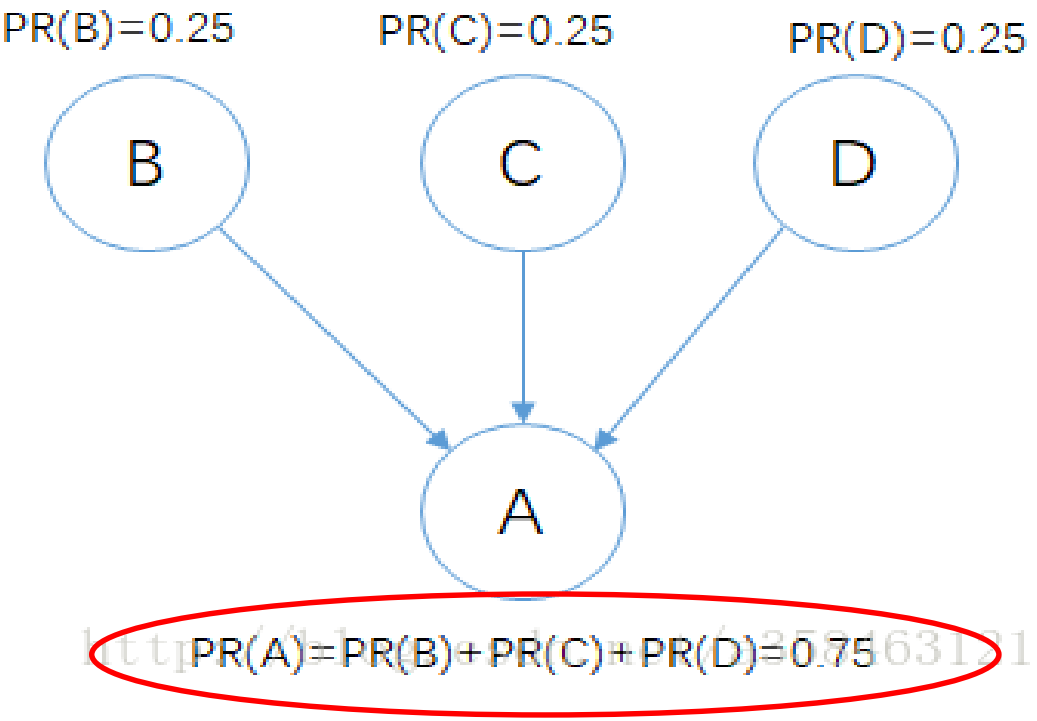
- PageRank算法计算：

  （充分利用了两个假设：数量假

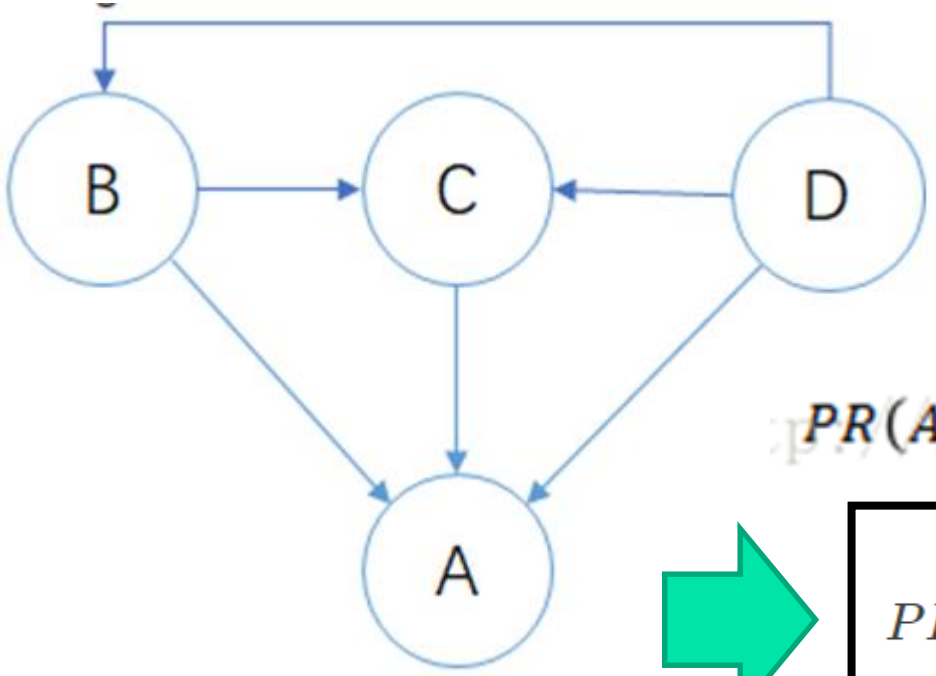
- 页面A的PageRank值*PR(A)*
  - 1）初值：
    - 每个页面设置相同的PageRank值
  - 2）更新页面PageRank值：
    - 将每个页面当前的PageRank值平均分配到本页面包含的出链上，这样每个链接即获得了相应的权值。
    - 每个页面将所有指向本页面的入链所传入的权值求和，即可得到新的PageRank得分
    - 当每个页面都获得了更新后的PageRank值，就完成了一轮PageRank计算
  - 3）不断迭代

