A Survey of Ranking Algorithms

Alessio Signorini
alessio-signorini@uiowa.edu

Department of Computer Science
University of Iowa

September 11, 2005

Abstract

With the huge number of web pages that exist today, search engines assume an important role in the current Internet. But even if they allow to find relevant pages for any search topic, nowadays the number of results returned is often too big to be carefully explored. Moreover, the needs of the users vary, so that what may be interesting for one may be completely irrelevant for another. The role of ranking algorithms is thus crucial: select the pages that are most likely be able to satisfy the user's needs, and bring them in the top positions. In this survey I will cover the most popular algorithms used today by the search engines: PageRank, HITS and SALSA.

1. Introduction

An efficient ranking algorithm is important in any information retrieval system. In a web search engine, due to the dimensions of the current web, and the special needs of the users, its role become critical. Recent studies [1] estimated the existence of more than 11.5 billion pages on the web. Nowadays, it is common for simple search queries to return thousands or even millions of results. Internet users do not have the time and the patience to go trough all them to find the ones they are interested in, and often, as has been shown by some studies [2, 3] they don't even look beyond the first page of results. Therefore, it is crucial for the ranking algorithm to output the desired results within the top few pages, otherwise, the search engine could be considered useless.

Moreover, what the users expect from a web search engine is very different from a traditional information retrieval system. For example, user who looks for "microsoft" on a web search engine is most likely looking for the homepage of the Microsoft Corporation, rather than the page of some random user complaining about a new Microsoft product, even though in a traditional information retrieval sense, the latter page may be highly relevant to the query.

Web users are most interested in pages that are not only *relevant*, but also *authoritative*, that is "trusted sources of correct information that have a strong presence on the web". Thus, in web search the focus shifts from relevance to *authoritativeness*. The task of the ranking function becomes to identify and rank highly the authoritative documents within a collection of web pages.

The web helps this task providing a rich context of information which is expressed by the *hyperlinks*. The hyperlinks define the "context" in which a web page appears: intuitively, a link from page p to page q denotes an endorsement for the quality of page q. In this sense, we can think of the web as a network of recommendations which contains information about the authoritativeness of the pages. The task of the ranking function becomes to extract this informations and produce ranking scores in according to the relative authority of web pages.

Unfortunately, not all links are informative. There are many kinds of link which confer little or no authority to the target pages. Consider, for example the *intradomain links*, whose purpose is to provide navigational aid in a complex web site, or *advertisements/sponsorship links*. Another kind of non-informative links are those which result from *link-exchange agreements*. These are bidirectional links between two web sites, whose only purpose is to increase the visibility and link popularity of their pages. Moreover, it is common to find at the bottom of web pages links to the *web browser* used by the author for testing, to the download page for Acrobat Reader or to some kind of new software plug-in. Even those links "distract" the ranking algorithm, endorsing quality to a page that is often completely unrelated to the presented topic.

2. Background

In this section I present the necessary background for the following sections. First, I give a brief explanation on the *web graph* representation, then I explain the initial input any ranking algorithm, and finally I introduce some mathematical notation that will be used in the rest of the paper.

2.1. The Web Graph

The *web graph* has been intensively studied over the last four years. In [6], it has been shown that the *web graph* has a *bow-tie* shape, as illustrated in Figure 1. The picture show a core made by large strongly connected components (SCC), and four sets of vertices distinguishable from their relation with the core: (1) the *upstream nodes* (IN), than can reach SCC but not be reached from SCC; (2) the *downstream sets* (OUT), that can be reached by SCC but cannot reach it; (3) the *tendrils*, or sets made by the elements that cannot neither reach nor be reached from the core; and (4) the *disconnected components*, a small group of vertices not connected to the bow-tie. In the same paper has been also demonstrated that many features of the *web graph* follow a power law distribution (i.e. a function such as $\frac{1}{k^\alpha}$). For instance, the in-degree and out-degree distributions follow a power law with different values of α.

Ranking algorithms have to be aware of the web structure to improve their performance, since capturing a snapshot of the entire web is not feasible. Millions of web pages are created or disappear every minute, and most of them are never referenced anywhere else. Thus, the ranks of the pages will (and have to) change constantly, while the crawler updates its database with both new and deleted pages.

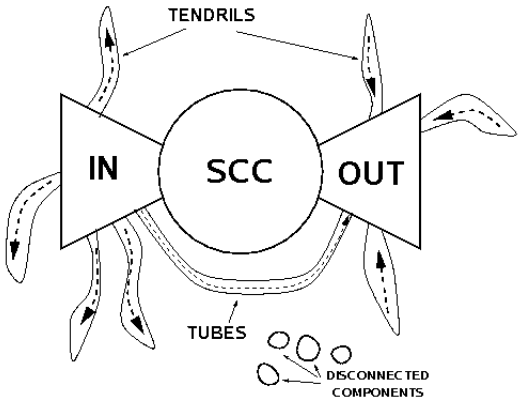

Figure 1: Structure of the *web graph*

2.2. The Initial Set of web Pages

Every ranking algorithm based on link analysis starts with a set of web pages. Dedepending on how this set is obtained, algorithms are classified in two subcategories: *query independent* algorithms, and *query dependent* algorithms. The first category contains the PageRank algorithm by Brin and Page: it starts with a set gene rated by all the web pages captured by the spider and produces ranking scores for all of them, regardless of the query. In contrast, the HITS algorithm by Kleinberg belongs to the second class of algorithms. It uses the user query to obtain a *root set* of pages, and augment it adding pages linked, or that link to, pages of the *root set*.

Given the set of web pages, the next step is to construct the underlying hyperlink graph. A node is created for every web page, and a *directed edge* is placed between two nodes if there is an hyperlink between the corresponding web pages. In some systems, as in the one of Bharat and Henzinger [4], a weight (estimated with content analysis of the web pages) is assigned to each edge. The resulting graph is *simple*: no self-loops are allowed, and even if there are multiple links between two pages only a single edge is placed. In addition, links within the same web site are usually removed, since they serve only for navigation and do not convey any endorsement. After refinement, isolated nodes are removed from the graph.

