浅析PageRank算法

作者 张洋 | 发布于 2012-07-02

Google PageRank 搜索引擎 算法

很早就对Google的PageRank算法很感兴趣，但一直没有深究，只有个轮廓性的概念。前几天趁团队outing的机会，在动车上看了一些相关的资料（PS：在动车上看看书真是一种享受），趁热打铁，将所看的东西整理成此文。

本文首先会讨论搜索引擎的核心难题，同时讨论早期搜索引擎关于结果页面重要性评价算法的困境，借此引出PageRank产生的背景。第二部分会详细讨论PageRank的思想来源、基础框架，并结合互联网页面拓扑结构讨论PageRank处理Dead Ends及平滑化的方法。第三部分讨论Topic-Sensitive PageRank算法。最后将讨论对PageRank的Spam攻击方法：Spam Farm以及搜索引擎对Spam Farm的防御。

搜索引擎的难题

Google早已成为全球最成功的互联网搜索引擎，但这个当前的搜索引擎巨无霸却不是最早的互联网搜索引擎，在Google出现之前，曾出现过许多通用或专业领域搜索引擎。Google最终能击败所有竞争对手，很大程度上是因为它解决了困扰前辈们的最大难题：对搜索结果按重要性排序。而解决这个问题的算法就是PageRank。毫不夸张的说，是PageRank算法成就了Google今天的低位。要理解为什么解决这个难题如此重要，我们先来看一下搜索引擎的核心框架。

搜索引擎的核心框架

虽然搜索引擎已经发展了很多年，但是其核心却没有太大变化。从本质上说，搜索引擎是一个资料检索系统，搜索引擎拥有一个资料库（具体到这里就是互联网页面），用户提交一个检索条件（例如关键词），搜索引擎返回符合查询条件的资料列表。理论上检索条件可以非常复杂，为了简单起见，我们不妨设检索条件是一至多个以空格分隔的词，而其表达的语义是同时含有这些词的资料（等价于布尔代数的逻辑与）。例如，提交“张洋 博客”，意思就是“给我既含有‘张洋’又含有‘博客’词语的页面”，以下是Google对这条关键词的搜索结果：

可以看到我的博客出现在第五条，而第四条是我之前在博客园的博客。

当然，实际上现在的搜索引擎都是有分词机制的，例如如果以“张洋的博客”为关键词，搜索引擎会自动将其分解为“张洋 的 博客”三个词，而“的”作为停止词（Stop Word）会被过滤掉。关于分词及词权评价算法（如TF-IDF算法）是一个很大的话题，这里就不展开讨论了，为了简单此处可以将搜索引擎想象为一个只会机械匹配词语的检索系统。

这样看来，建立一个搜索引擎的核心问题就是两个：1、建立资料库；2、建立一种数据结构，可以根据关键词找到含有这个词的页面。

第一个问题一般是通过一种叫爬虫（Spider）的特殊程序实现的（当然，专业领域搜索引擎例如某个学术会议的论文检索系统可能直接从数据库建立资料库），简单来说，爬虫就是从一个页面出发（例如新浪首页），通过HTTP协议通信获取这个页面的所有内容，把这个页面url和内容记录下来（记录到资料库），然后分析页面中的链接，再去分别获取这些链接链向页面的内容，记录到资料库后再分析这个页面的链接……重复这个过程，就可以将整个互联网的页面全部获取下来（当然这是理想情况，要求整个Web是一个强连通（Strongly Connected），并且所有页面的robots协议允许爬虫抓取页面，为了简单，我们仍然假设Web是一个强连通图，且不考虑robots协议）。抽象来看，可以将资料库看做一个巨大的key-value结构，key是页面url，value是页面内容。

第二个问题是通过一种叫倒排索引（inverted index）的数据结构实现的，抽象来说倒排索引也是一组key-value结构，key是关键词，value是一个页面编号集合（假设资料库中每个页面有唯一编号），表示这些页面含有这个关键词。本文不详细讨论倒排索引的建立方法。

有了上面的分析，就可以简要说明搜索引擎的核心动作了：搜索引擎获取“张洋 博客”查询条件，将其分为“张洋”和“博客”两个词。然后分别从倒排索引中找到“张洋”所对应的集合，假设是{1， 3， 6， 8， 11， 15}；“博客”对应的集合是{1， 6， 10， 11， 12， 17， 20， 22}，将两个集合做交运算（intersection），结果是{1， 6， 11}。最后，从资料库中找出1、6、11对应的页面返回给用户就可以了。

