一、PageRank

1 概述

Google创始人拉里·佩奇和谢尔盖·布林于1997年构建早期的搜索系统原型时提出的链接分析算法，自从Google在商业上获得空前的成功后，该算法也成为其他搜索引擎和学术界十分关注的计算模型。目前很多重要的链接分析算法都是在PageRank算法基础上衍生出来的。PageRank是Google用于用来**标识网页的等级/重要性的一种方法**，是Google用来衡量一个网站的好坏的标准。在揉合了诸如Title标识和Keywords标识等所有其它因素之后，Google通过PageRank来调整结果，使那些更具"等级/重要性"的网页在搜索结果中另网站排名获得提升，从而提高搜索结果的相关性和质量。其级别从0到10级，10级为满分。PR值越高说明该网页越受欢迎（越重要）。例如：一个PR值为1的网站表明这个网站不太具有流行度，而PR值为7到10则表明这个网站非常受欢迎（或者说极其重要）。一般PR值达到4，就算是一个不错的网站了。Google把自己的网站的PR值定到10，这说明Google这个网站是非常受欢迎的，也可以说这个网站非常重要。

2. 从入链数量到Pagerank

在PageRank提出之前，已经有研究者提出利用网页的**入链**数量来进行链接分析计算，这种入链方法假设一个网页的入链越多，则该网页越重要。早期的很多搜索引擎也采纳了入链数量作为链接分析方法，对于搜索引擎效果提升也有较明显的效果。 PageRank**除了考虑到入链数量的影响，还参考了网页质量因素**(例如，可能虽然一个网页的入链很多，但是这些都是广告)，两者相结合获得了更好的网页重要性评价标准。 对于某个互联网网页A来说，该网页PageRank的计算基于以下两个基本假设：

- 数量假设：在Web图模型中，如果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：指向页面A的入链质量不同，质量高的页面会通过链接向其他页面传递更多的权重。**所以越是质量高的页面指向页面A，则页面A越重要。**

例如，在微博上，如果我们想要计算某个人的影响力，该怎么做呢？一个人的微博粉丝数并不一定等于他的实际影响力。如果按照 PageRank 算法，还需要看这些粉丝的质量如何。如果有很多明星或者大 V 关注，那么这个人的影响力一定很高。如果粉丝是通过购买僵尸粉得来的，那么即使粉丝数再多，影响力也不高。

利用以上两个假设，PageRank算法刚开始赋予每个网页相同的重要性得分，通过**迭代**计算来更新每个页面节点的PageRank得分，直到得分**稳定**为止。 PageRank计算得出的结果是网页的重要性评价，这和用户输入的查询是没有任何关系的，即算法是主题无关的。假设有一个搜索引擎，其相似度计算函数不考虑内容相似因素，完全采用PageRank来进行排序，那么这个搜索引擎的表现是什么样子的呢？这个搜索引擎对于任意不同的查询请求，返回的结果都是相同的，即返回PageRank值最高的页面。

3. 算法原理

3.1 朴素pagerank

PageRank的计算充分利用了两个假设：数量假设和质量假设。步骤如下： **1）在初始阶段：** 网页通过链接关系构建起Web图，每个页面设置相同的PageRank值，通过若干轮的计算，会得到每个页面所获得的最终PageRank值。随着每一轮的计算进行，网页当前的PageRank值会不断得到更新。

2）在一轮中更新页面PageRank得分的计算方法： 在一轮更新页面PageRank得分的计算中，每个页面将其当前的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。而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。

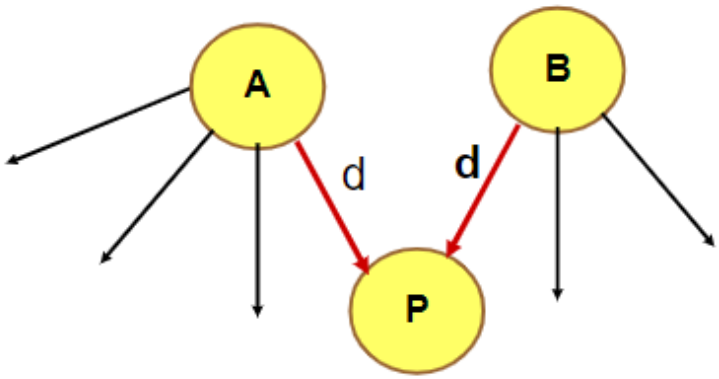
3.2 改进的pagerank

（1）问题一：等级泄露(rank leak)