2.3. Mathematical Notations

Let P denote the resulting set of nodes, and let n be the size of the set P. Let $G = (P, E)$ denote the underlying graph, where each directed edge connects two nodes if a link exists between them. The input of any link analysis algorithm is the adjacency matrix W of the graph G where $W[i, j] = 1$ if there is a link from node i to node j, and zero otherwise. The output of the algorithms is an n-dimensional vector a, where a_i is the authority weight of node i in the graph. These weights are used to rank the pages.

For some node i, we will denote by $B(i) = \{j : W[j, i] = 1\}$ the *backward links set*, that is, the set of nodes that point to node i. Similarly we denote by $F(i) = \{j : W[i, j] = 1\}$ the *forward links set*, that is, the set of nodes pointed to by node i.

Furthermore, we define an *authority node* in the graph G to be a node with non-zero in-degree, and a *hub node* in the graph G to be node with a non-zero out-degree. We will use the letter A to denote the set of authority nodes, and the letter H to denote the set of hub nodes. Consequentially we have that $P = A \cup H$.

We define the *undirected authority graph* $G_a = (A, E_a)$ on the set of authorities A, as the graph where we place an edge between two authorities i and j, if $B(i) \cap B(j) \neq 0$.

3. The Algorithms

In this section I will provide a description of the *Link Analysis Ranking* (LAR) algorithms currently used by most well-known search engines. After a brief introduction of the heuristic InDegree algorithm, father of all the following ranking algorithms, I will describe the PageRank algorithm used by Google and the HITS algorithm used by Teoma. The last algorithm presented is an extention of the HITS algorithm called SALSA.

3.1. In-Degree

A simple heuristic that can be viewed as the predecessor of all link analysis ranking algorithms is to rank the pages according to their *popularity*. The popularity of a page on the web is measured by the number of pages that point to it. We refer to this algorithm as the InDegree algorithm, since it ranks pages according to their in-degree in the graph G. That is, for every node i, $a_i = |B(i)|$. This simple heuristic was applied by several search engines (Altavista, HotBot, ...) in the early days of web Search.

In one [5] of his studies, Kleinberg makes a convincing argument that this algorithm is not sophisticated enough to capture the authoritativeness of a node, even when restricted to a query dependent subset of the web. In addition, if search engines would apply this simple ranking scheme it would be very easy for a web master to influence authoritary: they could simply create thousands of linked pages that point to the "authoritative" page.

3.2. PageRank

The following description of the PageRank algorithm follows closely the description given by Brin and Page in their paper "The PageRank Citation Ranking: Bringing Order to the web".

The intuition underlying the InDegree algorithm is that a good authority is a page pointed to by many other pages in the graph G. Brin and Page [7] extended this idea further by observing that not all links have the same importance. For example, if a web page has a link off the Yahoo! home page, it may be just one link but it is a very important one. This page should be ranked higher than many pages with more links but from obscure places. PageRank is an attempt to see how good an approximation of importance can be obtained just from the link structure.

Web pages vary greatly in terms of the number of backlinks they have. For example, the UI Computer Science Department home page has[1] 269 backlinks, compared to the Netscape page that has 50,900 backlinks (some other web pages have just few links). Generally speaking, highly linked pages are more

[1] At the time of this writing, in according to the Google database

"important" than pages with just few links[2]. The PageRank algorithm provides a more sophisticated method for doing citation counting. The reason that PageRank is interesting is that there are many cases where simple citation counting does not correspond to our common sense notion of importance. Based on the discussion above, is it possible to give the following intuitive description of PageRank: *a page has high rank if the sum of the ranks of its backlinks is high.* This covers both the case when a page has many backlinks and when a page has just a few (but highly ranked) backlinks.

Let u be a web page. Then let F_u be the set of pages u points to and B_u the set of pages that point to u. Let $N_u = |F_u|$ the number of links from u and let c be a factor used for normalization (so that the total rank of all web pages is constant). We can now define a simple ranking R which is a slightly simplified version of the actual PageRank:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_u}$$

This formalizes the intuition in the previous section. The rank of a page is divided among its forward links evenly to contribute to the ranks of the pages they point to. Note that $c < 1$ because there are a number of pages with no forward links and their weight is lost from the system. The presented equation is recursive but it may be computed by starting with any set of ranks and iterating the computation until convergence.

There is a small problem with this simplified ranking function. Consider two web pages that point to each other but to no other page, and suppose there is some web page which points to one of them. Then, during iteration, this loop will accumulate rank but will never distribute any rank (since there are no outedges). The loop forms a sort of trap (rank sink). To overcome this problem of rank sinks, we need to introduce a rank source:

Definition 1 *Let $E(u)$ be some vector over the web pages that corresponds to a source of rank. Then, the PageRank of a set of web pages is an assignment, R', to the web pages which satisfies*

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_u} + cE(u)$$

such that c is maximized and $\|R'\|_1 = 1$ ($\|R'\|_1$ denotes the L_1 norm of R').

where $E(u)$ is some vector over the web pages that corresponds to a source of rank. Note that if E is all positive, c must be reduced to balance the equation. Therefore, this technique corresponds to a decay factor. In matrix notation we have $R' = c(AR' + E)$. Since $\|R'\|_1 = 1$, we can rewrite this as $R' = c(A + E * 1)R'$ where 1 is the vector consisting of all ones. So, R' is an eigenvector of $(A + E * 1)$.

Random Surfer Model The definition of PageRank above has another intuitive basis in random walks on graphs. The simplified version corresponds to the standing probability distribution of a random walk on the graph of the web. Intuitively, this can be thought as modeling the behavior of a "random surfer".

[2]By analogy, simple citation counting has been used in the past to speculate on the future winners of the Nobel Prize!

