International Journal of Recent Engineering Research and Development (IJRERD)
Volume No. 02 – Issue No. 02, ISSN: 2455-8761
www.ijrerd.com, PP. 25-36

Search Engines

Yogesh Deoram Desale
Department of Information Technology, Vishwakarma Institute of Management, Pune

Abstract: The world produces up to 2 exabytes of data every year – that's 2,00,00,00,000 gigabytes. It's taken 3, 00,000 years for humans to accumulate 12 exabytes of information. It will take just 2.5 years more to create the next 12 exabytes. How do we make use and sense of all this data? How do we archive and track it? How much of it is useful? These are the pressing questions, now more than ever before.

The World Wide Web is a huge collection of web pages which is growing continuously. The task of searching them efficiently becomes a major challenge. The algorithms to achieve this should retrieve the most relevant matches in order for a given query and use the storage space efficiently.

The World Wide Web creates many new challenges for information retrieval. It is very large and heterogeneous. Current estimates are that there are over 150 million web pages, growing exponentially every year. This explosion in the volume information coupled with the addition of new users, inexperienced in the art of web search, makes the task of designing a web search algorithm challenging. No search engine indexes more than about 16% of the web. Even this amount is much larger than what an individual user can process. So the issue is not the volume of information that can be returned, but the relevance with regards to the initial query.

A good search engine should return the results in the decreasing order of their relevance to the input query i.e. the more relevant results should be returned before the lesser relevant results. First we will take a brief look into the conventional search engines, which use the text based techniques to rank the search results. Then we will look into a couple of next generation ranking algorithms that are completely link based i.e. they view the web as a directed graph and rank the web pages based on their connection with the other web pages.

In this report, we will first discuss the conventional searching tools and describe the problems faced by them and then we move on to the discussion of the architecture of a crawler based search engine. Next we will learn about web crawling technique, discussing about meta-tags and robot-exclusion protocol in particular and then we step to web indexing technique in detail. Then we will look into web searching techniques such as Salsa algorithm and Kleinberg algorithm. Lastly we consider the result ranking techniques with emphasis on page-ranking algorithm and HITS along with few variations of HITS to effectively see how these algorithms overcome the problems faced by the conventional search engines.

Keywords: Crawling, HITS, Indexing, Information Retrieval, PageRank, Pigeon Search, Search Engines, Spider, Web Searching Algorithm.

I. INTRODUCTION

The good news about the Internet and its most visible component, the World Wide Web, is that there are hundreds of millions of pages available, waiting to present information on an amazing variety of topics. Though it is aspect to cheer, the bad news about the Internet is that there are hundreds of millions of pages available, most of them titled according to the whim of their author, almost all of them sitting on servers with cryptic names. When you need to know about a particular subject, how do you know which pages to read? If you're like most people, you visit an Internet search engine.

Internet search engines are special sites on the Web that are designed to help people find information stored on other sites. A search engine is a database of resources extracted from the Internet through an automated "crawling" process and is searchable through user queries. There are differences in the ways various search engines work, but they all perform three basic tasks:

- They search the Internet -- or select pieces of the Internet -- based on important words.
- They keep an index of the words they find, and where they find them.
- They allow users to look for words or combinations of words found in that index.

Early search engines like "gopher" and "Archie" held an index of a few hundred thousand pages and documents stored on servers connected to the Internet and received maybe one or two thousand inquiries each day, but still they did their jobs. Today, a top search engine will index hundreds of millions of pages, and respond to tens of millions of queries per day in order to dramatically reduce the amount of time required to find programs and documents. In this article, we'll tell you how these major tasks are performed, and how Internet search engines put the pieces together in order to let you find the information you need on the Web.

Table1.1: Searches per day (top 5 search engines)

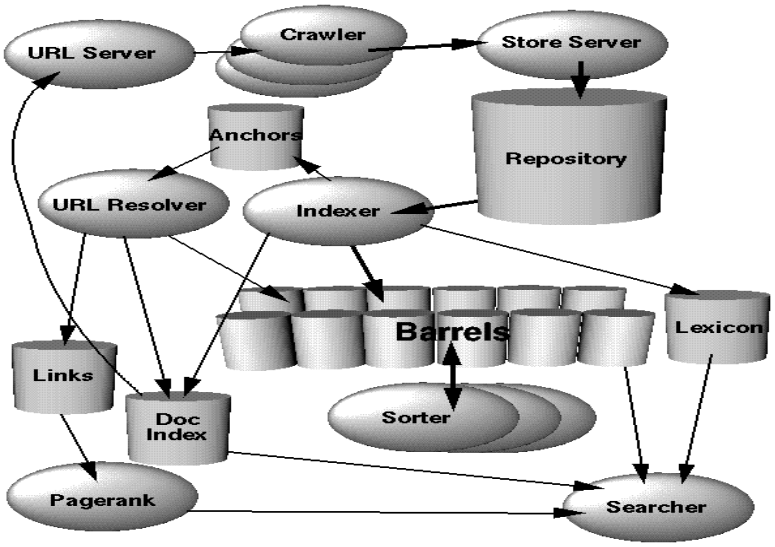
Top 5 Search Engines	Searches per day
Google	250 million
Overture	167 million
Inktomi	80 million
Look Smart	45 million
Find What	33 million