如果一个网页没有出链(孤立节点，dangling node)，就像是一个黑洞一样，**吸收了其他网页的影响力而不释放**，最终会导致其他网页的 PR 值为 0。

（2）问题二：等级沉没（rank sink）-- 成环

为了解决简化模型中存在的等级泄露和等级沉没的问题，拉里·佩奇提出了 PageRank 的随机浏览模型。他假设了这样一个场景：用户并不都是按照跳转链接的方式来上网，还有一种可能是不论当前处于哪个页面，都有概率访问到其他任意的页面，比如说用户就是要直接输入网址访问其他页面，虽然这个概率比较小。 所以他定义了阻尼因子 d，这个因子代表了用户按照跳转链接来上网的概率，通常可以取一个固定值 0.85，**而 1-d=0.15 则代表了用户不是通过跳转链接的方式来访问网页的**，比如直接输入网址。 因为加入了阻尼因子 d，一定程度上解决了等级泄露和等级沉没的问题。（因为总有一定的概率会跳出原来的环\没有出链的点）



PageRank of P is

d*[(PageRank of A)/4 + (PageRank of B)/3)] +(1-d)/n

通过数学定理（这里不进行讲解）也可以证明，最终 PageRank 随机浏览模型是可以收敛的，也就是可以得到一个稳定正常的 PR 值。

二、HITS算法

1. 算法来源

HITS算法的全称是Hyperlink-Induced Topic Search。在HITS算法中，每个页面被赋予两个属性：**hub属性和authority属性** 。同时，网页被分为两种：hub页面和authority页面。hub，中心的意思，所以hub页面指那些包含了很多指向authority页面的链接的网页，比如一些**门户网站**；authority页面则指那些包含有实质性内容的网页。HITS算法的目的是：当用户查询时，返回给用户高质量的authority页面。

2. 算法原理

HITS算法基于下面两个假设：

> - 一个高质量的authority页面会被很多高质量的hub页面所指向。
> - 一个高质量的hub页面会指向很多高质量的authority页面。
>
> (mutually reinforcing relationship)