The "random surfer" simply keeps clicking on successive links at random. However, if a real web surfer ever gets into a small loop of web pages, it is unlikely that the surfer will continue in the loop forever. Instead, the surfer will jump to some other page. The additional factor E can be viewed as a way of modeling this behavior: the surfer periodically "gets bored" and jumps to a random page chosen based on the distribution E. Thus, each step in the PageRank algorithm is of one of two types:

1. From the given state s, choose at random an outgoing link of s, and follow that link to the destination page.
2. Choose a web page uniformly at random, and jump to it.

PageRank chooses a parameter d, $0 < d < 1$, and each state transition is of the first transition type with probability d and of the second type with probability $1 - d$. Since PageRank examines a single random walk on the entire WWW, the ranking of web pages in Google is independent of the search query (a global ranking), and no distinction is made between hubs and authorities.

Dangling Links One issue with this model is created by dangling links. These are links that point to a page with no outgoing links. There are lots of them in the *web graph*, and it is not clear if their weights should be distributed or not. Often these dangling links are simply pages that have not been downloaded yet (it is not easy to sample the entire web!) by the crawling engine.

Because dangling links do not directly affect the ranking of any other page, they can simply be removed from the system until all the ranks are calculated. After that, they can be added back without affecting things significantly. Clearly, removing a link from a page will change the distribution of its weight across the others. However, the removal of the dangling links should not have a big impact on the final ranks.

Computing PageRank The computation of PageRank is fairly straightforward if we ignore the issues of scale. Let S be almost any vector over web pages (for example E) and let A be a square matrix with the rows and column corresponding to web pages. Let $A_{u,v} = 1/N_u$ if there is an edge from u to v and $A_{u,v} = 0$ if not. If we treat R as a vector over web pages, then we have $R = cAR$. So R is an eigenvector of A with eigenvalue c. In fact, we want the dominant eigenvector of A. It may be computed by repeatedly applying A to any nondegenerate start vector. With these basis PageRank may be computed as follows:

$$
\begin{aligned}
R_0 &\longleftarrow S \\
loop: & \\
R_{i+1} &\longleftarrow AR_i \\
d &\longleftarrow \|R_i\|_1 - \|R_{i+1}\|_1 \\
R_{i+1} &\longleftarrow R_{i+1} + dE \\
\delta &\longleftarrow \|R_{i+1} - R_i\|_1 \\
while(\delta > \epsilon) &
\end{aligned}
$$

Note that the d factor increases the rate of convergence and maintains $\|R\|_1$. An alternative normalization is to multiply R by the appropriate factor. The use of d may have a small impact on the influence of E.

Modifications Over the last few years the interest around PageRank has increased progressively. Google established its leadership in the search engine market, and a lot of studies have been made on how to accelerate the calculation of the PageRank. McSherry at Microsoft Research reformulated [15] the power iteration algorithm in such a way that the nodes do not communicate anymore the current probability values to their neighbors, instead, they communicate only changes in the probability value. This reformulation enabled large degree of flexibility in the manner in which nodes update their value, leading to faster convergence, efficient incremental updating, and a robust distributed implementation. Kamvar, Haveliwala and Golub at Stanford University studied [11] some adaptive methods for computation of the PageRank. Some of their papers talk about exploiting the *block structure* of the web and the use of some extrapolation methods to accelerate the computations. Langville and Meyer proposed [12] some reordering of the algorithm steps that should increase the speed in the calculation of the ranks. Other studies, such as the one [13] of Boldi, Santini and Vigna at the University of Milan, studied how the behavior of the PageRank algorithm changes according to the *damping factor* α, giving also a demonstration of how the k-th iteration of the *power method* gives exactly the same value of a MacLauring polynomial of degree k. Finally, a recent study [14] of Pan-Chi Lee, Golub and Zenios at the Stanford University proposed a fast two-stage algorithm for the computation of the PageRank.

3.3. HITS

The following description of the HITS algorithm follows closely the description given by Kleinberg in his paper "Authoritative Sources in a Hyperlinked Environment".

Independent of Brin and Page, Kleinberg [5] proposed an improved notion for the importance of a web page. While PageRank computes the page ranks on the entire *web graph*, the HITS algorithm tries to distinguish between hubs and authorities within a subgraph of *relevant* pages. Given any set of web pages, and a specific query string δ, we could, for example, restrict the analysis to the set Q_δ of all pages containing the query string. Unfortunately this has two significant drawbacks: first, this set may still contain well over a million pages (and thus a considerable computational cost), and second, some or most of the best authorities may not belong to this set since the query string is not contained in their pages. However strange this may sound, it is very common on the current web: lots of corporate's home pages are, for example, made out of graphics only and do not contain any text[3], and not all the car manufacturer companies have the word "car" or "automobile" on their home pages. Ideally, the collection S_δ of pages should have the following properties:

1. S_δ is relatively small
2. S_δ is rich in relevant pages
3. S_δ contains most (or many) of the strongest authorities

[3]Adding OCR capabilities to the web crawlers is possible, but has not yet been done

By keeping S_δ small, we are able to afford the computational cost of applying a non-trivial algorithm, and by ensuring it is rich in relevant pages we facilitate the task of finding good authorities, as they are likely to be heavily referenced within S_δ.

As stated in the introduction, the HITS algorithm starts with a *root set* of pages R, obtained using a text-based search engine. This set is then increased adding the pages pointed to, or that point to, any page in the *root set*. To be sure that the mechanism does not degenerate, a parameter d is introduced: we allow a single page in R_δ to bring in at most d pages pointing to it into S_δ.