- PageRank算法核心思想：
  https://blog.csdn.net/a358463121/article/details/79342109
  - 计算一个用户随机点击一个网站，然后不停点击从而到达各个网站的概率。
  - 一个网站的打开概率又取决于那些指向他自己的那些网站的概率，所以这个概率的计算是一个不断迭代的过程

一个简单的例子：



- PageRank算法核心思想（续）：
  - 如果网站D有3个外链，那么你从网站D跳到网站B的概率就不一定是100%了，要给它做一个权重衰减——PR值除以3



$$PR(B) = \frac{PR(D)}{3}$$

$$PR(C) = \frac{PR(B)}{2} + \frac{PR(D)}{3}$$

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1} + \frac{PR(D)}{3}$$

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

- PR值计算公式

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

- 其中:
  - L:表示结点的出度
  - $v \in B_u$:所有指向u的结点

- PR值的衰减:
  - 一个用户在点击网页的时候是不会无限点下去，他最终会在某个结点上停止
  - 于是，引入一个damping factor(阻尼系数d）来表达这种关系
  - 当你计算PR的时候，要乘一个衰减的系数d来认为有一定概率会在上一个页面停止，而不会跳转到这个页面来

- PR的公式改写成如下:

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

- d一般取0.85,N是结点数量
- (1-d)/N是为了保证这个概率值在0到1之间
- $M(p_i)$: 所有指向p_i的网页

2.3.4 PageRank

- PageRank计算公式（另一种表示）：

$$PR(A) = (1-d) + d(PR(T_1)/C(T_1) + \ldots + PR(T_n)/C(Tn))$$

- $PR(A)$ 是页面A的PR值
- $PR(T_i)$是页面T_i的PR值，页面T_i是**指向A**的所有页面中的某个页面
- $C(T_i)$是页面T_i的出度，也就是T_i指向其他页面的边的个数
- d 为阻尼系数，其意义是，在任意时刻，用户到达某页面后并继续向后浏览的概率
  - 该数值是根据上网者使用浏览器书签的平均频率估算而得，通常d=0.85

2.3.4 PageRank

- PageRank采用以下公式计算：［Sergey Brin 1998］

$$PR_n(A) = (1-d) + d \times \left(\sum_{i=1}^{m} \frac{PR_{n-1}(T_i)}{C(T_i)}\right)$$

