Abstract

Two major techniques have been proposed for using the structure of links in the World Wide Web to determine the relative significance of Web Pages. The PageRank algorithm [Brin 98], which is a critical part of the Google search engine, gives a single measure of importance of each page in the Web. The HITS algorithm [Kleinberg 98] applies to a set of pages believed relevant to a given query, and assigns two values to each page: the degree to which the page is a hub and the degree to which it is an authority. Both algorithms have a natural interpretation in terms of a random walk over the set of pages involved, and in both cases the computation involved amounts to computing an eigenvector over the transition matrix for this random walk.

This paper surveys the literature discussing these two techniques and their variants, and their connection to random walks and eigenvector computation. It also discusses the stability of these techniques under small changes in the Web link structure.

Survey: Eigenvector Analysis in Webpage Rankings

TR 2003-835, Computer Science Department, New York University

In fulfillment of the survey paper requirement for Ph.D program

©by Hung-Hsien Chang, hubert@cs.nyu.edu

April 2002

Contents

Chapter 1

Search Engine circa 1998: Eigenvector

Search engine, more specifically search engine methods and algorithms around 1998, is the focus of this survey. This survey gives you the intuitions of how and why the search algorithms work, their variants and applications. Mathematical proofs are reduced to its minimum form and are presented in non-theorem style.

1.1 The Birth of Search Engine

Since the debut of World Wide Web (abbrev. the Web or WWW), webpages and websites have grown steadily. It becomes difficult to find a pages related to one's need. There are two approaches to solve this problem. One is to categorize the webpages, and the other is to resort to a search engine.

The first method is to build a directory, such as Yahoo. Yahoo has tried to systematically categorize the webpages manually but the size of the work is overwhelmingly large. It is not possible to efficiently categorize all contents manually.

The second method is to resort to a search engine. A search engine usually comes with a fleet of crawlers that traverse the network and collect the pages for local storage. Search engine server will pre-processes the cached content to extract useful information for search queries. When a user enter a query, the search engine will respond by returning the pre-processed and prepared data.

1.2 Before 1998: Indexing, Terms Frequency, and Click Through

We can trace back the origin of search engine to the database community but they do not call it a search engine. Search engine is therefore a terminology used for the Web. The search engine differs from traditional query engine in that it is equipped crawlers that will collect data from 'outside' its host. The full set of data and content a search engine handles are distributed over the network. Traditional query engine usually has its data in the same host. There is no distributed database with the same scale of the Web.

As described in [Brin 98], the implementations of search engines are usually kept as secrets within the commercial site and seldom be revealed to the public. But we can guess that most of the conventional search engine, if not all of them, use schemes such as indexing webpages, computing terms frequency, and tracking click through.

Indexing is a basic task for databases. Servers find out what terms appear in the documents and build a inverse index for the terms. In other words, we can find out which documents contain a given term. Terms frequency is used in estimating the relative relevance of the document. Salton [Salton 75][Salton 89] has pioneered many of the work in indexing and term frequency. His method is widely used in information retrieval. The Web environment provides a few new data sources that can track the users' activities. One of the data source is click through. When user traverse the web by following a hyperlink, the action will be recorded by the server. Search engine servers can record the activities of users on their choice of click through. Some search engines incorporate the click through as the indicator of importance. The higher the click through of a link, the higher it should be listed on the returned result.

Indexing for the Web is a daunting task. From the wealth and the growth of the Web, a returned list from a query term can be too long to be throughly examed by the users. A document can be represented by a term vector whose entry is the frequency of the term in the document. Term frequency has it limitation as we can not afford to use the complete set of lexicons from the web as the term base vector for a document. Most search engines combine the indexing and term frequency with other heuristics. One of them is click through. But as we know people tend to pick the top listing and this will generate a phenomena that a 'reliable' site will be more 'reliable' and the newer websites will never get a chance to move to the top rank even if they have good content. So none of the above methods or a combination of them have given a satisfactory search result.

Indexing and term frequency are interesting topics themselves. However, we will not diverge into techniques used before 1998. Readers who are interested in these areas can find more information in [Salton 89].

1.3 1998: Linkage Analysis

1998 is the turning point in the history of search engine. Researchers start to take notice at the hyperlinks as they have ignored in the past search engine algorithms. Using hyperlinks as a source for search result is not new; it was proposed in a closed hyperspace environment before[Botafogo 91]. The linkage analysis proposed in 1998 emphasizes the relationship between the links and the involvement of human intelligence. Webpages are produced by people, even those that are automatically generated through software templates are designed by people; therefore, a link most likely contains some level of human intelligence. It turns out that a link and its anchor text have a wealth of source of information that can be explored and used by a search engine. Individual hyperlink is a fragment of human intelligence and when we collect all these hyperlinks, we have a good estimate on the statistical evaluation of all webpage authors.

Linkage analysis revolves around modeling the Web as a large graph with directional edges (hyperlinks). It is natural to use a matrix to describe a graph algebraically. And it is natural to compute eigenvalues and eigenvectors for a matrix. We will see in the next few chapters that when we give the hyperlinks values and meanings, we will have an interesting interpretation on matrix's eigenvectors.

In fact, search engine methods in 1998 can be summarized by one word: eigenvector. Within the eigenvector frame work, we can classify algorithms into two categories. One is following Kelinberg's Hubs and Authorities [Kleinberg 98]; the other is Brin and Page's PageRank [Page 98]which is the main ingredient behind the search engine Google [Google].