Unless Search engine are well equipped to identify and categorize the web documents it is going to be a great deal of difficulty in retrieving data from web. The WWW is based on the principle of hypertext, a non-sequential method of viewing information. It is easy to find information on the web you just need to know what you're doing, knowing how to use the web intelligently means knowing how to locate useful information on the web. The data present is huge but then most of it is unstructured, i.e. it is in free-text form, unlike structured data, which is readily machine-readable. Consider for example, a filled-in electronic form. The unstructured nature of most data means that retrieving information is a non-trivial business. There is more data out there than we can easily make sense of, which is why there is a lot being done along this direction.

II. ARCHITECTURE OF CRAWLER BASED SEARCH ENGINE

A search engine's architecture should be optimized so that a large document collection can be crawled, indexed, and searched with little cost. Although, CPUs and bulk input output rates have improved dramatically over the years, a disk seek still requires about 10 ms to complete. Thus a search engine has to be designed to avoid disk seeks whenever possible, as this factor has had a considerable influence on the design of the architecture. A general crawler-based search engine has following architecture.

Figure1: High Level Architecture of a Crawler-Based Search Engine



- **Crawler:** The web crawling (downloading of web pages) is done by several distributed crawlers. URL-server: The URL-server sends a list of URLs to be fetched by the crawlers.

- **Store-server:** The web pages that are fetched by the crawlers are sent to the store-server.

- **Repository:** The repository contains the full HTML of every web page. The web pages in the store-server are compressed and stored into the repository. Every web page has an associated ID number called a doc-ID which is assigned whenever a new URL is parsed out of a web page.

- **Indexer:** The indexing function is performed by the indexer. The indexer reads the repository, decompresses the documents, and parses them. Each document is converted into a set of word occurrences called hits. The hits record the word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index. The indexer performs

another important function. It parses out all the links in every web page and stores important information about them in an anchors file.

- **Anchor File:** It contains enough information to determine where each link points from and to, and the text of the link.

- **URL-resolver:** The URL-resolver reads the anchors file and converts relative URLs into absolute URLs and in turn into doc-IDs. It puts the anchor text into the forward index, associated with the doc-ID that the anchor points to. It also generates a database of links which are pairs of doc-IDs.

- **Links Database:** The links database is used to compute PageRanks for all the documents.

- **Document Index:** The document index keeps information about each document. It is a fixed width ISAM (Index sequential access mode) index, ordered by doc-ID. The information stored in each entry includes the current document status, a pointer into the repository, a document checksum, and various statistics. If the document has been crawled, it also contains a pointer into a variable width file called docinfo which contains its URL and title.

- **Lexicon:** The lexicon has several different forms. One important change from earlier systems is that the lexicon can fit in memory for a reasonable price. In the current implementation we can keep the lexicon in memory on a machine with 256 MB of main memory. The current lexicon contains 14 million words (though some rare words were not added to the lexicon). It is implemented in two parts -- a list of the words (concatenated together but separated by nulls) and a hash table of pointers. For various functions, the list of words has some auxiliary information which is beyond the scope of this paper to explain fully.

- **Sorter:** The sorter takes the barrels, which are sorted by doc-ID and resorts them by word-ID to generate the inverted index. This is done in place so that little temporary space is needed for this operation. The sorter also produces a list of word-IDs and offsets into the inverted index. A program called Dump-Lexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher.

- **Searcher:** The searcher is run by a web server and uses the lexicon built by Dump-Lexicon together with the inverted index and the Page Ranks to answer queries.

III. WEB CRAWLING

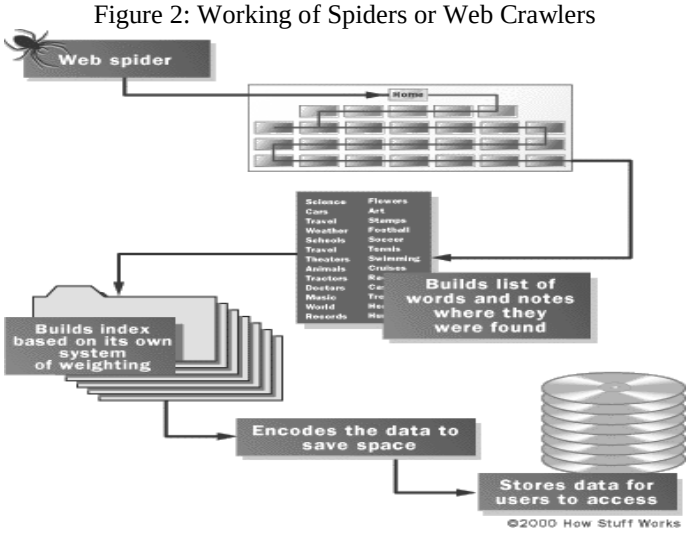
Before a search engine can tell you where a file or document is, it must be found. To find information on the hundreds of millions of Web pages that exist, a search engine employs special software robots, called spiders, to build lists of the words found on Web sites. When a spider is building its lists, the process is called Web crawling.