（1）$PR_n(A)$：网页A的PageRank值。

（2）$PR_{n-1}(T_i)$：网页T_i存在指向A的链接，并且网页T_i在上一次迭代时的PageRank值。

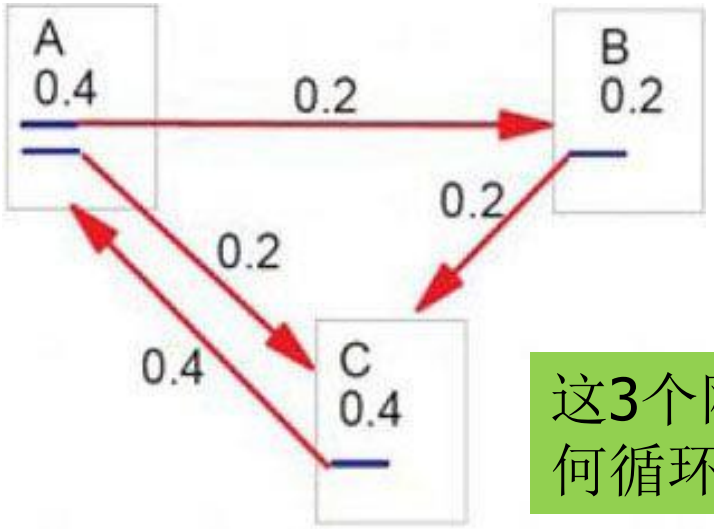
（3）C（T_i）：网页T_i的外链数量。

（4）d：0<d<1，阻尼系数，表示在随机冲浪模型中网页T_i将自身d的份额的PageRank值平均分给每个外链。

（5）m是所有指向A的结点

2.3.4 PageRank

- PR值的收敛：
  - 因为万维网彼此相连，如果分数被这样不断地传递下去，这样的计算会没完没了吗？



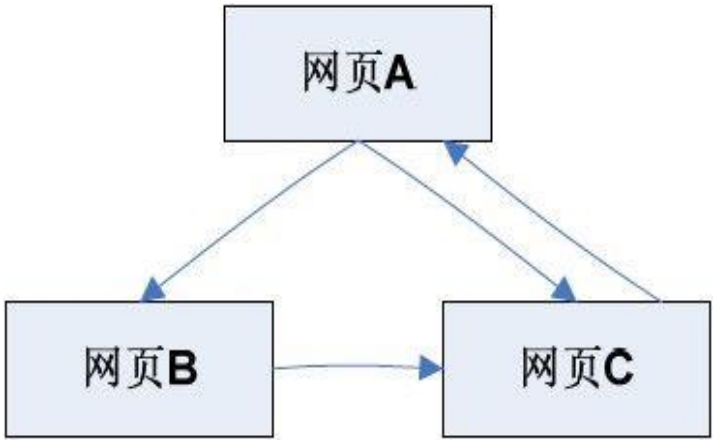
- 无论怎样迭代的计算，每个网页的得分都是固定不变的。
- 网页A的分数平均分给了网页B和网页C
- 网页B又完全给了C
- 最后网页C把得到的分数又还给了网页A。

这3个网页得到的分数之和总是1，而且无论如何循环计算，这种分数分布不再发生变化

这种特性在数学上称为“不变分布”——PageRank计算收敛的最终奥秘

2.3.4 PageRank

- 例.假定存在如图所示的简单的网页链接关系
- 由PageRank的计算方法得到下列方程组，假定d＝0.5：
  - PR(A)＝0.5＋0.5×PR(C)/1
  - PR(B)＝0.5＋0.5×(PR(A)/2)
  - PR(C)＝0.5＋0.5×(PR(A)/2＋PR(B)/1)
- 解这个三元方程组，得到：
  - PR(A)＝14/13＝1.0769
  - PR(B)＝10/13＝0.76923
  - PR(C)＝15/13＝1.1538



2.3.4 PageRank

- 从最后的PageRank值可以清楚地看出：
- 在这3个网页中，网页C的重要性最高，它的得分为1.1538分；其次是网页A；最后是网页B。

- 这种通过链接关系的分析得到网页重要性是合理的
- 网页C在这个局部是更被“认可”的，它被网页A和网页B指向；
- 网页A被更高级别的网页C“认可”，而网页B被网页A“认可”，这样网页A和网页B同样具有一个Backlink。
- 由于网页C的级别大于网页A，因此最终网页A的重要性大于网页B。