We will describe each category in separated chapters. And a chapter summarizing on the stability of both algorithms is presented.

Chapter 2

Hubs and Authorities

Hubs and authorities are two properties proposed in Kleinberg's seminar paper[Kleinberg 98]. It is states that there are two types of pages. Pages contain many links that point to related material; we call these pages hubs. Authorities pages are content pages. These pages have contents. Under this assumption and the graph model where each page is represented as a node in a graph, we can separate the Web into a bipartite graph: a hub side and a authority side. But it is clear that many pages may have dual properties. They can be both hubs and authorities.

2.1 Basics

2.1.1 HITS: from Author's View

The algorithm described in this section is called HITS (Hypertext Induced Topics Search).

The greatest headache of a search engine is that for most queries, the returned result list is too long. In order to present the result in a 'quality' priority, we have to design a scheme to measure the 'quality'. Kleinberg's idea is simple: a good hub will point to many good authority, and a good authority will be pointed by many good hubs. Suppose we give the authority values and hub values to each page, then we expect the authority value of a page will be affected by the values of hubs pointing to it and the hub value of a page will be affected by the values of authorities to which it points. Let webpage i has authority value a_i and a hub value h_i, we can construct the following formula:

$$a_i = \sum_{j \to i} h_j \qquad\qquad h_i = \sum_{i \to j} a_j$$

In terms of the graph model, we have the following matrix computation. Let a be the authority vector and h be the hub vector. a_i and h_i will be the value of the authority and the hub for page i respectively. Let E be the graph matrix where $E_{ij} = 1$ if there is a hyperlink from page i to page j. With these notations, we now have:

$$a = E^t \times h \qquad\qquad h = E \times a$$

Notice that each authority and hub value is with respect to a specific query term. There are too many webpages containing a given term. Therefore we need to start from a limited set of pages. A quick candidate for such a set would be a returned result from a traditional search engine. However, in many search engines, they already limit the returned result size and it may not always be an ideal candidate set. To compensate the possible limitation, we can start from the returned set and grow it into a larger set. We call the returned result set from a traditional search engine a 'root set' and the final candidate a 'base set'. The idea is to collects all the urls that are pointed by root set and urls that are pointing to the root set. There are search engine services provide 'reversed' links[1]. Given a url, these services will return a list of pages with links pointing to this url. We have the following pseudo-code:

```
function get_base_set(query, k)
    root_set = { }
    returned_result = send_query_to_Alta_Vista(query)
    root_set = returned_result

    foreach this_url in returned_result
        url_pointing_set = url_pointing_to(this_url)
        root_set = base_set ∪ url_pointing_set
        url_pointed_set = url_pointed_by(this_url)
        root_set = base_set
                  ∪ select_top_k_url_from(url_pointed_set, k)
    end

    return root_set
end
```

Kleinberg regards the intra-links (links point to another page on the same site) to be navigational and therefore HITS does not collect these links. All the links between two different sites (domain names) are collected. From previous page's equation, after one iteration, we have $a = E^t \times E \times a$. Let $M = E^t \times E$, we can have the following loops to produce the hubs and authorities vectors:

z is a normalized vector without 0 in its entry.

[1]For instance, Alta Vista

```
$a = z$
$a' = z$
loop
    $a = a'$
    $a' = M \times a$
    Normalize $a'$
    $\delta = ||a' - a||_2$
until $(\delta < \epsilon)$
```

Now, we are ready to show that this computation always converges and it almost always converges to the largest eigenvector.

2.1.2 Convergence to the Largest Eigenvector

For simplicity, let us assume that the symmetric matrix E has different eigenvalues $\lambda_1 > \lambda_2 > \ldots > \lambda_n$. Because E is symmetric with no negative entry value, we will have $\lambda_1 > 0$ [Gloub 89]. Since these eigenvectors are linearly independent, we can use corresponding eigenvectors as a basis for the vector space. Let eigenvectors be $v_1, v_2, \ldots, v_n$. For any v in the space, we have

$$v = \sum_{i=1}^{n} c_i v_i$$

By applying A on both sides repeatedly for k times, we get

$$A^k v = \sum_{i=1}^{n} c_i {\lambda_i}^k v_i$$

But the ${\lambda_1}^k$ will be significantly larger than all other eigenvalues and its amplification to its own eigenvector will cause the $A^k v$ to swing to the v_1 direction. Therefore we will get v_1 eventually.

After obtaining the authority vector, HITS algorithm will sort the vector's entries in decreasing order and then picks the pages whose corresponding entry values are on the top of the sorted list. These pages are presented as the recommended page with respect to the search query. We can also compute hubs vector in one matrix computation. Note that HITS only computes the given query on an induced graph that is smaller than that of the entire Web.