How does any spider start its travels over the Web? The usual starting points are lists of heavily used servers and very popular pages. The spider will begin with a popular site, indexing the words on its pages and following every link found within the site. In this way, the spiderling system quickly begins to travel, spreading out across the most widely used portions of the Web. If a web page is never linked to in any other page, search engine spiders cannot find it. The only way a brand new page - one that no other page has ever linked to - can get into a search engine is for its URL to be sent by some human to the search engine companies as a request that the new page be included.

In order to scale to hundreds of millions of web pages, Google has a fast distributed crawling system. It uses multiple spiders, usually three at one time. A single URL-server serves lists of URLs to a number of crawlers. Each spider can keep about 300 connections to Web pages open at a time. At its peak performance, using four spiders, Google's system can crawl over 100 pages per second, generating around 600 kilobytes of data each second. It turns out that running a spider which connects to more than half a million servers, and generates tens of millions of log entries.

Keeping everything running quickly means building a system to feed necessary information to the spiders. A major performance stress is DNS lookup. Nowadays, crawler maintains its own DNS cache so it does not need to do a DNS lookup before crawling each document. Each of the hundreds of connections can be in a number of different states: looking up DNS, connecting to host, sending request, and receiving response. These factors make the crawler a complex component of the system.

Figure 2: Working of Spiders or Web Crawlers



When the spider looks at an HTML page, it takes note of two things:

- The words within the page
- Where the words were found

Words occurring in the title, subtitles, Meta-tags and other positions of relative importance are noted for special consideration during a subsequent user search. All these approaches attempt to make the spider operate faster; allow users to search more efficiently, or both. For example, some spiders will keep track of the words in the title, sub-headings and links, along with the 100 most frequently used words on the page and each word in the first 20 lines of text. Lycos uses this approach for spiderling. The push to completeness in this approach is matched by other systems in the attention given to the unseen portion of the Web page, the Meta-tags.

Meta-Tags:

Meta-tag describes your site's content, giving search engines' spiders an accurate summary filled with multiple keywords. The meta-tags allow the owner of a page to specify key words and concepts under which the page will be indexed. This can be helpful, especially in cases in which the words on the page might have double or triple meanings -- the meta-tags can guide the search engine in choosing which of the several possible meanings for these words is correct. Meta-tags are hidden in a document's source, invisible to the reader. Some search engines, however, are able to incorporate the content of meta-tags into their algorithms. No engines penalize sites that use meta-tags properly, so it's recommended that you always include them. The meta-tag is especially important because it's the only tag supported by some engines.
Here's an example of a meta-tag:

```
<html>
<head>
<meta name="description" content="Your site's summary here">
</head>
</html>
```

IV. BUILDING INDEX

Once the spiders have completed the task of finding information on Web pages, the search engine must store the information in a useful way. For this, spiders after finding pages pass them on to another computer program for "indexing". This program identifies the text, links, and other content in the page and stores it in the search engine database's files so that the database can be searched by keyword and whatever more advanced approaches are offered, and the page will be found if your search matches its content.

There are two key components involved in making the gathered data accessible to users:

- The information stored with the data
- The method by which the information is indexed

In the simplest case, a search engine could just store the word and the URL where it was found, but this would make for an engine of limited use, as there would be no way of telling whether the word was used in an important or a trivial way on the page, whether the word was used once or many times or whether the page contained links to other pages containing word.Thus, there would be no way of building the ranking list that presents the most useful pages at the top of the list of search results.

To make for more useful results, most search engines store more than just the word and URL. An engine might store the number of times that the word appears on a page. The engine might assign a weight to each entry, with increasing values assigned to words as they appear near the top of the document, in sub-headings, in links, in the meta-tags or in the title of the page. Each commercial search engine has a different formula for assigning weight to the words in its index. This is one of the reasons that a search for the same word on different search engines will produce different lists, with the pages presented in different orders.

Regardless of the precise combination of additional pieces of information stored by a search engine, the data will be encoded to save storage space. For example, the original Google paper describes using 2 bytes, of 8 bits each, to store information on weighting -- whether the word was capitalized, its font size, position, and other information to help in ranking the hit. Each factor might take up 2 or 3 bits within the 2-byte grouping (8 bits = 1 byte). As a result, a great deal of information can be stored in a very compact form. After the information is compacted, it's ready for indexing.

