Information Retrieval and Search Engine

1. Concepts of Internet

The Internet is a global system of interconnected governmental, academic, corporate, public, and private computer networks. It is based on the networking technologies of the Internet Protocol Suite. The Internet is also the communications backbone underlying the World Wide Web (WWW).

A Worldwide network of computers and data communications equipment connected by routers that are configured to pass traffic among any computers attached to network .

Internet is made up of millions of computers linked together around the world in such a way that information can be sent and retrieved from any computer to another at any place in the world and at any time as long as those computers are connected to that network or server. These computers can be at homes, schools, universities, government department or businesses. Also they can be of any types from personal computer, workstation or a network companies.

The Internet is often described as 'a network of networks' because all the smaller networks of organizations are linked together into the one giant network called the Internet . All computers in the Internet communicate with each other using a certain standard or protocols, such as Transmission Control Protocol/Internet Protocol (TCP/IP). This protocol allows computer to share files and services .

2.Internet protocols

The Internet Protocol is the method or protocol by which data is sent from one computer to another on the Internet [15]. These protocols are illustrating as below:

1. TCP/IP Protocol (Transmission Control Protocol / Internet Protocol)

The Internet protocols consist of a suite of communication protocols, of which the two best known are the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The TCP provides reliable transmission of data in an IP environment. TCP offers reliability by providing connection-oriented, end-to-end reliable packet delivery through an internetwork. It does this by sequencing bytes with a forwarding acknowledgment number that indicates to the destination of the next byte the source expects to receive. The reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets. A time-out mechanism allows devices to detect lost packets and request retransmission. TCP offers efficient flow control, which means that, when sending acknowledgments back to the source, the receiving TCP process indicates the highest sequence number it can receive without overflowing its internal buffers. Full-duplex operation means that TCP processes can both send and receive at the same time [16].

IP takes care of low level addressing and transport of data packets over a multi-subnet network. The IP protocol itself is useless without one of its peers, TCP, User Datagram Protocol (UDP), or Internet Control Message Protocol (ICMP). UDP is a datagram protocol that can send a message to a certain port on a certain computer without any guaranty of arrival. Its biggest use can be found in multimedia uses such as broadcasting, where retransmission is no real option. ICMP is the control protocol that is used by IP for internal purposes [9].

2. HyperText Transfer Protocol (HTTP)

The hypertext transfer protocol is a request, response protocol with the main purpose of sending files from the server to a client. The command set that is provided by HTTP is very limited. The most important of them take the form of the request of a resource specified by a Uniform Resource Locator (URL), possibly with arguments. The HTTP server (or web server) doesn't look at the contents of a file, so for related files such as images or style-sheets, new requests must be made. Further it is also not possible for a single request to request multiple files. HTTP was designed to be the protocol to carry hypertext files written in the Hypertext Mark-up Language (HTML) [9]. HTTP runs on top of the TCP/IP protocol [17].

3. File Transfer Protocol (FTP)

File Transfer Protocol is a common method of transferring files via the Internet from one computer to another [18]. A commonly used protocol for exchanging files over any network, which supports the TCP/IP protocol. FTP is perhaps the best, fastest and most efficient way to transfer large files over the Internet [19].

FTP works by means of a client-server architecture; the user runs client software to connect to a server on the Internet [20]. It is a more important one, because it hides the details of individual computer systems [21].

4. Simple Mail Transfer Protocol (SMTP) and Post Office Protocol (POP)

SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with other protocol POP that lets the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and POP for receiving e-mail [22].

5. Telnet

Telnet is a user command and an underlying TCP/IP protocol for accessing remote computers. Through Telnet, an administrator or another user can access someone else's computer remotely. On the Web, HTTP and FTP protocols allow you to request specific files from remote computers, but not to actually be logged on as a user of that computer. With Telnet, you log on as a regular user with whatever privileges you may have been granted to the specific application and data on that computer. A Telnet command request looks like this (the computer name is made-up):

```
telnet the.libraryat.whatis.edu
```

The result of this request would be an invitation to log on with a userid and a prompt for a password. If accepted, you would be logged on like any user who used this computer every day. Telnet is most likely to be used by program developers and anyone who has a need to use specific applications or data located at a particular host computer [23]. Telnet requires two arguments: the name of a computer on which the server runs and the protocol port number of the server [21].

3. Internet Strategy

Every computer connected to the Internet is assigned a unique Internet Protocol (IP) address. When client point-and-click on a link, browser sends out a request that is addressed to the website that houses the content client want. Routers along the way read the data packet’s address and relay it along the best route available. When the data packet arrives at the website, the server reads the request and sends the requested page back to client computer via a return address in the data packet: this is computer’s IP address. The data packet is routed back to the client (in actuality, several data packets) and browser interprets the content and displays the page .

Internet communication is made possible by the (TCP/IP) protocol on the computer. This protocol sends information to the connected computer, which then passes information onto other computers until it reaches the destination .

The message breaks into packets of information and sends each packet separately. If a packet is lost, the TCP/IP process simply resends the missing packet. If one of the connections between two computers is not working the nearby computers simply stop using that route until the connection is repaired. The Internet protocol recognizes the damage and route around it. The routing flexibility of TCP/IP protocol assures an accurate and steady flow of information, regardless of any one user's connection .

4. IP Address

TCP/IP assigns a unique number to every workstation in the world. This "IP number" is a four byte value that, by convention, is expressed by converting each byte into a decimal number (0 to 255) and separating the bytes with a period .

Most Internet protocols specify that addresses be supplied in the form of a fully-qualified host name or an IP address in dotted decimal form. However, spammers and others have found a way to obfuscate IP addresses by supplying the IP address as a single large decimal number.

Remember that IP addresses are 32-bit quantities. We write the address in dotted decimal for the convenience of humans; the computer still interprets dotted decimal as a 32-bit quantity. Therefore, writing the address as a single large decimal number will still allow the computer to see the address as a 32-bit number. For that reason, the following URLs will all take you to the same Web page [27]:

- http://www.garykessler.net
- http://209.198.111.31
- http://3519442719

5. URL (Uniform Resource Locator)

Address of a resource on the Internet. The resource can be any type of file stored on a server, such as a Web page, a text file, a graphics file, or an application program. The address contains three elements: the type of protocol used to access the file (e.g., HTTP for a Web page, ftp for an FTP site); the domain name or IP address of the server where the file resides; and, optionally, the pathname to the file (i.e., description of the file's location). For example, the URL http://www.britannica.com/heritage instructs the browser to use the HTTP protocol, go to the www.britannica.com Web server, and access the file named heritage [28]. So, URLs are composed of several parts, as shown in the figure (2-1) below [29]:

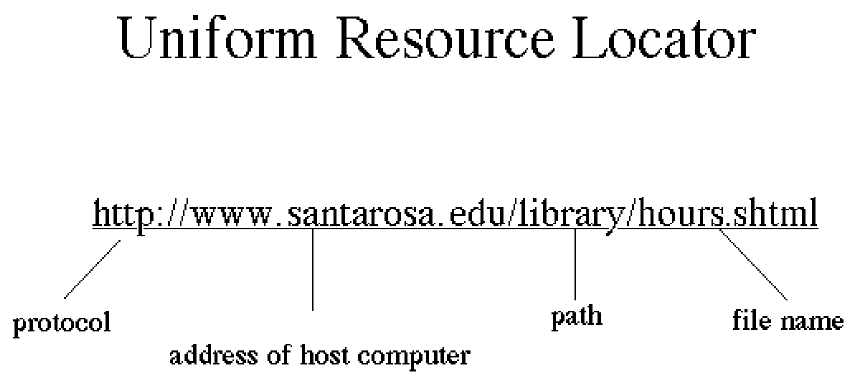


Figure (2-1) URL's component

To find a web page, URL can be entered into the browser and go directly to the web page. URL specifies the Internet address of a file or page stored on the web server. Every file on the Internet has a unique URL [6].