2.1.3 Relevance to Markov Process: Random Walk

The presentation here follows the notations used in Borodin et. al. [Bordoin 01] Recall the matrix $M = E^t \times E$ whose entry represent the number of path from a random walk doing a forward F (pointing) move and backward B (pointed by) move. We construct a undirected graph G_{BF} whose nodes are pages in the base set. G_{BF} will have an edge from i to j if there is a BF path between them, i.e. $M_{i,j} = 1$. Let $M(i) = \sum_j M_{i,j}$ be all the moves possible from i. If we take a unit vector as the initial authority vector a, then under one iteration, we have

$$a_i = \frac{\|M(i)\|_1}{\|M\|_1}$$

Let the probability of transition from i to j is $P_{i,j} = \frac{\|M_{i,j}\|_1}{\|M(i)\|_1}$. We now show the transition matrix P is a irreducible and aperiodic Markov chain:

- Row is a distribution: $\sum_j P_{i,j} = \sum_j \frac{\|M_{i,j}\|_1}{\|M(i)\|_1} = \frac{\|M(i)\|_1}{\|M(i)\|_1} = 1$
- Irreducible: Irreducible means any page i can reach any other page j through some directed path on the Web. G_{BF} has one connected component.
- Aperiodic: A state is periodic if a random walk only come back to this state at periodic steps. A random walk can traverse the same edge to go back to its origin. So there is a length of 2 closed walk. Since it is from the Web, we can assume that the graph is non-bipartite (it would be strangely surprising if it turns out to be bipartite. In general, this case rarely happens.) and we can have a odd cycle closed walk. But $gcd(2,3) = 1$, which tells us that P is aperiodic.

From the classical result of random walk [Motwani 95], the stationary distribution at i is exactly a_i shown above. We briefly describes the definitions and theorem.

Definition 1 *A Markov chain is said to be irreducible whenever its underlying graph consists of a single strong component.*

Definition 2 *The periodicity of a state i is the maximum integer T for which there exists an initial distribution $q^{(0)}$ and positive integer a such that, for all t, if at time t we have $q_i^{(t)} > 0$, then t belongs to the arithmetic progression $\{a + Ti | i \geq 0\}$. A state is said to be periodic if it has periodicity greater than 1, and is said to be aperiodic otherwise. A Markov chain in which every state is aperiodic is known as an aperiodic Markov chain.*

Theorem 1 *Any irreducible, finite and aperiodic Markov chain will have a unique stationary distribution π such that, for $1 \leq i \leq n, \pi_i > 0$.*

2.2 Variants

2.2.1 Simple Weighted Scheme: Avoiding Clique Attack

From the HITS algorithm, if a set of webpages are densely connected to each other, then their hub and authority values will be propagated quickly. It is conceivable that there can be a malicious attack on this algorithm. Malicious sites owner can use two sites she own, construct two sets of webpages, one for each site, and connect these pages together under the same anchor text. This is known as ‘clique attack’ (a.k.a Tightly Knit Community in [Lempel 00]). To avoid ‘clique attack’ Bharat & Henzinger [Bharat 98] modified the HITS. They put weights on links so if a page points to k other pages on another site, then this page contributes only $1/k$ of its hub value on each link. If a page is pointed by k pages from another site, then each pointing page only contributes $1/k$ of its authority value.

2.2.2 SALSA: Stochastic Approach to Link Structure Analysis

One can think of traversing the Web as a random walk. A person follows one of the links on a page and visit another linked page. Lempel & Moran [Lempel 00] design a special kind of random walk: one step forward and one step backward. In one random move, the user visit back and forth on pages following the links. Note that she may not be following the previously traversed forward link as its backward link. She is wiggling her way on the Web.

Their algorithm follows the outline of the HITS and they consider the hub and authority values for each page. First, they build the base set under a specific query. Then they transform the graph formed by the base set into a new graph where each node in the original graph is associated with two nodes: one for the hub and one for the authority. Let $G(V, E)$ be the base set graph which is a directed graph. Let $G'(V', E')$ be the transformed non-directed graph, where

$V' = V'_h \cup V'_a$
$V'_h = \{v_h | v \in V \text{ and } out - degree(v) > 0\}$ (hub vertices)
$V'_a = \{v_a | v \in V \text{ and } in - degree(v) > 0\}$ (authority vertices)
$E' = \{\{u_h, v_a\} | (u, v) \in E \text{ i.e. } u \to v\}$

There are two random walks. An authority walk starts from the authority side, and a hub walk starts from the hub side. They define two Markov chains for these two walks. Let M_h and M_a be the two Markov chains for the walk respectively. Entry (p, q) of a matrix represents the probability of visiting page q when the current location is page p.

$$H_{p,q} = \sum_{\{r|\{p_h,r_a\},\{q_h,r_a\}\in G'\}} \frac{1}{deg(p_h)} \cdot \frac{1}{deg(r_a)}$$

$$A_{p,q} = \sum_{\{r|\{r_h,p_a\},\{r_h,q_a\}\in G'\}} \frac{1}{deg(p_a)} \cdot \frac{1}{deg(r_h)}$$

Let M be the adjacent matrix of the original directed graph. M_r is the matrix whose entry is M's divided by the sum of the entries on the same row. M_c is the matrix whose entry is divided by the sum of the same column. We can formulate $H = M_r M_c^t$ and $A = M_c^t M_r$. H and A will play the same role as M described earlier in HITS. Lempel & Moran proves that both matrices have stationary distribution and can be computed easily from a page's degrees. Let

$$W = \sum_{i\in A} d_{in}(i) = \sum_{k\in H} d_{out}(k)$$

we will have

$$\pi_i = \frac{d_{in}(i)}{W} \quad \text{for all } i \in A \qquad \pi_k = \frac{d_{out}(k)}{W} \quad \text{for all } k \in H$$

Contrary to the HITS looping, SALSA just needs to find out the indegrees, outdegree of each page and the sum of all indegrees. The hub and authority of each page can be calculated directly using the above formula.