4.1 Indexing Techniques of Different Search Engines

Google: The heart of Google indexing technique is "PageRank". Google interprets a link from page A to page B as a vote, by page A, for page B. Google also analyses the page that casts the vote. Votes cast by "important" pages, weigh more heavily and make other pages "important."

Yahoo: Yahoo the most popular hierarchically organized search engine uses robots to discover the sites and humans are relied upon for indexing, index includes URL, HTML title tags, very short description of the site.

AltaVista: It uses crawlers to visit every site on the web, indexes all the sites they find there. It also uses meta-tag for indexing.

Lycos: Lycos indexes the title, URL, headings, subheadings and the first 20 lines of text; re-indexing its database fortnightly.

Excite: It uses robots to do full text indexing; it also employs "Intelligent concept extraction" which relies on clustering of words to locate the presence of concepts.

Infoseek: It uses robot to do full text indexing, meta descriptor tags are also indexed, and indexes third and fourth level also.

4.2 The List Of Some Things Indexers Look For:

Headings: The search engines view < h> tags as being terms of emphasis - they give weight to the words within them.

Bold: Of lesser importance than < h> tags. the < b> tags still emphasize terms of importance.

Alt text: Use descriptive short sentences in your alt tags. If it's a picture of a rose, and you're a florist try "Red Rose - Available at 'name' Flower Shop"

Keyword meta tags: Some engines use them directly, some check them as part of a validation process - "do they match the content" If they don't then it is a spam site?

Meta tag: Most engines look at this tag. Use distinct ones throughout your site, and distinct ones for each page. Make them particular to that page.

Key term placement: Terms that are higher up on a page are more heavily weighted.

Key term proximity: Terms that are close together are probably related, thus the site will show up in searches for those terms.

Number of links to the page. If the link to your web site is the only one from a page, it's viewed as being more valuable than being one link among 100.
Page Last Modified -(Freshness)- a page that is updated frequently is favoured.

Keyword frequency across pages. Does the content really talk to the subject about whom the page and the web site are?

Page Size - The engines tend to weigh content at the start of a document more than content further down. If a page is long, look at breaking it into sections. If a page is over 50k, then it's too long.

V. WEB SEARCHING ALGORITHMS

Though nearly all search engines are quite capable of performing keyword matching, they have the disadvantage of being of little or no value when used for searching general topics. For example, when searching for information about Mars, the results may be overcrowded with Web sites such as MarsMusic.com, and other extraneous web sites. Although it may be argued that further detail can help solve this problem, it makes general searches much more difficult, especially if the user doesn't know much about the topic. The Google query evaluation process is show below.

- Parse the query.
- Convert words into word-IDs.
- Seek to the start of the doc-list in the short barrel for every word.
- Scan through the doc-lists until there is a document that matches all the search terms.
- Compute the rank of that document for the query.
- If we are in the short barrels and at the end of any doc-list, seek to the start of the doc-list in the full barrel for every word and go to step 4.
- If we are not at the end of any doc-list go to step 4.
- Sort the documents that have matched by rank and return the top k.

To put a limit on response time, once a certain number (currently 40,000) of matching documents are found, the searcher automatically goes to step 8. This means that it is possible that sub-optimal results would be returned. Researchers are currently investigating other ways to solve this problem. In the past, they sorted the hits according to PageRank, which seemed to improve the situation.

5.1 Kleinberg Web Searching Algorithm

The Kleinberg algorithm operates by performing the following procedure:

- A simple text-based search engine is used to generate a small "root set" of web sites.
- The root set is expanded to include web pages that are pointed to by or that point to existing pages in the set. (This is the "base set".)
- Then each page is assigned an initial value for "hub" and "authority" and these weight vectors are normalized for further processing.
- The hub and authority weights of each page are then updated repeatedly by assigning the sum of all the hub weights of hubs that point to a given authority or assigning the sum of all the authority weights of authorities that are pointed to by a given hub.
- These weights should converge to the eigenvectors. We then select the web sites with the highest hub or authority weights.

5.2 Salsa Web Searching Algorithm

Although SALSA has some major similarities with the Kleinberg algorithm, the central ideas it uses are quite different. Instead of using the mutually reinforcing method of approaching the eigenvector given in the Kleinberg algorithm, SALSA uses random walks across the link structure in order to determine hub and authority weights.

The following is the SALSA algorithm:

- A simple text-based search engine is used to generate a small "root set" of web sites.