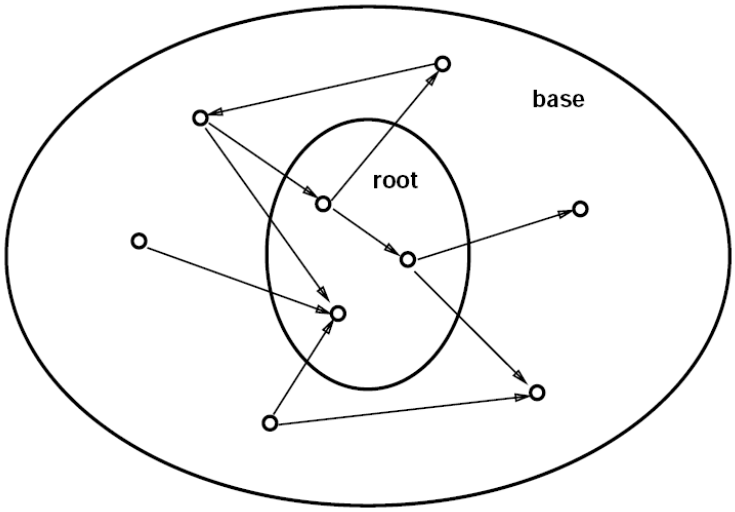

Figure 2: Expansion of the *root set* R

This method provides a small subgraph G_δ that is relatively focused on the query topic, with many relevant pages and strong authorities. The problem is now how to extract those authorities purely trough the analysis of the link structure of G_δ. The simplest approach would be to order pages by their *in-degree*. This method didn't work in the entire *web graph*, as argued in the introduction, but it may work now that we have explicitly constructed a small collection of relevant pages containing most of the authorities that we want to find. These authorities would both belong to G_δ and be heavily referenced by pages within G_δ.

Unfortunately this approach still retains some significant problems. For example, with the query `"Java"` the pages with the largest in-degree consist in `www.gamelan.com` and `java.sun.com`, together with pages advertising for Caribbean vacations and the home page of Amazon Books. This mixture is representative of the type of problems that arises with this simple ranking scheme: while the first two of these pages should certainly be viewed as "good" answers, the others are not relevant to the query topic (they have large in-degree but lack any thematic unity). This demonstrates the difficulty of distinguishing strong authorities from pages that are simply "universally popular". At this point one could wonder whether solving these problems would require making further use of the textual context of the pages, rather than just the link structure of G_δ.

This is not the case. It is in fact possible to extract information more effectively from the links, starting from the following observation:

> **Definition 2** *Authoritative pages relevant to the initial query should not only have large in-degree, but since they are all authorities on a common topic, there should also be considerable overlap in the sets of pages that point to them.*

Thus, in addition to highly authoritative pages we expect to find what could be called *hub pages*: these are pages that have links to multiple relevant authoritative pages. It is these hub pages that "pull together" authorities on a common topic, and allow us to throw out unrelated pages of large in-degree. Hubs and authorities exhibit what could be called a *mutually reinforcing relationship*: a good *hub* is a page that points to many good authorities, while a good *authority* is a page that is pointed to by many good hubs.

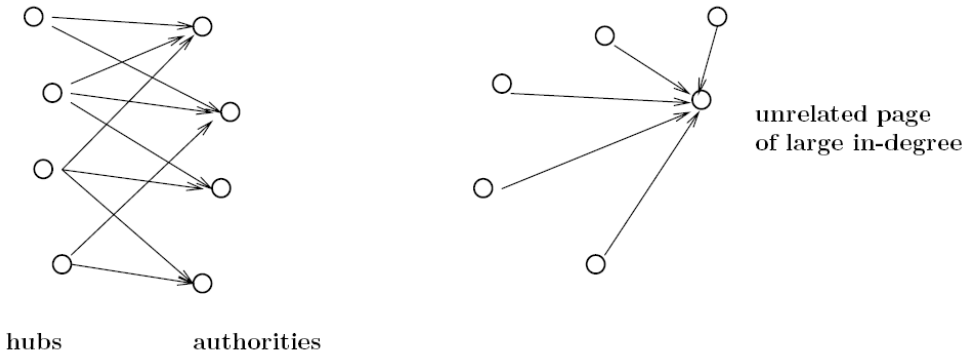

Figure 3: Examples of hubs and authorities

The TKC effect A *Tightly-Knit Community* is a small but highly interconnected set of web pages. Roughly speaking the TKC effect occurs when such a community scores high in link-analyzing algorithms, even though the sites are not authoritative on the topic, or pertain to just one aspect of the topic. Studies [8] of Lempel and Moran indicate that the mutual reinforcement approach is vulnerable to this effect and will sometimes rank the sites of a TKC in unjustifiably high positions. In their work, they provided a combinatorial construction of an infinite number of topologies in which the TKC effect is demonstrated.

For all $k \geq 3$ they build a collection C_k that contains two communities: a community C_s, with a small number of hubs and authorities where every hub points to all of the authorities, and a much larger community C_l, where the hubs points only to a portion of the authorities. The topic covered by C_l appears to be the dominant topic of the collection, and is probably of wider interest on the WWW. Since there are many C_l-authoritative pages, the hubs do not link to all of them, whereas the smaller C_s community is densely interconnected. The TKC effect occurs when the pages of C_s are ranked higher than those of C_l, as happen with the HITS approach. A special case of this problem was already identified by Bharat and Henwinger in 1998.

Computing HITS rank Two weights are assigned to each page p: a non-negative *authority weight* denoted by a_p and a non-negative *hub weight* denoted by h_p. We maintain the invariant that the weights of each type are normalized so their squares sum to 1: $\sum_{p \in S_\delta} (a_p)^2 = 1$ and $\sum_{p \in S_\delta} (h_p)^2 = 1$. The pages with a larger a-value are "better" authorities, as the pages with a larger h-value are "better" hubs.