2.2.3 Breadth-First-Search: Path Accumulation Weighting

Kleinberg's algorithm is looking at global information; every loop computation will re-inforce itself. SALSA is looking at local information; the computation has to do with only indegree and sum of indegrees. Borodin et. al. [Bordoin 01] describes a in-between version: BFS. Kleinberg's iteration can be regarded as computing the number of paths leaving page i. BFS consider the number of neighbors of page i. Let $(BF)^n(i)$ denotes the set of pages that can be reached from page i by following $(BF)^n$ path. They use exponentially decreasing weighting scheme to compute the authority:

$$a_i = 2^{n-1}|B(i) + 2^{n-2}|BF(i)| + 2^{n-3}|BFB(i)| + \ldots + |(BF)^n(i)|$$

The algorithm starts from page i and do a bread-first search. A page is collected only the first time it is visited. At each step, BFS will either do a forward or backward move depending on if it is odd or even iteration. The decreasing contribution has intuitive meaning on the distance. The further a neighbor is, the lesser contribution it should have.

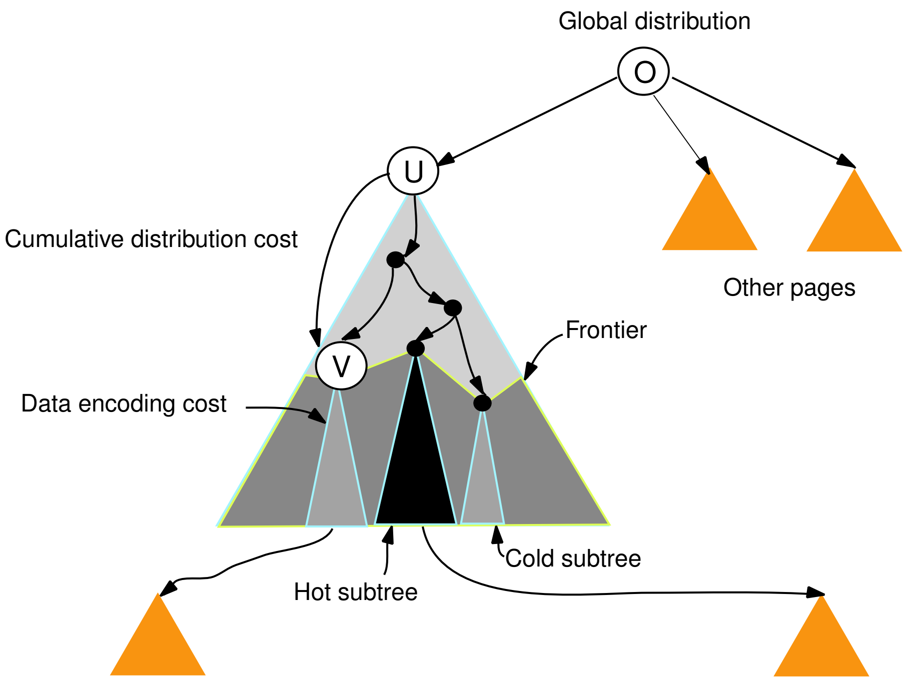

Figure 2.1: Refined Web Graph, encoding cost

2.2.4 Refined HITS: Incorporating DOM Structure

HITS is looking at the page level effect of hub and authority. We certainly can look at microscopic level. Chakrabarti[Chakrabarti 01.1] incorporates the DOM (Document Object Model) level view and looks at microscopic construct of the web. Each HTML page can be parsed into a DOM tree. With this modification, only leaves with hyperlinks leading to other pages are kept; other leaves are thrown away. We can apply HITS on this refined graph (graph includes DOMs). But it is shown that a tree node pointing to both a child within the same DOM tree and to an outside page will cause the unequal diffusion for hub and authority values. So HITS will not work well. Therefore, the paper proposes finding a cut across the DOM tree and a subtree under the cut will be collapsed as one node on the cut. This will prevent the unequal diffusion of hub and authority value. The question is how does one decide where to make the cut.

In this refined graph, we have hub scores at the leaves of a DOM tree and only the root of a DOM tree has authority value. As the choice of hyperlinks differs from ones author to another, the hubs score is affected. Imagine now that there is an universal hub score, a distribution that 'correctly' describe the consensus with respect to a fixed query at the time of request. And there is a super root connecting to all the DOMs. The distribution of nodes walking down from the root is determined by each author. Still, we have to decide how to make the cut across the DOM. If the cut is too low and close to the leaves, it will produce a lot of smaller subtrees, if it is too close to the root, it will contains a diverse scores. To choose the right boundary, we use MDL(Minimum Description Length) principle [Rissanen]. MDL says 'learning' is equivalent to minimizes the sum total of model and data encoding costs. We now describe each cost. Let θ_0 be the global consensus distribution.

- Data encoding cost: Consider an internal node w. Let θ_w be the associated distribution, L_w be the set of HREF leaf nodes in the subtree rooted at w, and the set of hub scores at these leaves is H_w. From Shannon's theorem, there is a efficient codes achieving lower bound of

  $$-\sum_{h \in H_w} \log Pr_{\theta_w}(h) \quad \text{bits}$$

  where $Pr_{\theta_w}(h)$ is the probability of h with respect to the distribution θ_w.