- The root set is expanded to include web pages that are pointed to by or that point to existing pages in the set.
- Then, using Markov chain theory, random walks are performed repeatedly in order to approach a good representation of the base set's link structure. (Each random walk is a two-step progression. If two sites in the set are connected by first stepping to a site pointed to by one of the sites, then finding the other site by stepping back, this represents an increased hub value for both sites. Alternatively, authority weights are derived from random walks in which the first step goes back to a site that points to the current site, then following another of its links to another site.)
- These random walks are continued until a decent representation of the link structure has been generated.
- Like the Kleinberg algorithm, the weights should converge to the eigenvectors. We then select the web sites with the highest hub or authority weights.
- The SALSA algorithm does not operate in the same mutually reinforcing manner that the Kleinberg algorithm does. This is because the hub or authority weight of each site does not have a direct effect on the weight of other sites.

VI. RANKING ALGORITHMS

These algorithms rank the search results depending upon their relevance to the search query. There are two categories of these algorithms viz. text based and link based.

6.1 Text-Based Ranking in Conventional Search Engines

The ranking algorithms of a search engine are responsible for returning the results in the decreasing order of their relevance to the search query. To do this, these algorithms rank each of the search results. The ranking scheme used in the conventional search engines is purely Text-Based i.e. the pages are ranked based on their textual content, which seems to be logical. In such schemes, the factors that influence the rank of a page are:

- Location Factors influence the rank of a page depending upon where the search string is located on that page. The search query string could be found in the title of a page or in the leading paragraphs of a page or even near the head of a page.
- Frequency Factors deal with the number of times the search string appears in the page. The more time the string appears, the better the ranking of the page.

Most of the times, the affect of these two factors is consider together. For example, if a search string appears a lot of times near the beginning of a page then that page should have a high rank.

6.1.1 Disadvantages of Text Based Ranking

Though the text based ranking scheme seems to be perfectly logical, it has the following drawbacks:

- Spamming: Spamming promotes the ranking of a page so that it will be listed in the first few results. This can be done by exploiting both the location and frequency factors that influence the ranking produced by a text based ranking algorithm. For example, a malicious user can create a dummy page that has a list of most common keywords in the title of a page or just repeating them in the text of the page. By doing so this dummy page gets a very high rank for most of the queries.

- Abundance and Relevance Problem: The crawler technique returns a large number of results all of which may not be relevant to the initial query specified. The focus should be to return a relatively small number of relevant results.

1) 6.2 Link-Based Ranking Algorithms

The next class of ranking algorithms is the link-based algorithms. These algorithms overcome the drawbacks of the text-based algorithms. They view the web as a directed graph where the web pages form the nodes and the hyperlinks between the web pages form the directed edges between these nodes. This web graph contains useful information. If a web page pi has a link pointing to web page pk, it indicates that the creator of pi considers pk to have relevant information for pi. The strength of the link based algorithms comes from such honest and unbiased opinions that are registered in the hyperlinks. In the rest of the paper, we extensively research the following two link-based algorithms

- PageRanking algorithm
- HITS (Hyperlink Induced Topic Search)

VII. PAGERANK ALGORITHM

PageRank algorithm was proposed by the thesis of Page and Sergey Brin from StanfordUniversity. It is a method for measuring the relative importance of web pages objectively based on the large and heterogeneous graph of the web. They have built the well-known Google search engine utilized the PageRank theory.

7.1 Definition

The general idea of PageRank algorithm is like the academic citation. If an academic paper has been reference by many other papers, it means this paper has valuable information for certain topics. On the hand, if it has never been referred to or just a few, it means that the paper may not have enough good information for related topics. In this section, we will discuss deeply for how the PageRank algorithm works.

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*

$$R(u) = c\sum_{v \in B_u} \frac{R(v)}{N_v} + cE(u) \quad (7.1)$$

This definition was given by Page and Brin. R(u) is the ranking of a web page u. B_u is a set of incoming edges of the vertex (web page) u which are also known as backlinks. N_v is the number of outgoing edges of a vertex v which is one of the web page points to u. The constant c is a factor used for normalization so that the total rank of all web pages is constant. The constant c is smaller than one because there are a number of pages with no forward links and their weight is lost from the system.

The E is some vector over the web pages that corresponds to a source or rank which solve the problem of rank sinks mentioned by Page and Brin. When there is a cycle for several web pages point to each other and there is no outgoing edges point to other web pages. If there is a web page points to this loop, the loop will accumulate rank but never distribute any rank. This forms the rank sink problem. The equation is recursive but it may be computed by starting with any set of ranks and iterating the computation until it converges.

The equation is a probability distribution which can also be thought of as modelling the behaviour of a "random surfer" who started by surfing a random page, and keep clicking on the links. This forms the first part of the equation, $\sum_{v \in B_u} \frac{R(v)}{N_v}$.

When he got into a loop of web pages, it is unlikely that he will continue in the loop forever. The surfer will jump to some other page. This is modelled by the distribution of E. E vector is often uniform, but it can also use to customize the PageRank. Brin and Page, did many experiments to get the best value of E as 0.15.