6. Domain Name System (DNS)

Internet system to translate names into IP addresses [30]. It is a combination of words and/or numbers of user choice, as well as an appropriate extension that is also of user choice. A full name includes everything from the www, the words in the middle, and the extension at the end [31]. Domain names are easy-to-remember names (URLs and email addresses) that are associated with one or more IP addresses. Since a web page is defined by its URL, the page can move to a different IP address without affecting visitors.

Example: www.singlespeed.com

- singlespeed.com is the domain name.
- com is the top level domain.
- singlespeed is a subdivision of com, and represents the second-level domain.
- www is a subdomain (also known as third-level domain or CNAME).

Top-level domains are the last part of a domain name - the letters after the last period. Some examples are: biz com org .Second-level domains are directly below top-level domains. Some current examples are:

Table (2-1) Second-level Domain Name

Second-level Domain	Domain Name
Google	google.com
Wikipedia	wikipedia.org
Ontariotravel	ontariotravel.com

Third-level domains are also known as subdomains and CNAMEs. In a URL, the subdomain is written before the domain name. Here are some examples [32]:

Table (2-2) Third-level Domain Name

Sub domain	URL
affiliates	http://affiliates.art.com
www	http://www.rockfound.org
men	http://men.style.com
mail	http://mail.google.com

bus	http://www.bus.umich.edu

A single Web server can serve Web sites for multiple domain names, but a single domain name can point to only one machine. For example, Apple Computer has Web sites at www.apple.com, www.info.apple.com, and store.apple.com. Each of these sites could be served on different machines. Then there are domain names that have been registered, but are not connected to a Web server. The most common reason for this is to have e-mail addresses at a certain domain name without having to maintain a Web site. In these cases, the domain name must be connected to a machine that is running a mail server .

7. Client / Server Technique

A network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources, such as files, devices, and even processing power.

In a client-server environment files are stored on a centralized, high speed file server PC that is made available to client PCs. Network access speeds are usually fast which is reasonable given the vast numbers of clients that this architecture can support. Nearly all network services like printing and electronic mail are routed through the file server, which allows networking tasks to be tracked. In the client-server, the client PCs separate and subordinate to the file server. The clients' primary applications and files are stored in a common location. File servers are often set up so that each user on the network has access to his or her "own" directory, along with a range of ”public" directories where applications are stored. If the two clients above want to communicate with each other, they must go through the file server to do it. A message from one client to another is first sent to the file server, where it is then routed to its destination. With tens or hundreds of client PCs, a file server is the only way to manage the often complex and simultaneous operations that large networks require .

Server can be a computer that holds files in a central location, delivering up information when requested. A server also can be a program running on a server computer and can be a program that interfaces with a client program .

8.The World Wide Web (WWW)

The World Wide Web (also known as “WWW”, “Web, or “W3”) is the universe of network-accessible information, the embodiment of human knowledge. It allows people to share information globally. That means it allows anyone to read and publish documents freely. The World Wide Web hides all the detail of communication protocols, machine locations, and operating systems from the user. It allows users to point to any other Web pages without any restrictions. Brin defines the Web as follows (1998):

“The Web is a vast collection of completely uncontrolled heterogeneous documents”.

The Web is accessible to anyone via a Web browser. Search engines answer tens of millions of queries every day (Brin, 1998). The amount of information on the Web grows exponentially. If we only count the textual data, it is estimated to be in the order of one terabyte (Baeza-Yates, 1999). Of course, the size of multimedia data types (image, audio, and video) is even bigger. Since this data is managed by many individuals, organizations, and companies, the Web is unstructured. Thus, the WWW can be seen as a large unstructured and ubiquitous database. Finding useful information on the Web is a very difficult task. There is a need to develop tools to manage, retrieve, and filter information in the big database. The solutions to search data in the WWW efficiently. In other words, how do we develop a fast, robust, and accurate search engine? [8].

9. Web Browser

Web browsers are software programs that read and interpret the code written on Web pages to create a nice visual display. The Web itself is a bunch of data, including raw HTML, Cascading Style Sheet (CSS) and other types of code. Without a Web browser, you'd be hard-pressed to find the specific information you're seeking on the Web [36].

Before data can be sent and received between your browser and a web server, a connection to the server's port must be established. Browsers use web addresses (such as http://www.google.com/) to specify a location of the information/connection request. The web address is converted from its user-friendly text format to a numerical TCP/IP address. Web browsers use TCP/IP to create a connection between your computer and a web server. TCPs readies data--it breaks web data down into 1,500-bit packets of data and readies it for transmission between your computer and a web server. The IP sends data and ensures that it reaches the recipient of the information--the computer or web server.

Web browsers use HTTP --a sub protocol of TCP/IP-- to communicate with web servers. Requests for connections, the termination of connections and requests for data--such as HTML and pictures--are communicated with HTTP. Once the connection between your computer and the web server is approved, data can be sent over the ports and later interpreted by your Internet browser as text and images. The most common form of data retrieved via HTTP from a web page on a web server is an HTML document -- it contains the text you see on a web page and links to embedded files such as images and flash. HTML documents contain instructions called "meta tags," which give your Internet browser general instructions on how the web page should be displayed -- though the exact interpretation of elements such as text spacing, paragraph alignment and font size varies between Internet browsers .

10. Web Server

A web server is a computer program that delivers (serves) content, such as web pages, using the (HTTP), over the WWW. The term web server can also refer to the computer or virtual machine running the program [38]. Every web server has a unique address so that other computers connected to the internet know where to find it on the vast network [39].

A Web server uses the client/server model and the (HTTP), serves the files that form Web pages to Web users (whose computers contain HTTP clients that forward their requests). Every computer on the Internet that contains a Web site must have a Web server program. Two leading Web servers are Apache , the most widely-installed Web server, and Microsoft's Internet Information Server (IIS) [40].

Every Web server has an IP address and possibly a domain name. For example, if the client enters the URL http://www.pcwebopedia.com/index.html in browser, this sends a request to the server whose domain name is pcwebopedia.com. The server then fetches the page named index.html and sends it to browser .