Numerically, the mutually reinforcing relationship between hubs and authorities can be expressed as follows: (1) if p points to many pages with large a-values, then it should receive a large h-value; if p is pointed to by many pages with large h-values, then it should receive a large a-value. This motivates the definition of two operations on the weights, denoted by I and O. Given weights a_p and h_p, the I operation updates the a-weights as follows

$$a_p \longleftarrow \sum_{q:(q,p) \in \epsilon} h_q$$

similarly the O operation updates the h-weights as follows:

$$h_p \longleftarrow \sum_{q:(p,q) \in \epsilon} a_q$$

Thus, the I and O operations are the basic means by which hubs and authorities reinforce one another. To find the desired "equilibrium" values for the weights, one can apply the I and O operations in an alternating fashion, and see whether a fixed point is reached. The HITS algorithm can be stated as:

Iterate(G,k)
G: a collection of n linked pages
k: a natural number

Let z denote the vector (1, 1, 1, ..., 1) $\in R^n$
Set $a_0 = z$
Set $h_0 = z$
For $i = 1, 2, ..., k$
  Apply the I operation to (a_{i-1}, h_{i-1}), obtaining new a-weights a'_i
  Apply the O operation to (a'_i, h_{i-1}), obtaining new h-weights h'_i
  Normalize a'_i, obtaining a_i
  Normalize h'_i, obtaining h_i
Return (a_k,h_k)

Similar-Page Queries The HITS algorithm, with some modifications, can be adapted to find similar pages, given a page p. Until now the initial *root set* has been created starting with query string δ. To accomplish our new task we have to create the *root set* formulating this new request to the search engine "Find t pages pointing to p". Thus, we will assemble *root set* R_p consisting of t pages that point to p; we will then grow this into a base set S_p as before. The result will be a subgraph G_p in which we can search for hubs and authorities using the known algorithm. This approach is very powerfull: compiling a list of *similar pages* using only text-based methods would be very hard since most of the home web pages today consist almost entirely of images with very little text. In addition, often the text that they contain has very little overlap.

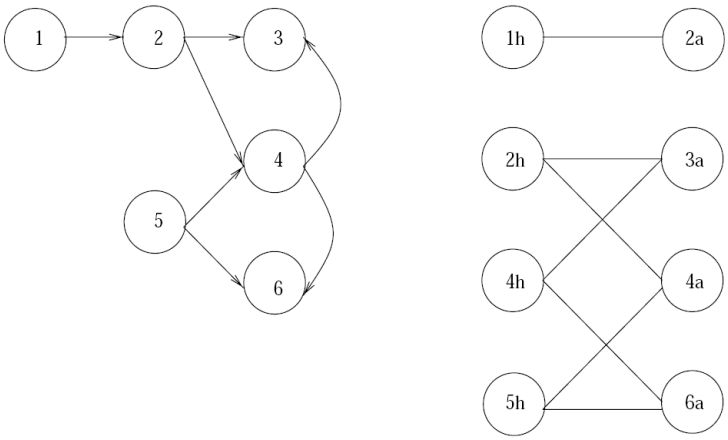

Figure 4: Transforming the set of nodes into a bipartite graph

3.4. SALSA

The following description of the SALSA algorithm follows closely the description given by Lempel and Moran in their paper "SALSA: The Stochastic Approach for Link-Structure Analysis".

An alternative algorithm, that combines ideas from both PageRank and HITS, was proposed [8] by Lempel and Moran. The SALSA algorithm performs a random walk on the bipartite hubs-and-authorities graph, *alternating* between the hubs and authority sides. The random walk starts from some authority node selected uniformly at random and then proceeds by alternating between backwards and forward steps. When at a node on the authority side of the bipartite graph, the algorithm select one of the incoming links uniformly at random and moves to a hub node on the hub side. When at node on the hub side the algorithm selects one of the outgoing links uniformly at random and moves to an authority node.

The authority weights are defined to be the stationary distribution of this random walk. Formally, the Markov Chain of the random walk has transition probabilities

$$P_a(i,j) = \sum_{k:k\in B(i)\cap B(j)} \frac{1}{|B(i)|}\frac{1}{|F(k)|}$$

Recall that $G_a = (A, E_a)$ denotes the authority graph, where there is an (undirected) edge between two authorities if they share an hub. This Markov Chain corresponds to a random walk on the authority graph G_a, where we move from authority i to authority j with probability $P_a(i,j)$. Let W_r denote the matrix derived from matrix W by normalizing the entries such that, for each column, the sum of the entries is 1. Then the stationary distribution of the SALSA algorithm is the principal left eigenvector of the matrix $M_s = W_c^T W_r$.

If the underlying authority graph G_a consists of more than one component, then the SALSA algorithm selects a starting point uniformly at random and performs a random walk within the connected component that contains that node. Let j be a component that contains node i, let A_j denote the set of authorities in the component j, and E_j the set of links in the component j.

Then the weight of authority i in component j is

$$a_i = \frac{|A_j|}{|A|}\frac{|B(i)|}{|E_j|}$$

If the graph G_a consist of a single component (we refer to such graphs as authority connected graphs), that is, the underlying Markov Chain is *irreducible*, then the algorithm reduces to the InDegree algorithm. Furthermore, even when the graph G_a is not connected, if the starting point of the random walk is selected with probability proportional to the "popularity" (in-degree) of the node in the graph G, then the algorithm again reduces to the InDegree algorithm.

The SALSA algorithm can be thought as a variation of the HITS algorithm. In the I operation of the HITS algorithm the hubs *broadcast* their weights to the authorities, and the authorities sum up the weight of the hubs that point to them. The SALSA algorithm modifies the I operation so that instead of broadcasting, each hub *divides* its weight equally among the authorities to which it points. Similarly, the algorithm modifies the O operation so that each authority divides its weight equally among the hubs that point to it. Therefore,

$$a_i = \sum_{j:j\in B(i)} \frac{1}{|F(j)|} h_j \quad and \quad h_i = \sum_{j:j\in F(i)} \frac{1}{|B(j)|} a_j$$

However the SALSA algorithm does not really have the same "mutually reinforcing structure" that Kleinberg's algorithm does. Indeed, $a_i = \frac{|A_j|}{|A|}\frac{|B(i)|}{|E_j|}$, the relative authority of site i *within a connected component* is determined from local links, not from the structure of the component.