- Model encoding cost: We have θ_0 as the global distribution and we also have θ_w as a local distribution. The change of cost can be specified by Kullback-Leibler distance [Cover 91]

  $$KL(\theta_u; \theta_v) = \sum_x Pr_{\theta_u}(x) \log \frac{Pr_{\theta_u}(x)}{Pr_{\theta_v}(x)}$$

  Suppose we have cost changes $KL(\theta_u; \theta_v)$ and $KL(\theta_v; \theta_w)$ then the cost of the path $u \to v \to w$ is $KL(\theta_u; \theta_v; \theta_w) = KL(\theta_u; \theta_v) + KL(\theta_v; \theta_w)$ As we walk from root down to the node v, we will have $KL(\theta_0, \ldots, \theta_v)$.

Given the parent node θ_u and the observed data H_v, we will choose θ_v to minimize the combined cost:

$$KL(\theta_u; \theta_v) - \sum_{h \in H_v} \log Pr_{\theta_v}(h)$$

Let the cut set be F, we are looking to minimize the value of

$$\sum_{v \in F} \left(KL(\theta_0; \ldots; \theta_v) - \sum_{h \in H_v} \log Pr_{\theta_v}(h) \right)$$

The actual optimization is not practical. We opt for an approximation by moving the cut from the root gradually down to the leaves. The decision of whether the cut will expand on a node to its children depending on comparing the following two values:

- The cost of encoding all data in H_u with respect to the model θ_u.

$$-\sum_{h \in H_u} \log Pr_{\theta_u}(h)$$

- The cost of expanding u to its children and cost of encoding the subtree with respect to its local distribution.

$$\sum_{v \in L_u} \left(KL(\theta_u; \theta_v) - \sum_{h \in H_v} \log Pr_{\theta_v}(h) \right)$$

We take the action on whichever is smaller. When the former is less then u becomes part of the cut. The final problem is how to simulate the global distribution θ_0 and other local distribution? We can choose the exponential distribution. which is said to observe the hub score more closely. The other convenience is that the average of hub value for a subtree leaves can be used as the parameter for the distribution for the root of the subtree. Now we are ready to present the full algorithm:

```
$h = E \times a$

foreach document DOM root $u$
    $F =$ find _cut($u$)

        ($L_v$ : set of leaves rooted at $v$)

        foreach $v \in F$
            $h(v) = \sum_{w \in L_v} h(w)$
            foreach $w \in L_v$
                $h(w) = h(v)$
            end
            $h(v) = 0$
        end
  end

  $a = E^t \times h$
  Normalize $a$   ($\sum_u a(u) = 1$)
```

Chapter 3

PageRank: Linearly Order All Webpages

At the same time when Kleinberg proposed his hubs and authorities approach, Brin & Page presented another method: PageRank[Page 98]. Their idea is simple. A page is linked because it has been judged by a webpage author that it has related content and substance of certain degree and she would like her reader to know. The construction of a link can be seen as a fragment of human intelligence. A page's rank (quality), thus PageRank, is the accumulation of contributing fragments. While HITS also has similar implication, PageRank puts in a division operation rather than just pure addition for accumulation. Note the PageRank has nothing to do with the context, it has to do with the quality of a page, more precisely, the statistical quality of a page. Since each page has its rank value, you can sort any set of pages according to their PageRank. You can linearly order every single page on the Web.

3.1 Basics

3.1.1 From Author's and Reader's View

PageRank method makes the assumption that each page has a quality indicator and this quality is contributed by other pages pointing to it. The amount of contribution from a page is its own PageRank divided equally among all the out-going links.

$$PageRank(i) = \sum_{j \to i} \frac{PageRank(j)}{OutDegree(j)}$$

The contribution of other pages is from an author's point of view. An author links whatever she think is relevant. But we have another important actor in the web: the reader. Imagine a reader who traverses the Web from links to links. She can get tired of it and decide to just 'jump' to another page without following any link on currently visiting page. This makes sense. In another scenario when she is trapped in a community which only points to its own members without any out-going link to the rest of the Web. The formula now becomes:

$$PageRank(i) = (1-d)\sum_{j\to i}\frac{PageRank(j)}{OutDegree(j)} + d\cdot\frac{1}{N}$$

where d, $0 < d < 1$, is the probability a read does a 'jump', N is the total number of pages on the Web. We assume a reader jumps uniformly randomly to any of the page on the Web.

3.1.2 Convergence to the Largest Eigenvector

In fact, PageRank introduces a probabilistic distribution among the pages. To simplify notation, we use $R(i)$ as $PageRank(i)$ and $OD(j)$ as $OutDegree(j)$. Let us sum up all the PageRank over the Web:

$$\sum_i R(i) = \sum_i((1-d)\sum_{j\to i}\frac{R(j)}{OD(j)}) + \sum_i d\cdot\frac{1}{N}$$