In theory, web servers stay connected to the Internet 24 hours a day, seven days a week, 365 days a year. In truth, they experience occasional downtime due to maintenance and technical problems. Web servers with consistent records of an uptime of 99.5% or better are considered reliable.

11. Hyper Text Markup Language (HTML)

HTML which is a simple "programming" language based on **tags** that provided instructions for creating web pages such as where to place an image or which text to be made large or which one to be put in *italics*. Web browsers could understand and interpret HTML and display the web page to the viewer. Remember, the images and other media such as music files, animation are separate from the actual web page HTML file and are only *embedded* on the page when it's displayed by a browser [42].

The tags define the page layout, fonts and hypertext links to other documents on the Web. Each link contains the URL (the address) of a Web page residing on the same server or any server worldwide, hence "WWW". The HTML also defines all the graphic elements used on the page, which are separate files on a local or remote server [43].

12. HTML Element

HTML documents are composed entirely of HTML elements that, in their most general form have three components: a pair of element ***tags*** with a "start tag" and "end tag"; some element ***attributes*** given to the element within the tags; and finally, all the actual textual and graphical information ***content*** that will be rendered on the display. An HTML element is everything between and including the tags. A **tag** is a keyword enclosed in angle brackets. A common form of an HTML element is:

<tag>*content to be rendered*</tag>

The name of the HTML element is also the name of the tag. Note that the end tag's name starts with a slash character, "/".

The most general form of an **HTML element** is:

<tag attribute1="value1" attribute2="value2">*content to be rendered*</tag>

By not assigning attributes most start tags default their attribute values [43]. There are some basic types of tags:

1. The <html> tag

The <html> tag surrounds the entire HTML document. This tag tells the client browser where the document begins and ends [44].

<html>
... document contents ...
</html>.

2. The Head Tag

The heading section of the document contains certain heading information for the document. The document’s title, meta information, and, in most cases, document scripts are all contained in the <head> section [44]. Heading of the HTML : <head>...</head>. For example [43]:

<head>
<title>The title</title>
</head>

3. The Title Tag

The <title> element determines what the browser displays as the page title—on Windows machines, the document title appears in the browser’s title bar as shown [44,45].

<title>Document Title</title>

4. The Meta Tag

The <head> section of document can also include <Meta> tags. These tags are not rendered as visible text in the document—they are used to pass information and commands to the client browser.

As its name implies, the <meta> tag contains meta information for the document. Meta information is information about the document itself, instead of information about the document's contents. Most of a document's meta information is generated by the Web server that delivers the document. However, using <meta> tags, supply different or additional information about the document.

It always includes at least a minimum amount of information in documents to aid search engines in correctly categorizing documents. Two important pieces of meta information are a **description** of the document and **keywords** relating to its content. The description and keywords information is provided by the following two <meta> tags:

<meta name="description" content="The latest movie news">

<meta name="keywords" content="movie, movies, production, genre, sci fi, horror, drama, comedy, anima, manga, news, chat, bbs, discuss, review, recent"> [44, 45]. Table (2-3) abstracts the processing of important HTML tags [11].

Table (2-3) the processing of important HTML tags

Tag Type	**Processing**
<TITLE>	Determine the title of the web page.
<META>	Determine keyword set, Author's name, publishing date, modification date, descriptor, and the title if the <TITLE> tag didn't exists in the web page.

<BODY>	Determine the general text color of the web page.
<FONT>	Determine the (color, face, and size)of each word occurred in the web page.
<H1>...<H6>	Determine the style and the size of each word occurred in the web page.
<B>, <I>, <U>	Determine the style of the word occurred in the web page.
<STRONG>, <SMALL>	Determine the size of the word occurred in the web page.
<A>	Determine the text that used in representing the links from the current web page to another.
<AREA>	Determine a specified area from an image that used in representing the links from the current page to another.
<IMG>	Determine the text that used for describing an image during the loading time. The text is represented by alt attribute.

- *Information Retrieval and Search Engine*

Baeza-Yates and Ribeiro-Neto [1999] define Information Retrieval (IR) as the "part of computer science which studies the retrieval of information from a collection of written documents.

The retrieved documents aim at satisfying a user information and usually need to be expressed in natural language." Salton and McGill [1983] note that "information retrieval is concerned with the representation, storage, organization, and accessing of information items." [12].

Clearly, this full description of the user information need cannot be used directly to request information using the current interfaces of Web search engines. Instead, the user must first translate this information need into a query which can be processed by the search engine (or IR system). In its most common form, this translation yields a set of keywords (or index terms) which summarizes the description of the user information need. Given the user query, the key goal of an IR system is to retrieve information which might be useful or relevant to the user. The emphasis is on the retrieval of information as opposed to the retrieval of data [2]. In data retrieval, the result of a query must be accurate: it should return the exact match tuples of the query, no more and no less. If there is no change to the database, the result of a query executed at different times should be the same. Data retrieval deals with data that has a well-defined structure and semantics (e.g. a relational database). The goal of an IR system is to retrieve all the documents, which are relevant to a query while retrieving as few non-relevant documents as possible. To achieve this goal, IR needs users to provide a set of words which convey the semantics of the information need. Also, a document is represented by a set of keywords or index terms for IR to extract. These keywords or index terms can be derived from information experts or a computer through eliminating articles and connectives, the use of stemming (which reduces distinct words to their common grammatical root), and identifying nouns (which eliminates adjectives, adverbs, and verbs).

There is a difference between IR and searching the Web. IR allows access to whole documents, whereas, search engines do not. The reason is that it is too expensive to store all the Web pages locally and too slow to access remotely on other Web servers [8].

Thus a search engine is the practical application of information retrieval techniques to large scale text collections [1].

Search Engines

Search engines allow users to search the internet for content using keywords. Although the market is dominated by a few, there are many search engines that people can use. When a user enters a query into a search engine, a search engine results page (SERP) is returned, ranking the found pages in order of their relevance. How this ranking is done differs across search engines.

List of the 10 best search engines in 2021, ranked by popularity.

1. ***Google***
2. ***Microsoft Bing***
3. ***Yahoo***
4. ***Baidu***
5. ***Yandex***
6. ***DuckDuckGo***
7. ***Ask.com***
8. ***Ecosia***
9. ***Aol.com***
10. ***Internet Archive***

Types of Search Engines

The term "search engine" is often used generically to describe both crawler-based search engines and human-powered directories. These two types of search engines gather their listings in radically different ways .

1. Crawler-Based Search Engines

Crawler-based search engines, such as Google, create their listings automatically. They "crawl" or "spider" the web, then people search through what they have found. If you change your web pages, crawler-based search engines eventually find these changes, and that can affect how you are listed. Page titles, body copy and other elements all play a role.

2. Human-Powered Directories

A human-powered directory, such as the Open Directory, depends on humans for its listings. You submit a short description to the directory for your entire site, or editors write one for sites they review. A search looks for matches only in the descriptions submitted. Changing your web pages has no effect on your listing. Things that are useful for improving a listing with a search engine have nothing to do with improving a listing in a directory. The only exception is that a good site, with good content, might be more likely to get reviewed for free than a poor site.