3.5. Further extensions

The seminal works of Kleinberg [5], and Brin and Page [7] were followed by a number of extensions and modifications. In one of their works Bharat and Henzinger considered an improvement on the HITS algorithm, called *topic distillation*, that used textual information to weight the importance of nodes and links. Rafiei and Mendelzon presented a variant of the HITS algorithm that uses random jumps, similar to SALSA. Jordan, Ng and Zheng proposed a similar randomized version of HITS called *randomized HITS* and an extension of it that uses multiple eigenvectors. Tomlin generalized the PageRank algorithm to compute flow values for the edges of the web graph that allows to compute the TrafficRank for each page.

A different line of research exploits the application of probabilistic and statistical techniques for computing rankings. The *pHITS* algorithm assumes a probabilistic model in which a link is caused by latent "factors" or "topics". It uses the Expectation Maximization (EM) algorithm to compute the authority weights of the pages. Following that work Hofmann proposed a similar algorithm which also takes into account the text of the documents.

Another interesting line of research aims to combine PageRank with temporal information [9]. On the web, the temporal information for outgoing links is under the control of source pages and is, therefore, susceptible to bias. On the other hand, the incoming links reflect the attention that a web page has attracted and being more democratic in their nature, they are less susceptible to manual influence. Among those incoming links, the link emanating from the random surfer's current position can be picked out and treated in a special way. These observations suggest that the probability of the random surfer choosing y

when leaving their current page x is a combination of many factors: the freshness $f(y)$ of the target page y, the freshness $f(x, y)$ of the link from x to y, and the average freshness of all incoming links of y. In a similar way, the random jump probability of a target page y is a (weighted) combination of the freshness of y, the activity of y, the average freshness of the incoming links of y, and the average activity of the pages that link to y.

In addition, there are many modification of the PageRank which consider graphs with different levels of granularity (HostRank, PageRank on host instead of web pages), or with different link weight assignments (internal, external, etc.). Another interesting line of research tried to avoid the use of an arbitrary chosen teleportation factor (α) in the PageRank algorithm. In particular, Boldi and Vigna in one of their papers provided a closed formula for computing the PageRank using a *Maclaurin polynomial* of degree k. Using that formula the PageRank of a page can be seen as a rational function of α and can be approximated quite efficiently. This fact is used in the paper to define a new form of ranking, the TotalRank, that averages PageRanks over all possible α-s (a form of ranking without damping).

4. Comparisons

The following results are taken by a recent paper [10] on link Analysis Algorithms (LAR) written by Borodin, Roberts, Rosenthal and Tsaparas, at the beginning of this year. In their study, the authors compared the known algorithms introduced in Section 2 with some new algorithms described in their paper.

They studied the rankings produced by the algorithms, and how they relate to each other. In addition, they tried assess the quality of the algorithms and to analyze how theoretically predicted properties show in practical settings.

4.1. The queries

The ranking algorithms have been tested using the following 34 queries:

> abortion, affirmative action, alcohol, amusement parks, architecture, armstrong, automobile industries, basketball, blues, cheese, classical guitar, complexity, computational complexity, computational geometry, death penalty, genetic, geometry, globalization, gun control, iraq war, jaguar, jordan, moon landing, movies, national parks, net censorship, randomized algorithms, recipes, roswell, search engines, shakespeare, table tennis, weather, vintage cars

Many of those queries have already appeared in previous works, and have been used for sake of comparison. The remaining queries have been chosen by the authors because they corresponds to topics for which there are opposing communities, such as "death penalty" and "gun control", or because they are of interest to different communities depending on the interpretation of the word (e.g., "jordan" or "complexity"). The objective is to observe how different algorithms represent these different (and usually unrelated) communities in the top positions of the ranking. Is also interesting to observe the behavior of the algorithms when the query is shifted from a broad topic as "geometry" or "complexity" to a more specific subset of

the topic as "computational geometry" or "computational complexity". Some of the queries, such as "automobile industries" or "search engines", have been selected to study the behavior of the algorithms when the most relevant results do not contain the query words.

4.2. Base Set construction

For the construction of the *base set* the method described by Kleinberg has been followed. The starting *root set* was obtained querying Google and downloading the first 200 pages returned by the search engine. The set is then augmented with pages that point to, or are pointed to by, a page in the *root set*. To discover the web pages that point to a page already in the *root set* the *link:* feature of Google has been used, and the first 50 results have been added to the *base set*. Finally, the links between the pages in the *base set* have been extracted to construct the hyperlink graph.

Then, using a heuristic function of their own design, they tried to remove from the obtained graph the *navigational links*. For each address they compared the first 3 bytes of the IP address. If they were the same, the link was labeled as navigational and discarded. If they differed, they looked at the actual URLs. For each URLs in the form "$http : //string_1/string_2/...$" the domain identifier was chosen to be $string_1$. The domain is of the form "$x_1.x_2.\cdots.x_k$". If $k \geq 3$ they used $x_2.\cdots.x_{k-1}$ as the domain identifier, otherwise, they used only x_1. If the domain identifiers were the same for both the source and target pages of the link, it was labeled as navigational and removed. After all the navigational links were removed, they also eliminated any isolated pages from the set to produced the final *base set* P and the graph $G = (P, E)$.

4.3. Measures

The measure that they used to evaluate quality of the rankings is *precision over top-10*. This is the fraction of documents in the top 10 positions of the ranking that are relevant to the query. This approach is similar to the one used in the TREC conferences for evaluating web search algorithms.