Google search engine used the following definition which choosing the uniform value of E:

Definition 2 *We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:*

$$PR(A) = (1-d) + d\ (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn)) \quad (7.2)$$

Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.

PageRank is a numeric value that represents how important a page is on the web. Google figures that when one page links to another page, it is effectively casting a vote for the other page. The more votes that are cast for a page, the more important the page must be. Also, the importance of the page that is casting the vote determines how important the vote itself is. Google calculates a page's importance from the votes cast for it.

7.2 Pseudo Algorithm for PageRank

Algorithm PageRank (G)
Input: An n-element array of G which represent set of vertex or web pages
Output: An n-element array of PR which represent PageRank for each web page
Comment: Start with a uniform distribution
For $i \leftarrow 0$ to $n-1$ do
Let J be a n-element of array

$J[i] \leftarrow \frac{1}{n}$

$damp \leftarrow 0.85$

Repeat

For $i \leftarrow 0$ to $n-1$ do

Let PR be a n-element of array

Comment:

$PR[i] \leftarrow 1 - damp$

For all pages Q such that Q links to PR[i] do

Let O_q be the number of outgoing edge of Q

$PR[i] \leftarrow PR[i] + damp \times \frac{J[Q]}{O_q}$

If the difference between J and PR is small do

Return PR

For $i \leftarrow 0$ to $n-1$ do

$J[i] \leftarrow PR[i]$

7.3 Examples of PageRank

Let's use the above algorithm to calculate the PageRank of the following examples.

Example 1

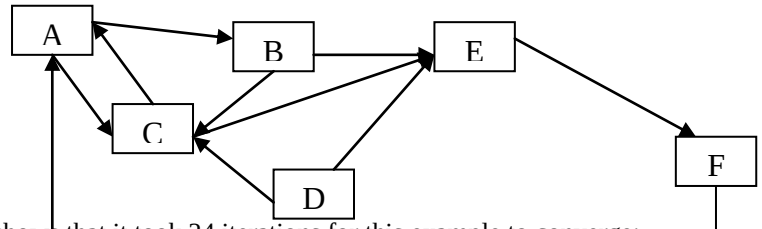


Figure 8.1

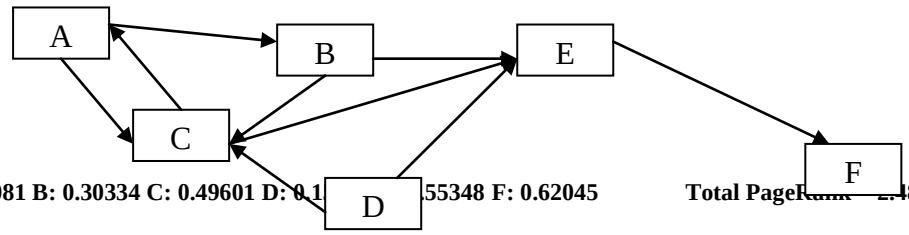
The result shows that it took 24 iterations for this example to converge:

A: 0.16667 B: 0.16667 C: 0.16667 D: 0.15000 E: 0.16667 F: 0.16667
A: 0.36250 B: 0.30406 C: 0.50412 D: 0.15000 E: 0.35725 F: 0.62364
A: 0.89435 B: 0.53010 C: 0.81914 D: 0.15000 E: 0.78718 F: 0.81910
A: 1.19437 B: 0.65761 C: 1.00084 D: 0.15000 E: 0.91859 F: 0.93080
A: 1.36654 B: 0.73078 C: 1.10511 D: 0.15000 E: 0.99400 F: 0.99490
A: 1.46534 B: 0.77277 C: 1.16495 D: 0.15000 E: 1.03728 F: 1.03169
A: 1.52204 B: 0.79687 C: 1.19928 D: 0.15000 E: 1.06211 F: 1.05280
A: 1.55457 B: 0.81069 C: 1.21899 D: 0.15000 E: 1.07636 F: 1.06491
A: 1.57324 B: 0.81863 C: 1.23030 D: 0.15000 E: 1.08454 F: 1.07186
A: 1.58396 B: 0.82318 C: 1.23678 D: 0.15000 E: 1.08924 F: 1.07585
A: 1.59011 B: 0.82580 C: 1.24051 D: 0.15000 E: 1.09193 F: 1.07814
A: 1.59363 B: 0.82729 C: 1.24264 D: 0.15000 E: 1.09347 F: 1.07945
A: 1.59566 B: 0.82816 C: 1.24387 D: 0.15000 E: 1.09436 F: 1.08021
A: 1.59682 B: 0.82865 C: 1.24457 D: 0.15000 E: 1.09487 F: 1.08064
A: 1.59749 B: 0.82893 C: 1.24498 D: 0.15000 E: 1.09516 F: 1.08089
A: 1.59787 B: 0.82910 C: 1.24521 D: 0.15000 E: 1.09533 F: 1.08103
A: 1.59809 B: 0.82919 C: 1.24534 D: 0.15000 E: 1.09543 F: 1.08111
A: 1.59822 B: 0.82924 C: 1.24542 D: 0.15000 E: 1.09548 F: 1.08116
A: 1.59829 B: 0.82927 C: 1.24546 D: 0.15000 E: 1.09551 F: 1.08119
A: 1.59833 B: 0.82929 C: 1.24549 D: 0.15000 E: 1.09553 F: 1.08120
A: 1.59835 B: 0.82930 C: 1.24550 D: 0.15000 E: 1.09554 F: 1.08121
A: 1.59837 B: 0.82931 C: 1.24551 D: 0.15000 E: 1.09555 F: 1.08122
A: 1.59838 B: 0.82931 C: 1.24552 D: 0.15000 E: 1.09555 F: 1.08122