搜索引擎的核心难题

上面阐述了一个非常简单的搜索引擎工作框架，虽然现代搜索引擎的具体细节原理要复杂的多，但其本质却与这个简单的模型并无二异。实际Google在上述两点上相比其前辈并无高明之处。其最大的成功是解决了第三个、也是最为困难的问题：如何对查询结果排序。

我们知道Web页面数量非常巨大，所以一个检索的结果条目数量也非常多，例如上面“张洋 博客”的检索返回了超过260万条结果。用户不可能从如此众多的结果中一一查找对自己有用的信息，所以，一个好的搜索引擎必须想办法将“质量”较高的页面排在前面。其实直观上也可以感觉出，在使用搜索引擎时，我们并不太关心页面是否够全（上百万的结果，全不全有什么区别？而且实际上搜索引擎都是取top，并不会真的返回全部结果。），而很关心前一两页是否都是质量较高的页面，是否能满足我们的实际需求。

因此，对搜索结果按重要性合理的排序就成为搜索引擎的最大核心，也是Google最终成功的突破点。

早期搜索引擎的做法

不评价

这个看起来可能有点搞笑，但实际上早期很多搜索引擎（甚至包括现在的很多专业领域搜索引擎）根本不评价结果重要性，而是直接按照某自然顺序（例如时间顺序或编号顺序）返回结果。这在结果集比较少的情况下还说得过去，但是一旦结果集变大，用户叫苦不迭，试想让你从几万条质量参差不齐的页面中寻找需要的内容，简直就是一场灾难，这也注定这种方法不可能用于现代的通用搜索引擎。

基于检索词的评价

后来，一些搜索引擎引入了基于检索关键词去评价搜索结构重要性的方法，实际上，这类方法如TF-IDF算法在现代搜索引擎中仍在使用，但其已经不是评价质量的唯一指标。完整描述TF-IDF比较繁琐，本文这里用一种更简单的抽象模型描述这种方法。

基于检索词评价的思想非常朴素：和检索词匹配度越高的页面重要性越高。“匹配度”就是要定义的具体度量。一个最直接的想法是关键词出现次数越多的页面匹配度越高。还是搜索“张洋 博客”的例子：假设A页面出现“张洋”5次，“博客”10次；B页面出现“张洋”2次，“博客”8次。于是A页面的匹配度为5 + 10 = 15，B页面为2 + 8 = 10，于是认为A页面的重要性高于B页面。很多朋友可能意识到这里的不合理性：内容较长的网页往往更可能比内容较短的网页关键词出现的次数多。因此，我们可以修改一下算法，用关键词出现次数除以页面总词数，也就是通过关键词占比作为匹配度，这样可以克服上面提到的不合理。

早期一些搜索引擎确实是基于类似的算法评价网页重要性的。这种评价算法看似依据充分、实现直观简单，但却非常容易受到一种叫“Term Spam”的攻击。

Term Spam

其实从搜索引擎出现的那天起，spammer和搜索引擎反作弊的斗法就没有停止过。Spammer是这样一群人——试图通过搜索引擎算法的漏洞来提高目标页面（通常是一些广告页面或垃圾页面）的重要性，使目标页面在搜索结果中排名靠前。

现在假设Google单纯使用关键词占比评价页面重要性，而我想让我的博客在搜索结果中排名更靠前（最好排第一）。那么我可以这么做：在页面中加入一个隐藏的html元素（例如一个div），然后其内容是“张洋”重复一万次。这样，搜索引擎在计算“张洋 博客”的搜索结果时，我的博客关键词占比就会非常大，从而做到排名靠前的效果。更进一步，我甚至可以干扰别的关键词搜索结果，例如我知道现在欧洲杯很火热，我就在我博客的隐藏div里加一万个“欧洲杯”，当有用户搜索欧洲杯时，我的博客就能出现在搜索结果较靠前的位置。这种行为就叫做“Term Spam”。