Presenting their results they also used a more refined notion of *relevance*. Given a query, they classified a document as non-relevant, relevant, or highly relevant to the topic of the query. High relevance is meant to capture the notion of authoritativeness. An highly relevant document is one that you would definitely want to be in the few first page of results of a search engine. For example, for the query "movies", the web page `http://abeautifulmind.com/`, the official site for the movie "A Beautiful Mind", is relevant to the topic of movies but it cannot be considered highly relevant. Instead, the page `http://www.imdb.com/`, the Internet Movie Data Base (IMDB) site that contains movie informations and reviews, is a page that is highly relevant to the topic. This is a result that a web user would most likely want to retrieve when posing that particular query. For each algorithm the authors of the paper estimated the *high relevance ration*, the fraction over the top-10 results that are highly relevant.

The study also examined of how the algorithms relate to each other. For the comparison of the rankings provided by two different algorithms they used the *geometric distance measure* d_1, calculated as:

Geometric Distance Measure *The LAR vectors can be viewed as points in a n-dimensional space, thus is possible to use common geometric measures of distance. The distance considered is the Manhattan distance, that is, the L_1 distance of two vectors. Let a_1, a_2 be the two LAR vectors, we define the d_1 distance measure between a_1 and a_2 as*

$$d_1(a_1, a_2) = min \sum_{i=1}^{n} |\gamma_1 a_1(i) - \gamma_2 a_2(i)| \qquad \forall \gamma_1, \gamma_2 \geq 1$$

where the constants γ_1 and γ_2 are meant to allow for an arbitrary scaling of the two vectors, thus eliminating large distances that are caused solely due to normalization factors.

They also introduced another comparison measure called *strict rank distance* $d_r^{(1)}$, calculated as follows:

Strict Rank Distance *We define the Kendall's tau $K^{(1)}$ as the number of bubble sort swaps that are necessary to convert one permutation to another. The maximum value of Kendall's tau is then $n(n-1)/2$, and it occurs when one ranking is the reverse of the other. With this tool, we define the string rank distance $d_r^{(1)}$ as follows*

$$d_r^{(1)}(a_1, a_2) = \frac{1}{n(n-1)/2} K^{(1)}(a_1, a_2)$$

In the paper they considered two additional comparison parameters, intended to more closely reflect a user's point of view: the *intersection over top-k*, denoted by $I(k)$, that is the number of documents that the two rankings have in common in the top k results; and the *weighted intersection over top-k*, denote by $WI(k)$, that is the averaged intersection over the top k results, where the average is taken over the intersection over the top-1, top-2, up to top-k.

4.4. User study

In order to assess the relevance of the documents, the authors performed an on-line user study. The introductory page contained the queries with links to the results, together with some instructions. By clicking on a query, the union of the top-10 results of all algorithms was presented to the user in a permuted and anonymous order. The users were then asked to rate each document as "Highly relevant", "Relevant" or "Non-Relevant".

The users'feedback have been used in the following way: given the users'feedback for a specified document, the document has been rated "Relevant" if the "Relevant" and "Highly Relevant" votes were more than the "Non-Relevant" votes (ties were resolved in favor of "Non-Relevant"). Among the documents that are deemed as "Relevant", they rated "Highly Relevant" the ones for which the "Highly Relevant" votes were more than the "Relevant" ones.

4.5. Results

The *average d_1 distances* table allow us to compare the final rankings returned by the algorithms. A value close to 0 means that the two algorithms produce very close ranking values, while high values indicate a very different ranking attribution.

Figure 5 clearly show how the SALSA and InDegree algorithm provide practically identical results, while HITS and PageRank return very different rankings.

	HITS	PageRank	InDegree	SALSA
HITS	-	1.64	1.22	1.25
PageRank	1.64	-	0.94	0.93
InDegree	1.22	0.94	-	0.11
SALSA	1.25	0.93	0.11	-

Figure 5: Average d_1 distance

While interesting from a technical point of view, the average d_1 distance table does not provide a good estimate of the difference between algorithms: for the end user, the ranking values of the returned pages are not really important, what matters is their order. The *average d_r distance* compares the actual order in which the results are returned, allowing us to make better "end-user comparisons".

	HITS	PageRank	InDegree	SALSA
HITS	-	0.53	0.42	0.45
PageRank	0.53	-	0.32	0.30
InDegree	0.42	0.32	-	0.08
SALSA	0.45	0.30	0.08	-

Figure 6: Average d_r distance

As before, we see how the SALSA and InDegree algorithms provide very similar results. On the other hand, HITS and PageRank still provide orders that differ for more than the 50%.

The last two comparison parameters, the *average intersection over top-k* and *weighted intersection over top-k*, give us an idea of the actual overlap that exists between the algorithms in a typical first page of results:

	HITS	PageRank	InDegree	SALSA
HITS	-	1.1	4.1	4.1
PageRank	1.1	-	3.2	3.1
InDegree	4.1	3.2	-	9.8
SALSA	4.1	3.1	9.8	-

Figure 7: Average $I(10)$ measure

	HITS	PageRank	InDegree	SALSA
HITS	-	1.0	3.7	3.6
PageRank	1.0	-	2.8	2.7
InDegree	3.7	2.8	-	9.7
SALSA	3.6	2.7	9.7	-

Figure 8: Average $WI(10)$ measure

Once more, we see how the InDegree and the SALSA algorithms provide basically the same results, while HITS and PageRank share, on average, just one result over their top 10.

To understand which algorithm works better from an "end-user" point of view, we need to know how many relevant pages are returned by the algorithm in the top-10 results. Table 9 shows the performance of the four analyzed search engines.