Because in-degree links come from out-degrees of other pages, we therefore have the sum of all the in-degrees of all webpages equal the sum of all out-degrees of all webpages. We also have $\sum_i 1/N = 1$. The equation becomes

$$\sum_i R(i) = (1-d)\sum_i\sum_{i\to k}\frac{R(i)}{OD(i)} + d$$

But $\sum_{i\to k} R(i)/OD(i) = R(i)$. So,

$$\sum_i R(i) = (1-d)\sum_i R(i) + d$$

which gives us

$$\sum_i R(i) = 1 \qquad (\|R\|_1 = 1)$$

As the sum of the parts is no less than its part. We have

$$1 \geq R(i) \geq 0$$

Indeed the PageRank produces a distribution among the pages. Now, let us write PageRank down in a matrix format, we have

$$R = (1-d)\cdot M \times R + d\cdot E$$

where R is the PageRank vector, and E is a jump choice vector (in the previous assumption its entry is uniformly equal). Matrix M is constructed such that $M_{i,j}$ is $1/OutDegree(j)$ if there is a link from page j to page i (notice the order); otherwise, it is 0.

$$\begin{array}{rll} & R = M' \times R + E' & [\text{absorbing the constants}] \\ \Rightarrow & R = (M' + E' \times 1^t) \times R & [||R||_1 = 1 \Rightarrow E' \times 1^t \times R = E'] \\ \Rightarrow & R = M'' \times R & \end{array}$$

The last formula shows that R is the eigenvector of eigenvalue 1, and we will show that 1 is the largest eigenvalue. For any vector x, let M_i be M's column vector (or we can choose M''), we have

$$\begin{array}{rll} & Mx = \sum_i x_i M_i & \\ \Rightarrow & \|Mx\|_1 \leq \sum_i |x_i| \|M_i\|_1 & [\|x+y\|_1 \leq \|x\|_1 + \|y\|_1,\ \|cx\|_1 = |c|\|x\|_1] \\ \Rightarrow & \|Mx\|_1 \leq \sum_i |x_i| & [\|M_i\|_1 = 1 \text{ so is } \|M_i''\|_1 = 1\ ] \\ \Rightarrow & \|Mx\|_1 \leq \|x\|_1 & \end{array}$$

But for any eigenvector x and its eigenvalue λ we have $|\lambda|\|x\|_1 = \|Mx\|_1 \leq \|x\|_1$. Thus $|\lambda| \leq 1$ for all λ. So 1 is its largest eigenvalue and R is the corresponding largest eigenvector. The computation for R is a looping process which performs self-correction:

$$\begin{array}{rl} & R = M' \times R + d \cdot E \\ \Rightarrow & R - M' \times R = d \cdot E \end{array}$$

From the formula, we can approximate the R gradually:

```
$R = [\frac{1}{N}]^t$
$R_{new} = [\frac{1}{N}]^t$

loop
    $R = R_{new}$
    $R_{new} = M' \times R$
    $d = \|R\|_1 - \|R_{new}\|_1$
    $R_{new} = R_{new} + d \cdot E$
    $\delta = \|R - R_{new}\|_1$
until $(\delta \leq \epsilon)$
```

Because $\|R\| - \|R_{new}\| \leq \|R - R_{new}\|$ so the computation will not blow up and will converge. In implementation, Google takes out pages with only incoming links or with only outgoing links and computes the PageRank.

3.1.3 Relevance to Markov Process: Random Walk

Another view of the PageRank algorithm is to see how users behave on the Web. They look at the user's browsing behavior as a kind of random walk. From Markov chains convergence theorem [Motwani 95], if a transition matrix is irreducible, aperiodic, and has a stationary distribution, then the Markov computation will converge to the stationary distribution (here, it would be our PageRank vector). We will show that the matrix is irreducible and aperiodic. Let's take a look at the matrix M'' on page 15. Under PageRank's model, the matrix M'' is constructed such that its (i,j) entry is $\frac{(1-d)}{OutDegree(j)} + \frac{d}{N}$ if and only if page j points to i. Otherwise, its value is $\frac{d}{N}$.

- Column is a distribution:

$$\sum_i M''_{i,j} = \sum_{j \to i} \frac{1-d}{OutDegree(j)} + \sum_i \frac{d}{N} = (1-d) + d = 1$$

- Irreducible: Irreducible means any page i can reach any other page j through a directed path on the Web. Since we have for each entry with value at least $\frac{d}{N}$, we have made a 'virtual' link from page j to page i. This link is produced from the random jump. So the virtual graph represented by the M'' is one strong component. It is irreducible.

- Aperiodic: A state is periodic if a random walk only comes back to its starting state at periodic stpes. Let us pick any page i and start a random walk, it is obvious we can traverse through page j (any j) and back to i. Or we can traverse through page j, and k where $j \neq k$, and back to page i. The traversability is enabled because every two pages are linked by our design. Therefore, we have paths of length 2 and 3. But $gcd(2,3) = 1$. So the M'' is aperiodic.

Therefore the Markov computation will converge to the stationary distribution.

3.2 Applications

3.2.1 Rank the Search

PageRank only gives you the quality listing of the page through statistical consensus but it tells you nothing regarding to what the pages are about. We recall that the problem of the returned result of a conventional search engine is its abundance. There are too many pages. PageRank can assist in sorting the list out according to each page's ranking. The intuition is simple: a page will have high reputation on things it talks about. Its content. There is possibility that a link is produced randomly and we have already counted this probability in the 'jump' part. The 'jump' parameter plays a multiple characters. You can think of it as users jump to another page or you can think of it as a 'virtual' link.