A: 1.59838 B: 0.82931 C: 1.24552 D: 0.15000 E: 1.09555 F: 1.08122 Total PageRank = 6.00000

Observe the PageRank of D, we found that this page got 0.15 even if it has no outgoing edge. The reason is because of the equation 2: PR(D) = 1-0.85 + 0 = 0.15. Thus, the minimum value of the PageRank of a page is 0.15. The total PageRank is the number of vertexes, which are 6.

Example 2



A: 0.36081 B: 0.30334 C: 0.49601 D: 0.1 .55348 F: 0.62045 Total PageRank 2.48409

Figure 8.2

This example is almost the same as Exa n't have any outgoing edge. The total of PageRank is not equal to 6 because F isn't passing the age. This is called dangling link, and it is an issue for the PageRank. The solution is to remove all the dangling links until all the PageRanks are calculated, and add them back after that. This will not affect things significantly.

VIII. HITS (HYPERLINK INDUCED TOPIC SEARCH) ALGORITHM

HITS algorithm was proposed by Jon M. Kleinberg. The main objective of this algorithm is to produce the ordering of the search results i.e. to rank the search results by their relevance to the search query and present the results so that the most relevant results are presented before the rest. HITS is a link based algorithm and unlike the PageRanking algorithm, it uses both the in-degree and out-degree of a node to determine its final ranking.

8.1 HITS Algorithm

Let 'q' be the query given to the search engine. HITS algorithm produces the ordered results for this query by passing the query through the following steps:

8.1.1 Constructing a Focused Sub-Graph

The initial input to the HITS algorithm is a small set of web pages, Sq. To obtain this set, the query 'q' is first processed by a text based search engine like AltaVista. The top 't' results returned by this search engine form the set Sq. Let us call this the root set Rq. This set should have the following properties:

- The set should be relatively small
- The set should be rich in relevant pages
- The set should contain most of the strongest authorities
-

The first property can be satisfied by choosing 't' to be small. Since Rq is formed from the results of a text based search engine, it will be rich in the relevant pages. So the second property is also satisfied. But according to, it falls short of satisfying the third property because there will be extremely few links between the pages of Rq. So expand Rq to another set called the base set Bq, which satisfies all three properties. To do this we use a Region Growing Method.

Region Growing Method: Base set comprises of all the pages in the root set and also all the pages that are pointing to or pointed by any of the pages in the root set. To make sure that this set does not become too big, put a constraint saying that any page in the root set can bring in at most 'd' pages. If any page in the root set is referenced by more than d pages then an arbitrary set of those pages is picked.

The graph and refinements: Next construct a directed graph representing the base set. The web pages in the base set form the nodes of this graph and the hyperlinks between the web pages form the directed edges. If there are 'n' nodes in this graph then this graph can be represented by an n*n adjacency matrix. Each entry (i, j) of this matrix has a value 1 if there is an edge between the corresponding nodes i and j. Otherwise it is 0.

8.1.2 Identifying Hubs and Authorities

Kleinberg assigns two roles to each node in the graph [1]. Consider a directed edge from V1 to V2. Here V1 is called the Hub and V2 is called the Authority.

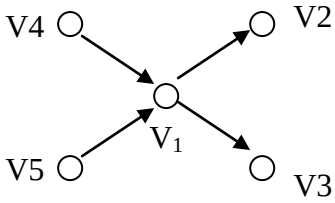


Figure9.1: V4, V5 are the Hubs; V2, V3 are the Authorities and V1 acts both as a Hub and Authority

Authority: A valuable and informative page is usually pointed by a large number of hyperlinks i.e. it has a large in degree. Such a webpage is called an Authority.

Hub: A webpage that points to many authority pages is itself a useful resource and is called a Hub. A hub has a very large out-degree.