早期搜索引擎深受这种作弊方法的困扰，加之基于关键词的评价算法本身也不甚合理，因此经常是搜出一堆质量低下的结果，用户体验大大打了折扣。而Google正是在这种背景下，提出了PageRank算法，并申请了专利保护。此举充分保护了当时相对弱小Google，也使得Google一举成为全球首屈一指的搜索引擎。

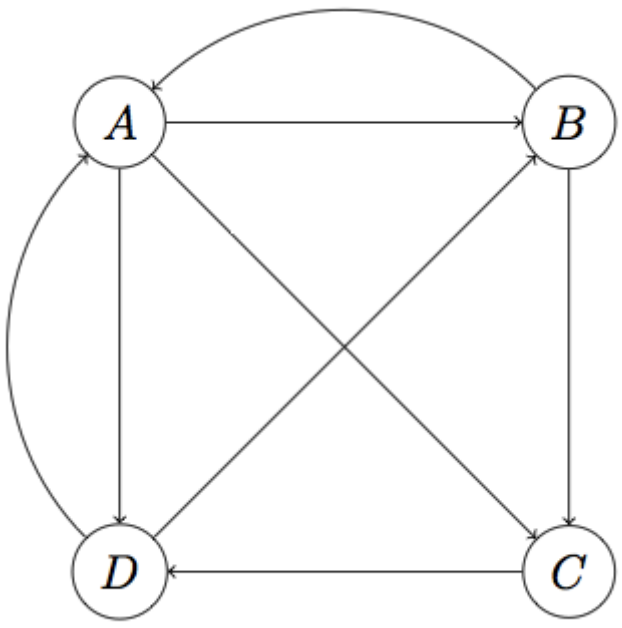
PageRank算法

上文已经说到，PageRank的作用是评价网页的重要性，以此作为搜索结果的排序重要依据之一。实际中，为了抵御spam，各个搜索引擎的具体排名算法是保密的，PageRank的具体计算方法也不尽相同，本节介绍一种最简单的基于页面链接属性的PageRank算法。这个算法虽然简单，却能揭示PageRank的本质，实际上目前各大搜索引擎在计算PageRank时链接属性确实是重要度量指标之一。

简单PageRank计算

首先，我们将Web做如下抽象：1、将每个网页抽象成一个节点；2、如果一个页面A有链接直接链向B，则存在一条有向边从A到B（多个相同链接不重复计算边）。因此，整个Web被抽象为一张有向图。

现在假设世界上只有四张网页：A、B、C、D，其抽象结构如下图：



显然这个图是强连通的（从任一节点出发都可以到达另外任何一个节点）。

然后需要用一种合适的数据结构表示页面间的连接关系。其实，PageRank算法是基于这样一种背景思想：被用户访问越多的网页更可能质量越高，而用户在浏览网页时主要通过超链接进行页面跳转，因此我们需要通过分析超链接组成的拓扑结构来推算每个网页被访问频率的高低。最简单的，我们可以假设当一个用户停留在某页面时，跳转到页面上每个被链页面的概率是相同的。例如，上图中A页面链向B、C、D，所以一个用户从A跳转到B、C、D的概率各为1/3。设一共有N个网页，则可以组织这样一个N维矩阵：其中i行j列的值表示用户从页面j转到页面i的概率。这样一个矩阵叫做转移矩阵（Transition Matrix）。下面的转移矩阵M对应上图：

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 0 \end{bmatrix}$$

然后，设初始时每个页面的rank值为1/N，这里就是1/4。按A-D顺序将页面rank为向量v：

$$v = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

注意，M第一行分别是A、B、C和D转移到页面A的概率，而v的第一列分别是A、B、C和D当前的rank，因此用M的第一行乘以v的第一列，所得结果就是页面A最新rank的合理估计，同理，Mv的结果就分别代表A、B、C、D新rank：