3. "Hybrid Search Engines" Or Mixed Results

In the web's early days, it was used to be a search engine either presented crawler-based results or human-powered listings. Today, it extremely common for both types of results to be presented. Usually, a hybrid search engine will favor one type of listings over another. For example, MSN Search is more likely to present human-powered listings from LookSmart. However, it does also present crawler-based results (as provided by Inktomi), especially for more obscure queries .

Search Engines Working

What follows is a basic explanation of how search engines work. More detailed and technical information about current methods used by search engines [47].

- Keyword Searching
- Refining Search
- Relevancy Ranking
- Meta Tags
- Concept-based Searching

Keyword Searching: This is the most common form of text search on the Web. Most search engines do their text query and retrieval using keywords. For example, with the subject webpage (how search engines work), useful keywords and key phrases for this page would be "search," "search engines," "search engine methods," "how search engines work," "ranking" "relevancy," "search engine tutorials," etc. Those keywords would actually tell a user something about the subject and content of this page. Unless the author of the Web document specifies the keywords for his document (this is possible by using meta tags), it's up to the search engine to determine them. Essentially, this means that search engines pull out and index words that appear to be significant. Since search engines are software programs, not rational human beings, they work according to rules established by their creators for what words are *usually* important in a broad range of documents. The title of a page, for example, usually gives useful information about the subject of the page. Words that are mentioned towards the beginning of a document are given more weight by most search engines. The same goes for words that are repeated several times throughout the document. Some search engines index every word on every page. Others index only part of the document. Full-text indexing systems generally pick up every word in the text except commonly occurring stop words such as "a," "an," "the," "is," "and," "or," and "www." Some of the search engines discriminate upper case from lower case; others store all words without reference to capitalization. There is a problem with keyword searching Keyword searches have a tough time distinguishing between words that are spelled the same way, but mean something different (i.e. hard cider, a hard stone, a hard exam, and the hard drive on computer). This often results in hits that are completely irrelevant to query. Some search engines also have trouble with so-called stemming -- i.e., if you enter the word "big," should they return a hit on the word, "bigger?" What about singular and plural words? What about verb tenses that differ from the word you entered by only an "s," or an "ed"? Search engines also cannot return hits on keywords that mean the same, but are not actually entered in query. A query on heart disease would not return a document that used the word "cardiac" instead of "heart" [47, 48].

Refining Search: Refining search options differ from one search engine to another, but some of the possibilities include the ability to search on more than one word, to give more weight to one search term than you give to another, and to exclude words that might be likely to muddy the results. You might also be able to search on proper names, on phrases, and on words that are found within a certain proximity to other search terms. Some search engines also allow to specify what form of results to appear in, and whether restrict search to certain fields on the internet (i.e., usenet or the Web) or to specific parts of Web documents (i.e., the title or URL). Many, but not all search engines allow using so-called Boolean operators to refine search. These are the logical terms AND, OR, NOT, and the so-called proximal locators, NEAR and FOLLOWED BY.

Relevancy Rankings: Most of the search engines return results with confidence or relevancy rankings. In other words, they list the hits according to how closely they think the results match the query. However, these lists often leave users shaking their heads on confusion, since, to the user, the results may seem completely irrelevant. Most search engines use search term frequency as a primary way of determining whether a document is relevant. If you're researching diabetes and the word "diabetes" appears multiple times in a Web document, it's reasonable to assume that the document will contain useful information. Therefore, a document that repeats the word "diabetes" over and over is likely to turn up near the top of your list. Some search engines consider both the frequency and the positioning of keywords to determine relevancy, reasoning that if the keywords appear early in the document, or in the headers, this increases the likelihood that the document is on target. For example, one method is to rank hits according to how many times keywords appear and in which fields they appear (i.e., in headers, titles or plain text). Another method is to determine which documents are most frequently linked to other documents on the Web. The more clearly relevant the results are, the more we're likely to value the search engine [

Meta Tags: Some search engines are now indexing Web documents by the meta tags in the documents' HTML (at the beginning of the document in the so-called "head" tag). What this means is that the Web page author can have some influence over which keywords are used to index the document, and even in the description of the document that appears when it comes up as a search engine hit. There is a lot of conflicting information out there on meta-tagging. If you're confused it may be because different search engines look at meta tags in different ways.

Some rely heavily on meta tags, others don't use them at all. The general opinion seems to be that meta tags are less useful than they were a few years ago, largely because of the high rate of spamdexing (web authors using false and misleading keywords in the meta tags). It seems to be generally agreed that the "title" and the "description" meta tags are important to write effectively, since several major search engines use them in their indices. Use relevant keywords in title, and vary the titles on the different pages that make up website, in order to target as many keywords as possible. As for the "description" meta tag, some search engines will use it as their short summary of URL, so make sure description is one that will entice surfers to site. Many search engine algorithms score the words that appear towards the top of your document more highly than the words that appear towards the bottom. Words that appear in HTML header tags (H1, H2, H3, etc) are also given more weight by some search engines [47].

Concept-based searching: Unlike keyword search systems, concept-based search systems try to determine what you mean, not just what you say. In the best circumstances, a concept-based search returns hits on documents that are "about" the subject/theme you're exploring, even if the words in the document don't precisely match the words you enter into the query. There are various methods of building clustering systems, some of which are highly complex, relying on sophisticated linguistic and artificial intelligence theory that we won't even attempt to go into here. Excite is used to a numerical approach. Excite's software determines meaning by calculating the frequency with which certain important words that appear. When several words or phrases that are tagged to signal a particular concept appear close to each other in a text, the search engine concludes, by statistical analysis that the piece is "about" a certain subject. For example, the word heart, when used in the medical/health context, would be likely to appear with such words as coronary, artery, lung, stroke, cholesterol, pump, blood, attack, and arteriosclerosis. If the word heart appears in a document with others words such as flowers, candy, love, passion, and valentine, a very different context is established, and a concept-oriented search engine returns hits on the subject of romance .

Search Engines Components

Basically, a search engine is a software program that searches for sites based on the words that you designate as search terms. Search engines look through their own databases of information in order to find what it is that you are looking for [52].

Search engines are the key to finding specific information on the vast expanse of the WWW. Without sophisticated search engines, it would be virtually impossible to locate anything on the Web without knowing a specific URL .

Search engines are not simple. They include incredibly detailed processes and methodologies, and are updated all the time. All search engines go by this basic process when conducting search processes, but because there are differences in search engines, there are bound to be different results depending on which engine you use.

1. The searcher types a query into a search engine.
2. Search engine software quickly sorts through literally millions of pages in its database to find matches to this query.
3. The search engine's results are ranked in order of relevancy .