	Relevance Ratio					*High Relevance Ratio*			
query	**HITS**	**PageRank**	**InDegree**	**Salsa**		**HITS**	**PageRank**	**InDegree**	**Salsa**
abortion	90%	70%	100%	100%		30%	10%	40%	40%
affirmative action	70%	50%	50%	50%		30%	0%	40%	40%
alcohol	90%	60%	90%	90%		60%	30%	60%	60%
amusement parks	100%	30%	30%	50%		50%	10%	30%	40%
architecture	10%	70%	70%	70%		0%	30%	70%	70%
armstrong	20%	50%	20%	20%		0%	0%	0%	0%
automobile industries	10%	10%	20%	30%		0%	10%	10%	20%
basketball	0%	70%	20%	20%		0%	60%	20%	20%
blues	60%	80%	60%	60%		60%	40%	40%	40%
cheese	0%	20%	30%	30%		0%	0%	0%	0%
classical guitar	90%	50%	70%	70%		40%	30%	30%	30%
complexity	0%	50%	50%	50%		0%	30%	20%	20%
computational complexity	90%	70%	90%	90%		30%	30%	30%	30%
computational geometry	100%	40%	70%	70%		40%	20%	40%	40%
death penalty	100%	70%	90%	90%		70%	30%	70%	70%
genetic	100%	70%	100%	100%		80%	20%	70%	70%
geometry	90%	20%	90%	90%		60%	10%	50%	50%
globalization	100%	70%	90%	90%		0%	30%	20%	20%
gun control	0%	50%	100%	100%		0%	50%	70%	70%
iraq war	40%	30%	30%	30%		0%	10%	10%	10%
jaguar	0%	30%	0%	0%		0%	20%	0%	0%
jordan	0%	30%	30%	30%		0%	10%	20%	20%
moon landing	0%	30%	20%	20%		0%	20%	10%	10%
movies	10%	20%	50%	40%		10%	10%	30%	30%
national parks	0%	50%	10%	10%		0%	50%	10%	10%
net censorship	0%	30%	80%	80%		0%	20%	80%	80%
randomized algorithms	70%	80%	80%	80%		0%	40%	10%	10%
recipes	0%	20%	70%	70%		0%	10%	60%	60%
roswell	0%	20%	40%	40%		0%	0%	0%	0%
search engines	80%	90%	100%	100%		60%	70%	100%	100%
shakespeare	100%	70%	100%	100%		0%	20%	50%	50%
table tennis	90%	60%	100%	100%		50%	20%	50%	50%
weather	80%	50%	80%	80%		60%	20%	60%	60%
vintage cars	20%	10%	60%	60%		0%	0%	40%	40%
average	47%	48%	61%	62%		21%	22%	36%	37%
max	100%	90%	100%	100%		80%	70%	100%	100%
min	0%	10%	0%	0%		0%	0%	0%	0%
standard deviation	43%	23%	31%	31%		27%	17%	26%	26%

Figure 9: Relevance and High-Relevance Ratios

As we can see from the averages given, the SALSA algorithm is the best in finding relevant and highly-relevant pages. Unexpectedly, the very simple InDegree algorithm also scores better than the more complicated HITS and PageRank algorithms which return only 50% of relevant results among their top-10.

In their paper, the authors present some new link analysis ranking algorithms. One of them, combines together the simple idea of the InDegree algorithm with the intuition of the HITS algorithm to create the *Breadth First Search* (BFS) algorithm. In their tests, this new ranking method is shown to outperform the others with an average relevance ratio of 78% and an average high relevance ratio of 44%.

5. Papers

This survey is based on the following papers:

> *The PageRank Citation Ranking: Bringing Order to the web* by L.Page and S.Brin, January 1998
> `http://dbpubs.stanford.edu:8090/pub/1999-66`
>
> *Authoritative Sources in a Hyperlinked Environment* by J.M.Kleinberg, May 1999
> `http://www.cs.cornell.edu/home/kleinber/auth.pdf`
>
> *SALSA: The Stochastic Approach for Link-Structure Analysis* by R.Lempel and S.Moran, April 2001
> `http://www.cs.technion.ac.il/~moran/r/PS/lm-feb01.ps`

References

[1] A.Gulli, A.Signorini, *The Indexable web is More than 11.5 Billion pages*, Proceedings of the 14th World Wide web Conference, 2005

[2] A.Broder, *Web Searching Technology Overview*, Advanced school and Workshop on Models and Algorithms for the World Wide web, 2002

[3] B.J.Jansen, A.Spink, J.Bateman, T.Saracevic, *Real life Information Retrieval: A study of user queries on the web*, ACM SIGIR Forum, 1998

[4] K.Bharat, M.R.Henzinger, *Improved algorithms for topic distillation in a hyperlinked environment*, Research and development in Information Retrieval, 1998

[5] J.Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of ACM (JASM), 1999

[6] A.Broder, R.Kumar, F.Maghoul, P.Raghavan, S.Rajagopalan, R.Stata, A.Tomkins, J.Wiener, *Graph structure of the web*, Proceedings of the Computer Networks and ISDN Systems, 2000

[7] S.Brin, L.Page, *The anatomy of a large-scale hypertextual web search engine*, Proceedings of the 7th International World Wide web Conference, 1998

[8] R.Lempel, S.Moran, *The stochastic approach for link-structure analysys (SALSA) and the TKC effect*, Proceedings of the 9th International World Wide web Conference, 2000

[9] K.Berberich, M.Vazirgiannis, G.Weikum, *T-rank: Time-aware authority ranking*, Proceedings of 3rd International Workshop on Algorithms and Models for the web-Graph, 2004

[10] A.Borodin, G.O.Roberts, J.S.Rosenthal, P.Tsaparas, *Link analysis ranking: algorithms, theory, and experiments*, ACM Transactions on Internet Technology, 2005

[11] S.Kamvar, T.Haveliwala, G.Golub, *Adaptive methods for the computation of PageRank*, Technical report of Stanford University, 2003

[12] A.N.Langville, C.D.Meyer, *A Reordering for the PageRank problem*, Submitted to SIAM Journal on Scientific Computing

[13] P.Boldi, M.Santini, S.Vigna, *PageRank as a Function of the Damping Factor*, Proceedings of the 14th World Wide Web Conference, 2005

[14] C.P.Lee, G.H.Golub, S.A.Zenios, *A fast two-stage algorithm for computing PageRank*, Technical report of Stanford University, 2003

[15] F.McSherry, *A Uniform Approach to Accelerate PageRank Computation*, Proceedings of the 14th World Wide Web Conference, 2005