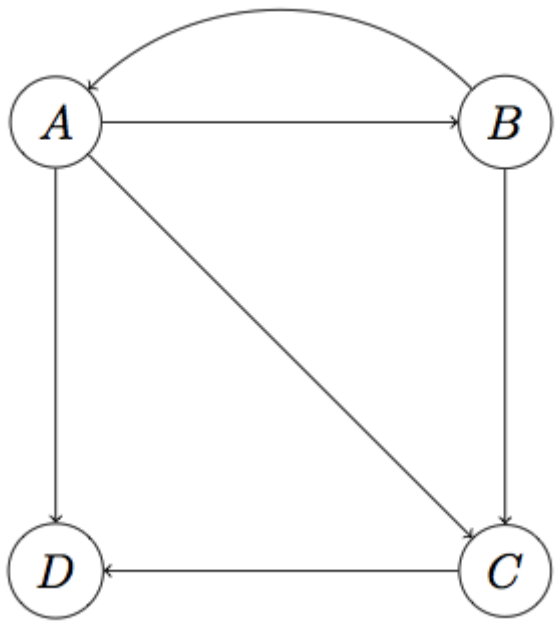
$$Mv = \begin{bmatrix} 1/4 \\ 5/24 \\ 5/24 \\ 1/3 \end{bmatrix}$$

然后用M再乘以这个新的rank向量，又会产生一个更新的rank向量。迭代这个过程，可以证明v最终会收敛，即v约等于Mv，此时计算停止。最终的v就是各个页面的pagerank值。例如上面的向量经过几步迭代后，大约收敛在（1/4, 1/4, 1/5, 1/4），这就是A、B、C、D最后的pagerank。

处理Dead Ends

上面的PageRank计算方法假设Web是强连通的，但实际上，Web并不是强连通（甚至不是联通的）。下面看看PageRank算法如何处理一种叫做Dead Ends的情况。

所谓Dead Ends，就是这样一类节点：它们不存在外链。看下面的图：



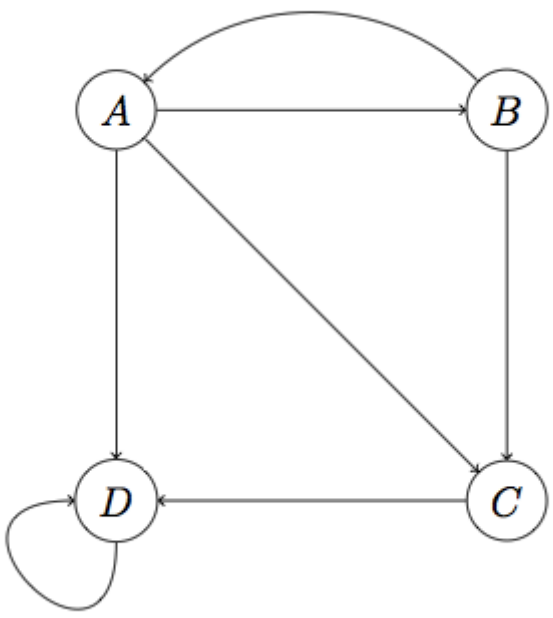
注意这里D页面不存在外链，是一个Dead End。上面的算法之所以能成功收敛到非零值，很大程度依赖转移矩阵这样一个性质：每列的加和为1。而在这个图中，M第四列将全为0。在没有Dead Ends的情况下，每次迭代后向量v各项的和始终保持为1，而有了Dead Ends，迭代结果将最终归零（要解释为什么会这样，需要一些矩阵论的知识，比较枯燥，此处略）。

处理Dead Ends的方法如下：迭代拿掉图中的Dead Ends节点及Dead Ends节点相关的边（之所以迭代拿掉是因为当目前的Dead Ends被拿掉后，可能会出现一批新的Dead Ends），直到图中没有Dead Ends。对剩下部分计算rank，然后以拿掉Dead Ends逆向顺序反推Dead Ends的rank。

以上图为例，首先拿到D和D相关的边，D被拿到后，C就变成了一个新的Dead Ends，于是拿掉C，最终只剩A、B。此时可很容易算出A、B的PageRank均为1/2。然后我们需要反推Dead Ends的rank，最后被拿掉的是C，可以看到C前置节点有A和B，而A和B的出度分别为3和2，因此C的rank为：1/2 * 1/3 + 1/2 * 1/2 = 5/12；最后，D的rank为：1/2 * 1/3 + 5/12 * 1 = 7/12。所以最终的PageRank为（1/2, 1/2, 5/12, 7/12）。