Same search on different search engines produce different results because not all indices are going to be exactly the same. It depends on what the spiders find or what the humans submitted. But more important, not every search engine uses the same algorithm to search through the indices. The algorithm is what the search engines use to determine the relevance of the information in the index to what the user is searching for [53]. Search engines have three major elements. First is the spider, also called the crawler. The spider visits a web page, reads it, and then follows links to other pages within the site. This is what it means when someone refers to a site being "spidered" or "crawled." The spider returns to the site on a regular basis, such as every month or two, to look for changes. Everything the spider finds goes into the second part of the search engine, the index. The index, sometimes called the catalog, is like a giant book containing a copy of every web page that the spider finds. If a web page changes, then this book is updated with new information. Sometimes it can take a while for new pages or changes that the spider finds to be added to the index. Thus, a web page may have been "spidered" but not yet "indexed." Until it is indexed -- added to the index -- it is not available to those searching with the search engine. Search engine software is the third part of a search engine. This is the program that sifts through the millions of pages recorded in the index to find matches to a search and rank them in order of what it believes is most relevant [44]. The figure (2-2) shows how The Web is crawled and placed into a local repository where it is indexed and retrieved when using a search engine.

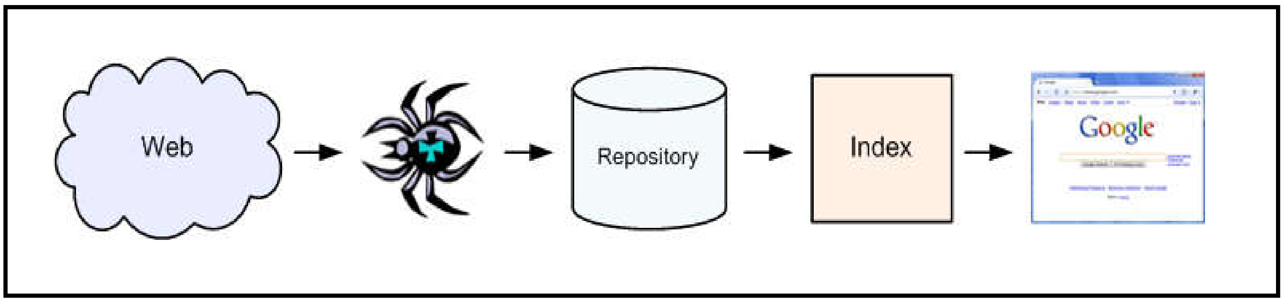


Figure (2-2) Crawling the Web, indexing, and retrieved when using a search engine.

So, in the third part there is a query processor which has several parts, including the user interface (search box), the "engine" that evaluates queries and matches them to relevant documents, and the results formatter. We can see that in the following figure [54]:

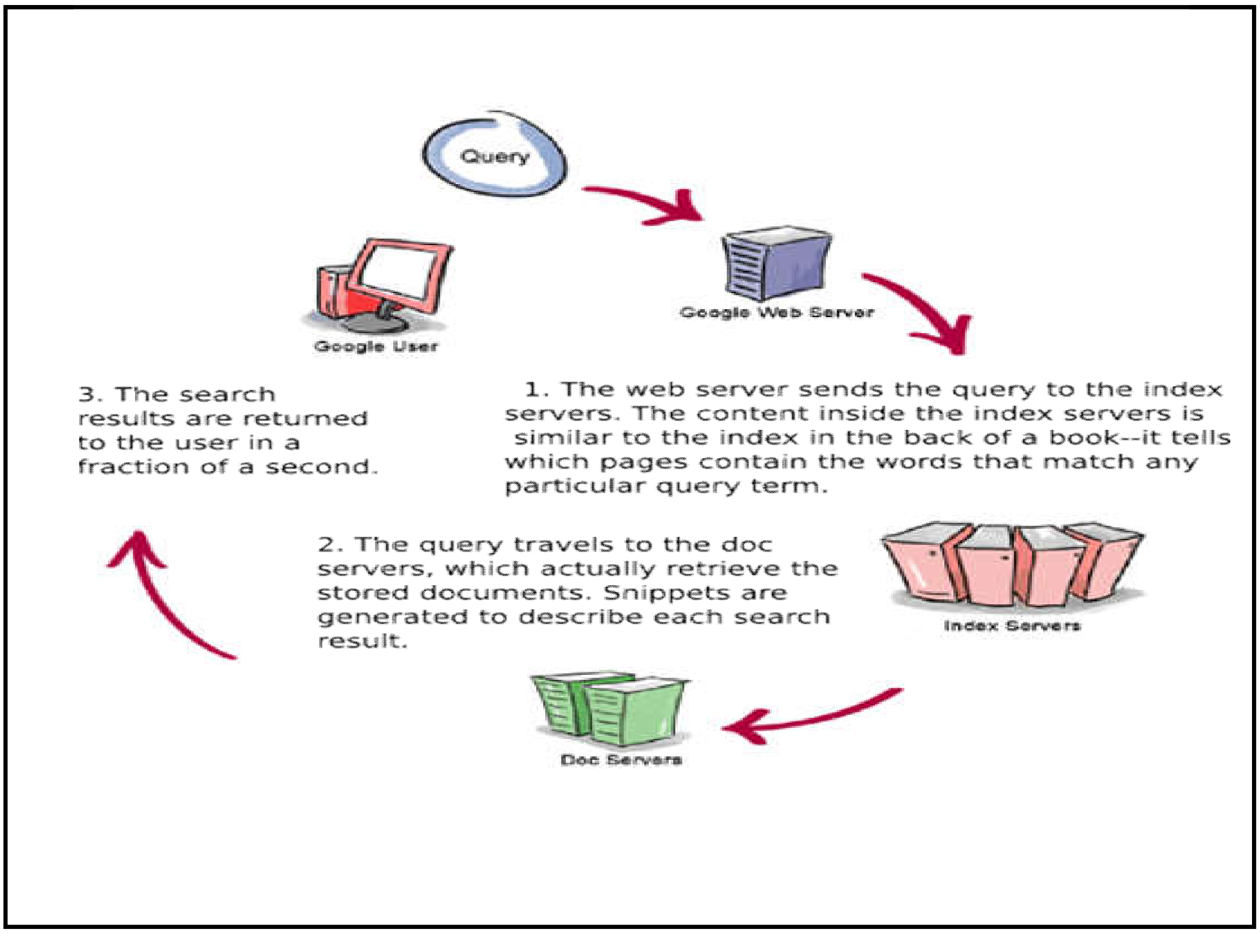

Query
Google Web Server
Google User
1. The web server sends the query to the index servers. The content inside the index servers is similar to the index in the back of a book--it tells which pages contain the words that match any particular query term.
3. The search results are returned to the user in a fraction of a second.
2. The query travels to the doc servers, which actually retrieve the stored documents. Snippets are generated to describe each search result.
Index Servers
Doc Servers