Google[Google] uses PageRank. It is vaguely described in the [Brin 98] how Google uses the PageRank in its merged rank which determine a page's final ranking. We can guess if a query string appears in an anchor text pointing to some high PageRank webpage, then this page will be listed higher on the top. The paper also mentioned the relative font size, font color, the proximity of words will play a role. For certain, anchor text and proximity of words play a more important role in the final ranking.

PageRank also defeats the popularity attack. Regardless how many users or agents(program) have traversed a link, PageRank of the destined page is not affected. The defects of counting the 'click-through' as an importance indicator is that the traffic can be easily generated by robot program equipped with IP spoofing technique. PageRank does not suffer from this attack as it does not depend on traffic.

3.2.2 Page Topic Generation

Search engine function can be viewed as from a query to a list of webpages. Rafiei & Mendelzon [Rafiei] asks the reversed question: what about giving a webpage and ask what 'query' should be produced? In other words, what is the topic of the webpage? Imitating the PageRank formula, they reformulate the equation:

$$R^n(i,t) = \begin{cases} (1-d)\sum_{j\to i}\frac{R^{n-1}(j,t)}{OutDegree(j)} + d\cdot\frac{1}{N_t} & \text{if page } i \text{ contains term } t \\ (1-d)\sum_{j\to i}\frac{R^{n-1}(j,t)}{OutDegree(j)} & \text{otherwise} \end{cases}$$

where $R^n(i,t)$ is the term rank on n-th step of page i with the term t. N_t is the number of pages where the term t occurs. The equation is in the same spirit of PageRank and the matrix is constructed similar to that of PageRank replacing N by N_t. They also imitate the HITS algorithm and mixs it with PageRank's flavor to come up with the following:

$$A^n(i,t) = \begin{cases} (1-d)\sum_{j\to i}\frac{H^{n-1}(j,t)}{OutDegree(j)} + d\cdot\frac{1}{N_t} & \text{if page } i \text{ contains term } t \\ (1-d)\sum_{j\to i}\frac{H^{n-1}(j,t)}{OutDegree(j)} & \text{otherwise} \end{cases}$$

$$H^n(i,t) = \begin{cases} (1-d)\sum_{j\to i}\frac{A^{n-1}(j,t)}{InDegree(j)} + d\cdot\frac{1}{N_t} & \text{if page } i \text{ contains term } t \\ (1-d)\sum_{j\to i}\frac{A^{n-1}(j,t)}{InDegree(j)} & \text{otherwise} \end{cases}$$

where $H^n(i,t)$ and $A^n(i,t)$ are term hub value and term authority value of page i on term t. The proofs of convergences follow closely to what we have done in PageRank and we will not repeat them here. For each term t appears in the input page, the above algorithms compute the corresponding values. The problem is it is computationally exhausting if you want to compute for every term for every page on the Web. Their experiment was limited to a smaller induced graph but not the entire Web graph. There is another example in the next chapter to show the advantage of ‘jump’ method in PageRank. It is a method to break away from the ‘clique phenomena’ where a user is ‘trapped’ in a densely connected small community.

Chapter 4

Stability of Algorithms

A natural question to ask about the search engine algorithms is how stable are they. In other words, in the fast changing world of the Web, how stable are the hubs and authorities value? How about PageRank? Ng et. al. [Ng 01.1] asks the question and provides some answers to the stability of both types of algorithms. We start with the HITS type.

4.1 Stability of Eigenvector under Perturbation

The question Ng et. al. asked for HITS is if we change the links by adding or deleting them from one page (only one), what is its effect on the overall authority vector value? They come up with the following result:

a^* : principal eigenvector
$\tilde{a}^*$: perturbed principal eigenvector
d : maximum out-degree in the graph
$\delta = \lambda_1 - \lambda_2$: eigengap
For any ϵ, let $\alpha = \frac{\epsilon\delta}{(4+\sqrt{2}\epsilon)}$, and $k < (\sqrt{d+\alpha} - \sqrt{d})^2$
k: number of links changed on one page

$$\Longrightarrow \quad \|a^* - \tilde{a}^*\|_2 \le \epsilon$$

Given a graph, the freedom will reside on the eigengap δ. This gives us little room to leverage the ϵ. As a matter of fact, they prove that there exists a $O(\delta)$ perturbation to the symmetric matrix M that causes a large $\Omega(1)$ change in the principal eigenvector.

4.2 Stability of Pagerank with Page Links Changes

The stability result for the HITS is limited but the sensitivity of PageRank gives a better hope. Recall that $R = (1-d) \cdot M \times R + d \cdot E$

M: original transition matrix
R: PageRank for M
$i_1, i_2, \ldots, i_n$: changed pages in any way $\Rightarrow$ $\|\tilde{R} - R\|_1 \leq \frac{2}{d}\sum_{j=1}^{j=k} R_{i_j}$
$\tilde{M}$: transition matrix after changes
$\tilde{R}$: PageRank for $\tilde{M}$

As long as the sum of PageRank of the changed pages is not large, the bound is controlled.

4.3 From Properties to Applications

From previous sections, we learn that PageRank seems to be more stable and flexible under perturbation. We can use this property. Note that one difference between HITS and PageRank is PageRank can 'jump'. This gives the PageRank power to escape the 'clique attack'. It prevents users to be trapped in the smaller and yet closely connected community. Adopting the 'jump' concept, we can have

$$a^{(t+1)} = d \cdot \vec{1} + (1-d)A_{row}^T h^{(t)} \qquad h^{(t+1)} = d \cdot \vec{1} + (1-d)A_{col}^T h^{(t+1)}$$

This is called randomized HITS. And it enjoys the same perturbation stability as PageRank.