Spider Traps及平滑处理

可以预见，如果把真实的Web组织成转移矩阵，那么这将是一个极为稀疏的矩阵，从矩阵论知识可以推断，极度稀疏的转移矩阵迭代相乘可能会使得向量v变得非常不平滑，即一些节点拥有很大的rank，而大多数节点rank值接近0。而一种叫做Spider Traps节点的存在加剧了这种不平滑。例如下图：



D有外链所以不是Dead Ends，但是它只链向自己（注意链向自己也算外链，当然同时也是个内链）。这种节点叫做Spider Trap，如果对这个图进行计算，会发现D的rank越来越大趋近于1，而其它节点rank值几乎归零。

为了克服这种由于矩阵稀疏性和Spider Traps带来的问题，需要对PageRank计算方法进行一个平滑处理，具体做法是加入“心灵转移（teleporting）”。所谓心灵转移，就是我们认为在任何一个页面浏览的用户都有可能以一个极小的概率瞬间转移到另外一个随机页面。当然，这两个页面可能不存在超链接，因此不可能真的直接转移过去，心灵转移只是为了算法需要而强加的一种纯数学意义的概率数字。

加入心灵转移后，向量迭代公式变为：

$$v' = (1-\beta)Mv + e\frac{\beta}{N}$$

其中β往往被设置为一个比较小的参数（0.2或更小），e为N维单位向量，加入e的原因是这个公式的前半部分是向量，因此必须将β/N转为向量才能相加。这样，整个计算就变得平滑，因为每次迭代的结果除了依赖转移矩阵外，还依赖一个小概率的心灵转移。

以上图为例，转移矩阵M为：

$$M = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 0 \\ 1/3 & 0 & 1 & 1 \end{bmatrix}$$

设β为0.2，则加权后的M为：

$$M = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 0 \\ 4/15 & 2/5 & 0 & 0 \\ 4/15 & 0 & 4/5 & 4/5 \end{bmatrix}$$

因此：

$$v' = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 0 \\ 4/15 & 2/5 & 0 & 0 \\ 4/15 & 0 & 4/5 & 4/5 \end{bmatrix} v + \begin{bmatrix} 1/20 \\ 1/20 \\ 1/20 \\ 1/20 \end{bmatrix}$$

如果按这个公式迭代算下去，会发现Spider Traps的效应被抑制了，从而每个页面都拥有一个合理的pagerank。

Topic-Sensitive PageRank

其实上面的讨论我们回避了一个事实，那就是“网页重要性”其实没一个标准答案，对于不同的用户，甚至有很大的差别。例如，当搜索“苹果”时，一个数码爱好者可能是想要看iphone的信息，一个果农可能是想看苹果的价格走势和种植技巧，而一个小朋友可能在找苹果的简笔画。理想情况下，应该为每个用户维护一套专用向量，但面对海量用户这种方法显然不可行。所以搜索引擎一般会选择一种称为Topic-Sensitive的折中方案。Topic-Sensitive PageRank的做法是预定义几个话题类别，例如体育、娱乐、科技等等，为每个话题单独维护一个向量，然后想办法关联用户的话题倾向，根据用户的话题倾向排序结果。

Topic-Sensitive PageRank分为以下几步：

1、确定话题分类。

一般来说，可以参考Open Directory（DMOZ）的一级话题类别作为topic。目前DMOZ的一级topic有：Arts（艺术）、Business（商务）、Computers（计算机）、Games（游戏）、Health（医疗健康）、Home（居家）、Kids and Teens（儿童）、News（新闻）、Recreation（娱乐修养）、Reference（参考）、Regional（地域）、Science（科技）、Shopping（购物）、Society（人文社会）、Sports（体育）。

2、网页topic归属。

这一步需要将每个页面归入最合适的分类，具体归类有很多算法，例如可以使用TF-IDF基于词素归类，也可以聚类后人工归类，具体不再展开。这一步最终的结果是每个网页被归到其中一个topic。

3、分topic向量计算。

在Topic-Sensitive PageRank中，向量迭代公式为

$$v' = (1-\beta)Mv + s\frac{\beta}{|s|}$$

首先是单位向量e变为了s。s是这样一个向量：对于某topic的s，如果网页k在此topic中，则s中第k个元素为1，否则为0。注意对于每一个topic都有一个不同的s。而|s|表示s中1的数量。