Figure (2-3) Query processing

- *Web Crawler*

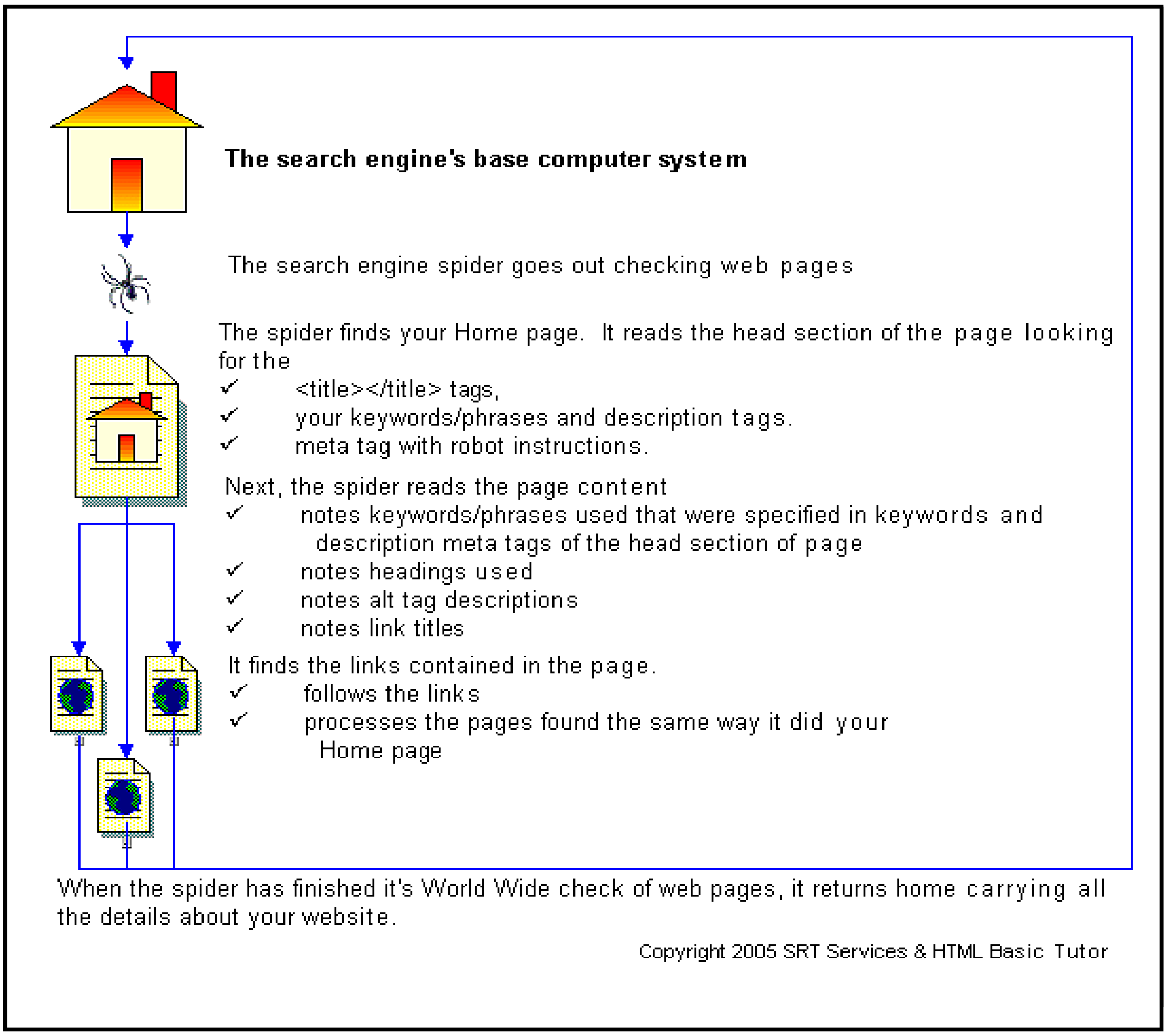
Web crawlers are programs created by search engines that go around the Internet and create an index of all collected information. The index allows search engines to take queries from users and show all matching pages. Web crawlers are also known as search engine spiders or robots .

This process is called Web crawling or spidering. Many legitimate sites, in particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine, which will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML codes. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses (usually for spam). A Web crawler is one type of bot, or software agent. In general, it starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. A typical web crawler starts by parsing a specified web page: noting any hypertext links on that page that point to other web pages. The Crawler then parses those pages for new links recursively. A crawler is a software or script or automated program which resides on a single machine. The crawler simply sends HTTP requests for documents to other machines on the Internet, just as a web browser does when the user clicks on links. All the crawler really does is to automate the process of following links .

The web crawler sets out from the search engine's base computer system looking for websites to index. Crawler collects information about the website and its links.

- the website URL
- the web page title
- the meta tag information
- the web page content
- the links on the page, and where they go to

When the web crawler returns home the information is indexed by the search engine. Figure (2-4) illustrates how web crawler work :



The search engine's base computer system
The search engine spider goes out checking web pages
The spider finds your Home page. It reads the head section of the page looking for the
✓ <title></title> tags,
✓ your keywords/phrases and description tags.
✓ meta tag with robot instructions.
Next, the spider reads the page content
✓ notes keywords/phrases used that were specified in keywords and description meta tags of the head section of page
✓ notes headings used
✓ notes alt tag descriptions
✓ notes link titles
It finds the links contained in the page.
✓ follows the links
✓ processes the pages found the same way it did your Home page
When the spider has finished it's World Wide check of web pages, it returns home carrying all the details about your website.
Copyright 2005 SRT Services & HTML Basic Tutor

Figure (2-4) How web crawler works

When one webpage redirects to another web page, sometimes indexes the page content under the URL of the entry or "source" page, and sometimes indexes it under the URL of the final, destination, or "target" page [58].

Web crawlers are a central part of search engines, and details on their algorithms and architecture are kept as business secrets. Crawler architecture is shown in the figure (2-5) :