什么叫"高质量"，这由每个页面的hub值和authority值确定。其确定方法为：

> - 页面hub值等于所有它指向的页面的authority值之和。
> - 页面authority值等于所有指向它的页面的hub值之和。

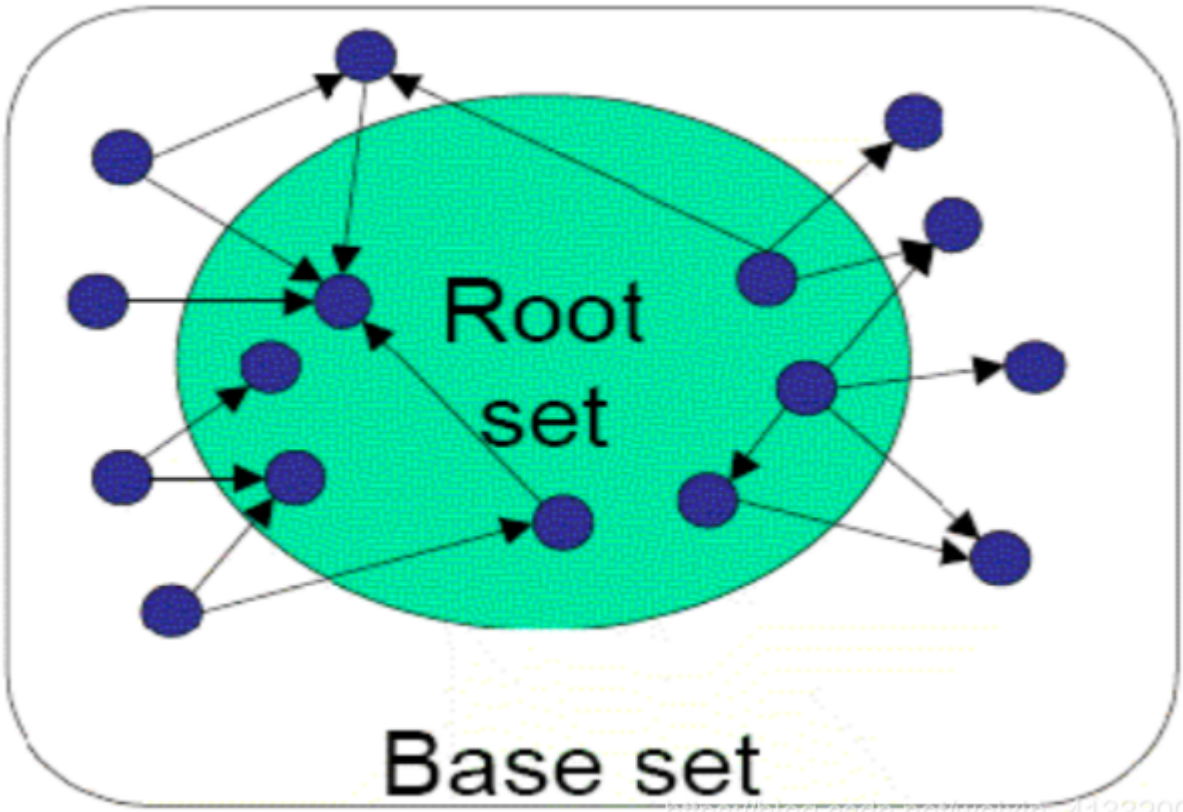
整个互联网中的网页之间的关系可以抽象为一个有向图G=(V,E)，当有一个搜索请求产生时（不妨设关键字为σ）,我们可以取所有包含关键字σ的网页组成的集合Q_σ为初始集合，并在这个集合上运行我们的HITS算法。然而，这个集合却有着明显的缺陷：这个集合可能非常之大，大到包含了数百万个网页，而这显然不是理想的集合大小。于是，我们进而想找到一个更小的集合S_σ,满足以下条件：

- Sσ确实足够小。
- Sσ包含很多与查询相关的页面。
- Sσ包含很多高质量的authority页面。

如何找到这个Sσ集合？我们假设用户输入关键字搜索，搜索引擎使用一个基于文本的引擎进行搜索。然后我们取排名（按照相关度排名）最靠前的t（t一般取200左右）个网页作为初始集合，记为根集合Rσ。这个集合满足我们上面提到的前两个条件，但是还远远不能满足第三个条件。

于是，我们需要扩展Rσ。一般认为，一个与关键字相关的高质量的网页即使不在Rσ中，那也很可能在Rσ中有某些网页指向它。

一开始我们令Sσ = Rσ。然后我们将**所有被Rσ中网页所指向的网页加入到Sσ中**，再把一定数量的指向Rσ集合中网页的那些网页（每个Rσ中网页最多能添加d个指向它的网页）加入到Sσ中。为了保证Sσ集合的合适的大小，d不能太大，一般设置为50左右。通常情况下，扩展之后集合的大小为1000~5000个网页，满足上面的三个条件。



> Note that while Rσ (root set) may fail to contain some “important” authorities, Sσ (base set) will probably contain them.

现在，就可以开始计算hub值和authority值了。 我们用h(p)表示页面p的hub值，a(p)表示页面p的authority值。首先令每个页面的初始hub值h(p)为1，初始authority值a(p)也为1。然后就开始迭代计算的过程：

网页p在此轮迭代中的Authority权值即为所有指向网页p页面的Hub权值之和：

$$\forall p, a(p) = \sum h(i)$$

网页p的Hub分值即为所指向的页面的Authority权值之和：

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

每一轮迭代结束，都需要进行标准化，使

$$\sum_{i=1}^{n} h(i)^2 = \sum_{i=1}^{n} a(i)^2 = 1$$

.

什么时候迭代结束呢？我们可以设置一个迭代次数上限k来控制，或者设定一个阈值，当变化小于阈值的时候迭代结束。然后只要返回给用户authority值靠前的十几个网页就行了。