还是以上面的四张页面为例，假设页面A归为Arts，B归为Computers，C归为Computers，D归为Sports。那么对于Computers这个topic，s就是：

$$s = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

而|s|=2。因此，迭代公式为：

$$v' = \begin{bmatrix} 0 & 2/5 & 0 & 0 \\ 4/15 & 0 & 0 & 0 \\ 4/15 & 2/5 & 0 & 0 \\ 4/15 & 0 & 4/5 & 4/5 \end{bmatrix} v + \begin{bmatrix} 0 \\ 1/10 \\ 1/10 \\ 0 \end{bmatrix}$$

最后算出的向量就是Computers这个topic的rank。如果实际计算一下，会发现B、C页在这个topic下的权重相比上面非Topic-Sensitive的rank会升高，这说明如果用户是一个倾向于Computers topic的人（例如程序员），那么在给他呈现的结果中B、C会更重要，因此可能排名更靠前。

4、确定用户topic倾向。

最后一步就是在用户提交搜索时，确定用户的topic倾向，以选择合适的rank向量。主要方法有两种，一种是列出所有topic让用户自己选择感兴趣的项目，这种方法在一些社交问答网站注册时经常使用；另外一种方法就是通过某种手段（如cookie跟踪）跟踪用户的行为，进行数据分析判断用户的倾向，这本身也是一个很有意思的话题，按时这个话题超出本文的范畴，不再展开细说。

针对PageRank的Spam攻击与反作弊

上文说过，Spammer和搜索引擎反作弊工程师的斗法从来就没停止过。实际上，只要是算法，就一定有spam方法，不存在无懈可击的排名算法。下面看一下针对PageRank的spam。

Link Spam

回到文章开头的例子，如果我想让我的博客在搜索“张洋 博客”时排名靠前，显然在PageRank算法下靠Term Spam是无法实现的。不过既然我明白了PageRank主要靠内链数计算页面权重，那么我是不是可以考虑建立很多空架子网站，让这些网站都链接到我博客首页，这样是不是可以提高我博客首页的PageRank？很不幸，这种方法行不通。再看下PageRank算法，一个页面会将权重均匀散播给被链接网站，所以除了内链数外，上游页面的权重也很重要。而我那些空架子网站本身就没啥权重，所以来自它们的内链并不能起到提高我博客首页PageRank的作用，这样只是自娱自乐而已。

所以，Spam PageRank的关键就在于想办法增加一些高权重页面的内链。下面具体看一下Link Spam怎么做。

首先明确将页面分为几个类型：

1、目标页

目标页是spammer要提高rank的页面，这里就是我的博客首页。

2、支持页

支持页是spammer能完全控制的页面，例如spammer自己建立的站点中页面，这里就是我上文所谓的空架子页面。

3、可达页

可达页是spammer无法完全控制，但是可以有接口供spammer发布链接的页面，例如天涯社区、新浪博客等等这种用户可发帖的社区或博客站。

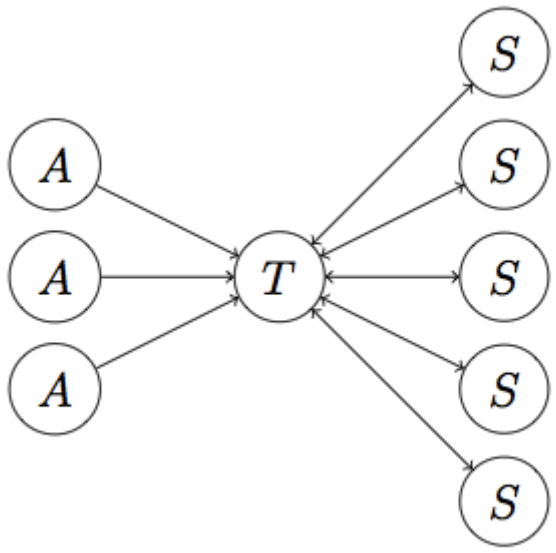
4、不可达页

这是那些spammer完全无法发布链接的网站，例如政府网站、百度首页等等。

作为一个spammer，我能利用的资源就是支持页和可达页。上面说过，单纯通过支持页是没有办法spam的，因此我要做的第一件事情就是尽量找一些rank较高的可达页去加上对我博客首页的链接。例如我可以去天涯、猫扑等地方回个这样的贴：“楼主的帖子很不错！精彩内容：http://codinglabs.org”。我想大家一定在各大社区没少见这种帖子，这就是有人在做spam。