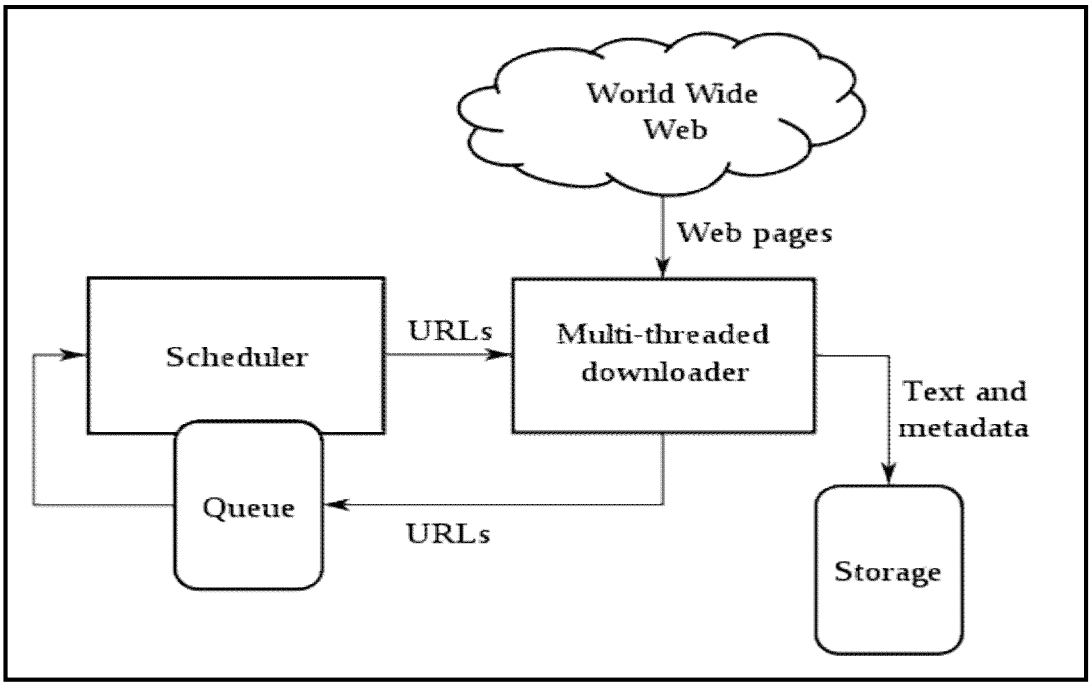


Figure (2-5) The architecture of web crawler

There are two important characteristics of the Web that generate a scenario in which Web crawling is very difficult:

1. Large volume of Web pages.
2. Rate of change on web pages.

Thus the implementer of a web crawler must be defined below in terms of the mentioned strategies:

- Selecting the better algorithm to decide which page to download.
- Strategizing how to re-visit pages to check for updates.
- Strategizing how to avoid overloading websites.

While selecting the search algorithm for the web crawler an implementer should keep in mind that algorithm must make sure that web pages are chosen depending upon their importance. The importance of a web page lies in its popularity in terms of links or visits, or even its URL. The crawler downloads as many resources as possible from a particular Web site. That way a crawler would ascend to every path in each URL that it intends to crawl. For example, when given a seed URL of http://foo.org/a/b/page.html, it will attempt to crawl /a/b/, /a/, and /.This is called (Path-ascending crawling) or (linked-based crawler). The advantage with Path-ascending crawler is that they are very effective in finding isolated resources, or resources for which no inbound link would have been found in regular crawling. The optimum method is to re-visit the web and maintain average high freshness of web page is to ignore the pages that change too often. The re-visiting methods considered here regard all pages as homogeneous in terms of quality ("all pages on the Web are worth the same"), something that is not a realistic scenario. Crawlers can retrieve data much quicker and in greater depth than human searchers, so they can have a crippling impact on the performance of a site. Needless to say if a single crawler is performing multiple requests per second and/or downloading large files, a server would have a hard time keeping up with requests from multiple crawlers .

Thus, the job of the crawler is twofold. Its main job is to get a page from a server that makes some kind of summary of the page, and stores the summary in a large database. The other job the robot does is to look at the page to find other pages that can be visited by the robot and put them in a list of pages to be visited. The latter job can often be done at the same time as the summarizing. A summary of a web-page as created by the robot can be seen as meta-data [9].

- The Indexer

The Search Indexer is the application which reads the text of the documents to be searched and stores them in an efficient searchable form usually called the index. Web site indexers must be able to save files in a web server directory, so that the search engine can locate it when a site visitor wants to search it. Remote search engines store the index files on their server, where it is used by the search engine when the user starts searching [60].

A search engine’s index is similar to the index in the back of a book: it is used to find the pages on which a word occurs. There are two main differences: the search engine’s index lists every occurrence of every word, not just the important concepts, and the number of pages is in the billions, not hundreds [61]. The search engine indexing process takes the detailed information collected by the search engine spider (web crawler) and analyses the information. The information the search engine spider found on each page is analyzed, building a list of the words and phrases within the document taking into account:

- Number of times a word/phrase is used on the web page.
- Weight of the word/phrase.

The weight of the word/phrase increases in value depending on where it is located.

- top of document
- sub headings
- text links
- meta tags
- title

Each search engine has a different way of analyzing these information therefore different results in the search engines is the result. A web page may do well in the search engine results on one search engine and poorly on another. The indexed information is saved in a database, waiting for someone to do a search. When someone does a search at the search engine, the words they entered in the search box are compared with the indexed information in the search engine's database and the list of web pages it feels are most relevant to the search are returned. Figure (2-6) below shows Indexing Process [62].