We notice that while the eigenvectors may change; the subspace spanned by the eigenvectors can be stable. Let us consider the first k eigenvectors $x_1, \ldots, x_k$ with eigenvalues of $\lambda_1, \ldots, \lambda_k$ of $E = A^T A$. Let e_j be the $j-$th basis vector. Authority scores is defined as

$$a_j = \sum_{i=1}^{k} f(\lambda_i)(e_j^T x_i)^2$$

which is the sum of the square of the length of projection of e_j onto the subspace spanned by $x_1, \ldots, x_k$ where x_i is weighted by $f(\lambda_i)$. This is called subspace HITS algorithm. There is a variety of choices for $f(\lambda)$.

- $f(\lambda) = \begin{cases} 1 & \text{when } \lambda \geq \lambda_{max} \\ 0 & \text{else} \end{cases}$ We have HITS in this case.
- $f(\lambda) = \lambda$ It is citation counting.
- $f(\lambda) = 1$ $a_j = \sum_{i=1}^{i=k} x_{ij}^2$

It is proved in their paper that:

f : for all x, y we have $|f(x) - f(y)| \leq L|x - y|$
$\tilde{M} = M + E, \quad \|E\|_F = \epsilon \quad (E \text{ is symmetric})$ $\Rightarrow$ $\|a - \tilde{a}\|_2 \leq L\epsilon$

It is shown that both randomized HITS and subspace HITS (with $f(\lambda) = \lambda^2$) behave more stable than HITS.

Chapter 5

Comparisons and Prospects

While both HITS and PageRank use linkage analysis and get the largest eigenvector for their result. There are operational difference. To summarize the difference between HITS and PageRank:

- Views: HITS algorithm is looking from webpage author's view and PageRank includes both the author's and reader's view.
- Preprocessing: PageRank will preprocess the Web graph to obtain the page ranking while HITS only processes in real time.
- Locality: PageRank will look at the complete graph while HITS only work on a reduced graph.
- Filtering: PageRank will filter out the query related pages and consult the rank to order them while HITS will filter the scaled down result first and then compute the authority and hub to order them.
- Precision: PageRank single outs pages with query terms while HITS tend to have generalization effect and tend to drift to broader topic.

While in practice there is no known search engine implementing the HITS method, its contribution on the notion of hub and authority is important. PageRank, on the other hand, is very practical and has pleasantly good precision in locating users' queries. It is author's belief that in future contextual computing will play an essential role in search engine. And we can expect that search engine will be more 'understanding' to our queries.

Bibliography

[Bharat 98] Krishna Bharat, Monika R. Henzinger. Improved Algorithms for Topic Distillation in a Hyperlinked Environment, SIGIR '98, pp.104-111.

[Bordoin 01] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, Panayiots Tsaparas. Finding Authorities and Hubs From Link Structure on the World Wide Web, WWW10, May 2001, pp.415-429.

[Botafogo 91] Rodrigo A. Botafogo, Ben Shneiderman. Identifying Aggregates in Hypertext Structures, Hypertext '91, pp. 63-74.

[Brin 98] Sergey Brin, Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine, WWW7, 1998.

[Page 98] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web, http://www-db.stanford.edu/ backrub/pageranksub.ps

[Chakrabarti 99.2] Soumen Chakrabarti, Byron E. Dom, David Gibson, Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins. Mining the Link Structure of the World Wide Web, IEEE Computer, 1999.

[Chakrabarti 01.1] Soumen Chakrabarti. Integrating the Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information extraction, WWW10, May 2001, pp. 211-220.

[Cover 91] Thomas M. Cover, Joy A. Thomas. Elements of Information Theory, Wiley Series in Telecommunications, 1991.

[Gloub 89] Gene H. Gloub, Charles F. Van Loan. Matrix Computations, second edition, The Johns Hopkins University Press, 1989.

[Google] Google, http://www.google.com

[Kleinberg 98] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment, SODA, 1998, pp.668-677.

[Lempel 00] Ronny Lempel, Shlomo Moran. The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect, WWW9, 2000.

[Motwani 95] Ranjeev Motwani, Randomized Algorithm, Cambridge University Press, 1995.

[Ng 01.1] Andrew Y. Ng, Alice X. Zheng, Michael I. Jordan. Stable Algorithms for Link Analysis. SIGIR '01, pp. 258-266.

[Ng 01.2] Andrew Y. Ng, Alice X. Zheng, Michael I. Jordan. Link Analysis Eigenvectors and Stability. IJCAI '01.

[Page 98] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web, http://www-db.stanford.edu/ backrub/pageranksub.ps

[Rafiei] Davood Rafiei, Alberto O. Mendelzon. What is this Page Known for? Computing Web Page Reputations. WWW9, 2000, pp. 823-835.

[Rissanen] J. Rissanen. Stochastic Complexity in Statistical Inquiry: in World Scientific Series in Computer Science, Vol 15, World Scientific, Singapore, 1989.

[Salton 75] Gerard Salton. A Theory of indexing, regional Conference Series in Applied Mathematics, SIAM, 1975.

[Salton 89] Gerard Salton. Automatic Text Processing, Addison-Wesley, 1989.