然后，再通过大量的支持页放大rank，具体做法是让每个支持页和目标页互链，且每个支持页只有一条链接。

这样一个结构叫做Spam Farm，其拓扑图如下：



其中T是目标页，A是可达页，S是支持页。下面计算一下link spam的效果。

设T的总rank为y，则y由三部分组成：

1、可达页的rank贡献，设为x。

2、心灵转移的贡献，为β/n。其中n为全部网页的数量，β为转移参数。

3、支持页的贡献：

设有m个支持页，因为每个支持页只和T有链接，所以可以算出每个支持页的rank为：

$$\frac{(1-\beta)y}{m} + \frac{\beta}{n}$$

则支持页贡献的全部rank为：

$$m(1-\beta)(\frac{(1-\beta)y}{m} + \frac{\beta}{n})$$

因此可以得到：

$$y = m(1-\beta)(\frac{(1-\beta)y}{m} + \frac{\beta}{n}) + x + \frac{\beta}{n}$$

由于相对β，n非常巨大，所以可以认为β/n近似于0。 简化后的方程为：

$$y = m(1-\beta)(\frac{(1-\beta)y}{m}) + x$$

解方程得：

$$y = x\frac{1}{2\beta-\beta^2}$$

假设β为0.2，则1/(2β-β^2) = 2.77则这个spam farm可以将x约放大2.7倍。因此如果起到不错的spam效果。

Link Spam反作弊

针对spammer的link spam行为，搜索引擎的反作弊工程师需要想办法检测这种行为，一般来说有两类方法检测link spam。

网络拓扑分析

一种方法是通过对网页的图拓扑结构分析找出可能存在的spam farm。但是随着Web规模越来越大，这种方法非常困难，因为图的特定结构查找是时间复杂度非常高的一个算法，不可能完全靠这种方法反作弊。

TrustRank

更可能的一种反作弊方法是叫做一种TrustRank的方法。

说起来TrustRank其实数学本质上就是Topic-Sensitive Rank，只不过这里定义了一个“可信网页”的虚拟topic。所谓可信网页就是上文说到的不可达页，或者说没法spam的页面。例如政府网站（被黑了的不算）、新浪、网易门户首页等等。一般是通过人力或者其它什么方式选择出一个“可信网页”集合，组成一个topic，然后通过上文的Topic-Sensitive算法对这个topic进行rank计算，结果叫做TrustRank。

TrustRank的思想很直观：如果一个页面的普通rank远高于可信网页的topic rank，则很可能这个页面被spam了。

设一个页面普通rank为P，TrustRank为T，则定义网页的Spam Mass为：(P – T)/P。

Spam Mass越大，说明此页面为spam目标页的可能性越大。

总结

这篇文章是我对一些资料的归纳汇总，简单介绍了PageRank的背景、作用、计算方法、变种、Spam及反作弊等内容。为了突出重点我简化了搜索引擎的模型，当然在实际中搜索引擎远没有这么简单，真实算法也一定非常复杂。不过目前几乎所有现代搜索引擎页面权重的计算方法都基于PageRank及其变种。因为我没做过搜索引擎相关的开发，因此本文内容主要是基于现有文献的客观总结，稍加一点我的理解。

文中的图使用PGF/TikZ for Tex绘制：http://www.texample.net/tikz/。

参考文献

[1] Anand Rajaraman, Jeffrey D. Ullman, Mining of Massive Datasets. 2010-2011

[2] S. Brin and L. Page, “Anatomy of a large-scale hypertextual web search engine,” Proc. 7th Intl. World-Wide-Web Conference, pp. 107–117, 1998.

[3] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Weiner, “Graph structure in the web,” Computer Networks 33:1–6, pp. 309–320, 2000.

[4] T.H. Haveliwala, “Topic-sensitive PageRank,” Proc. 11th Intl. World-Wide-Web Conference, pp. 517–526, 2002

[5] Z. Gy¨ongi, H. Garcia-Molina, and J. Pedersen, “Combating link spam with trustrank,” Proc. 30th Intl. Conf. on Very Large Databases, pp. 576–587, 2004.