Significance of Hubs and Authorities:

Kleinberg suggests that hubs and authorities exhibit a mutually reinforcing relationship. A good hub is a page that points to many good authorities, and, a good authority is a page that is pointed to by many good hubs. This relationship helps us overcome the difficulties faced in the situations where a web page, that is highly relevant to the search query, does not contain the query text. Earlier algorithms like Google might not return such pages but HITS, by utilizing this relationship, will successfully return them. This is because the hub pages tend to pull together authorities on a common topic.

8.2 Drawbacks of HITS algorithm

The HITS algorithm suffers from the following drawbacks:

- Topic drift when there are non-relevant pages in the root set and they are strongly connected. If the root set itself contains no-relevant pages then this will be reflected on to the pages in the base set. Furthermore, the web graph, which is constructed from the pages in the base set, does not have the most relevant nodes and as a result the algorithm does not find the highest ranked authorities and hubs.
- The rating of authorities and hubs could be inflated due to flaws by web page designer. The whole HITS algorithm is based on the assumption that a user creates a web page with his/her honest opinion i.e. when he puts a hyperlink from his page to another authority page, he honestly believes that the authority page is in some way related to his page. HITS is bound to fail if we design web pages such that a lot of highly ranked hubs point to an authority that actually does not have the relevant information.
- A page contains links to a large number of separate topics may receive a high hub rank which is not reflective of its relevance to the submitted query. An example of such a page would be your personal home page that contains many links to topics of your interest like sports, business, weather, etc. Though this page is not the most relevant source for any of the information, it still has a very high hub ranking if it points to highly ranked authorities.
- When a search topic is too narrow, the results for a broader topic may overwhelm the results for that specific topic. This drawback always exist in any given search engine.

IX. CONCLUSION

A large-scale web search engine is a complex system and much remains to be done. The biggest problem facing users of web search engines today is the quality of the results they get back. While the results are often amusing and expand users' horizons, they are often frustrating and consume precious time. In order to accomplish this, use of hypertextual information consisting of link structure and link (anchor) text is requisite along with the use of proximity and font information.

While evaluation of a search engine is difficult, it has been subjectively found that Google returns higher quality search results than current commercial search engines. The analysis of link structure via PageRank allows Google to evaluate the quality of web pages. The use of link text as a description of what the link points to; helps the search engine return relevant (and to some degree high quality) results. Finally, steps towards extending the use of link structure and link text are being taken. Simple experiments indicate PageRank can be personalized by increasing the weight of a user's home page or bookmarks.

Overall, these algorithms have shown the potential of using links on the web in identifying the best search results. Given the trend that on WWW semi-structured data or data with schema information such as XML are gaining popularity, new indexing techniques have already been developed for searching engines.

It appears that future research on ranking algorithms should also try to incorporate structural information available at data level into ranking algorithms. However it is difficult to say that one of these algorithms is better than the other. But, it is clear that they operate in different ways and hence have different strengths and weaknesses. Nonetheless, these algorithms and others that take into account the link-structure of the Internet may turn out to be the next major advancement in future for refining the web searching technology, as a Web search engine is a very rich environment for research ideas. We have far too many search engines to list here so we should not expect this Future Work to become much shorter in the near future.

X. REFERENCES

[1]. S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg, Automatic resource compilation by analyzing hyperlink structure and associated text, Proc. 7th International World Wide Web Conference, 1998, and March 2001.
[2]. Available at http://decweb.ethz.ch/WWW7/1898/com1898.htm
[3]. Borodin, G. Roberts, J Rosenthal, and P. Tsaparas, Finding Authorities and Hubs From Link Structures on the World Wide Web, Proc. 10th International World Wide Web Conference, 2001, and June 2001.
[4]. Available at http://citeseer.nj.nec.com/444373.html
[5]. Phil Craven, Google's PageRank Explained and how to make the most of it, Web Workshop. Available at http://www.webworkshop.net/pagerank.html
[6]. Ian Rogers, The Google PageRank Algorithm and How It Works, IPR May 2002.
[7]. Available at http://www.iprcom.com/papers/pagerank/index.html
[8]. Best of the Web 1994 - Navigators http://www.botw.org/1994/awards/navigators.html
[9]. Google Search Engine Technology http://www.google.stanford.edu/
[10]. Mauldin, Michael L. Lycos Design Choices in an Internet Search Service, IEEE Expert Interview http://www.computer.org/pubs/expert/1997/trends/x1008/mauldin.htm
[11]. Search Engine Watch http://www.searchenginewatch.com/
[12]. Robots Exclusion Protocol:
[13]. http://www.info.webcrawler.com/mak/projects/robots/exclusion.html
[14]. http://www.robotstxt.org/wc/exclusion.html
[15]. Web Growth Summary: http://www.mit.edu/people/mkgray/net/web-growth-summary.html
[16]. Search Engines for WWW: http://www.searcheasy.htmlplanet.com/
[17]. Google Bomb: www.corante.com/microcontent/articles/googlebombs.shtml