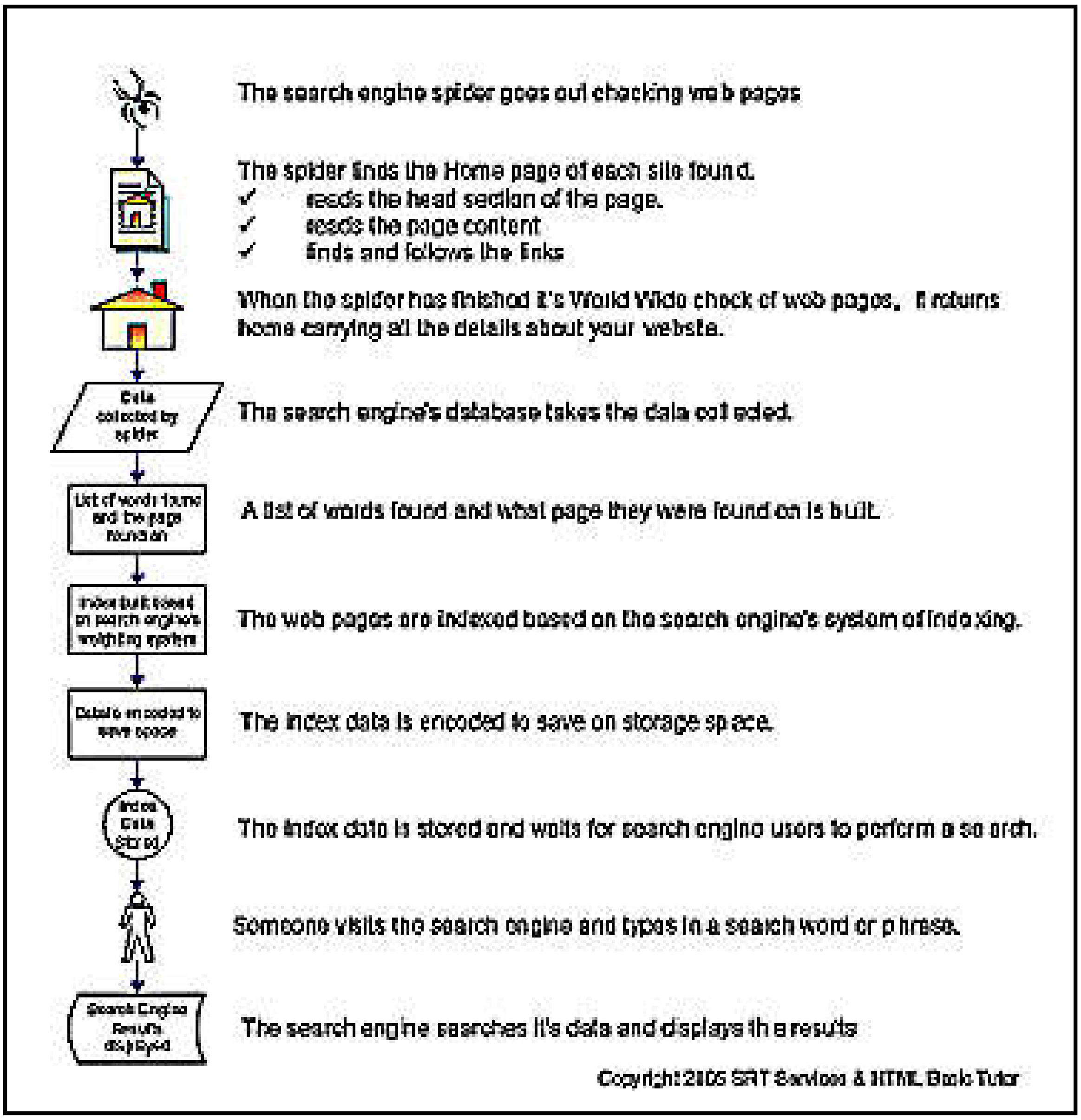

The search engine spider goes out checking web pages
The spider finds the Home page of each site found.
✓ reads the head section of the page.
✓ reads the page content
✓ finds and follows the links
When the spider has finished it's World Wide check of web pages, it returns home carrying all the details about your website.
Data collected by spider
The search engine's database takes the data collected.
List of words found and the page found on
A list of words found and what page they were found on is built.
Index built based on search engine's weighting system
The web pages are indexed based on the search engine's system of indexing.
Data is encoded to save space
The index data is encoded to save on storage space.
Index Data Stored
The index data is stored and waits for search engine users to perform a search.
Someone visits the search engine and types in a search word or phrase.
Search Engine Results displayed
The search engine searches it's data and displays the results
Copyright 2005 SRT Services & HTML Basic Tutor

Figure (2-6) Indexing process

Various techniques of compression and clever representation are used to keep the index “small,” but it is still measured in terabytes (millions of megabytes), which again means that distributed computing is required. Most modern search engines index link data as well as word data. It is useful to know how many pages link to a given page, and what are the qualities of those pages. This kind of analysis is similar to citation analysis in bibliographic work, and helps establish which pages are authoritative [61].

- Index Formats

There are two formats for building the index:

1. Inverted Index

For search systems to service requests efficiently, they must have rapid access to candidate results. An inverted index is a data structure that provides direct access to the locations at which terms appear within a document collection. Akin to the index found in a text book, an inverted index records, for each term in the collection, the documents in which the term occurs. Inverted indexes are an essential component of all web search engines and text retrieval systems [Zobel and Moffat, 2006]. An inverted index has two components: first, a vocabulary of the terms that occur in the searchable collection, and second, a set of postings that list the locations of occurrence of the search terms. Typically, searchable terms are words extracted from the text [Williams and Zobel, 2005] [12].

Because the inverted index stores a list of the documents containing each word, the search engine can use direct access to find the documents associated with each word in the query in order to retrieve the matching documents quickly. The following table is a simplified illustration of an inverted index:

Table (2-4) simplified illustration of an Inverted Index

Word	Documents
the	Document 1, Document 3, Document 4, Document 5
cow	Document 2, Document 3, Document 4
says	Document 5
moo	Document 7

This index can only determine whether a word exists within a particular document, since it stores no information regarding the frequency and position of the word; it is therefore considered to be a boolean index. Such an index determines which documents match a query but does not rank matched documents. In some designs the index includes additional information such as the frequency of each word in each document or the positions of a word in each document. Position information enables the search algorithm to identify word proximity to support searching for phrases; frequency can be used to help in ranking the relevance of documents to the query. Such topics are the central research focus of information retrieval.

The inverted index is a sparse matrix, since not all words are present in each document. To reduce computer storage memory requirements, it is stored differently from a two dimensional array. The index is similar to the term document matrices employed by latent semantic analysis. The inverted index can be considered a form of a hash table. In some cases the index is a form of a binary tree, which requires additional storage but may reduce the lookup time. In larger indices, the architecture is typically a distributed hash table [63].

2. Forward Index

The forward index stores a list of words for each document. The following table is a simplified form of the forward index:

Table (2-5) simplified illustration of a Forward Index

Document	Words
Document 1	the,cow,says,moo
Document 2	the,cat,and,the,hat
Document 3	the,dish,ran,away,with,the,spoon

The rationale behind developing a forward index is that as documents are parsing, it is better to immediately store the words per document. The delineation enables Asynchronous system processing, which partially circumvents the inverted index update bottleneck. The forward index is sorted to transform it to an inverted index. The forward index is essentially a list of pairs consisting of a document and a word, collated by the document. Converting the forward index to an inverted index is only a matter of sorting the pairs by the words. In this regard, the inverted index is a word-sorted forward index [63].

- How Search Engines Process Documents Before Indexing

Search engine indexing collects, parses, and stores data to facilitate fast and accurate information retrieval. Index design incorporates interdisciplinary concepts from linguistics, cognitive psychology, mathematics, informatics, physics and computer science [63].

More detail – there are linguistic preprocessing of documents before indexing occurs. The main concepts and issues involved:

1. Determining Document Encoding and Getting a Character Sequence

The first thing needs to be aware of is the fact that all we have is a byte stream. Don't necessarily know what kind of document have just fetched. Of course for a complex search engine that needs to deal with multiple document formats, before doing anything else, we need *to* determine the encoding of the document. To do this in order to correctly turn our byte stream into a character stream. Alternatively a document may be a binary format (e.g. PDF, or MS Word document) where getting the character stream is quite a bit more involved. Even when this is not the case, most documents will contain spurious data which we don't want to deal with (e.g. html or xml tags) and will therefore need to strip from our document. Having done all of that we eventually end up with a character sequence that we can process further [64].

2. Tokenizing the Document

Computers do not 'know' that a space character separates words in a document. Instead, humans must program the computer to identify what constitutes an individual or distinct word, referred to as a token. Such a program is commonly called a tokenizer or parser or lexer. Many search engines, as well as other natural language processing software, incorporate specialized programs for parsing. During tokenization, the parser identifies sequences of characters which represent words and other elements, such as punctuation, which are represented by numeric codes, some of which are non-printing control characters [1, 63].

3. Removing Stop Words

However in practice many optimizations to index processing have been proposed. One method is to reduce the average number of posting lists processed per query. A well-known simple technique to do this is stopping [Salton and McGill, 1983], that is, ignoring the long lists of function terms such as “the”, “of”, and “therefore”.
Stopping can be applied at either index construction time, or at query evaluation time. When applied at index construction time, the stopped terms are not stored in the index, significantly reducing the index size. When applied at query evaluation time, stopped terms are not retrieved from the index to be processed during the similarity calculation, and query processing time is significantly reduced [1, 12].

4. Token Normalization

The next thing to be done with tokens is to normalize them. This means turning each token into its canonical form. Doing this in a couple of ways:

- Equivalence classing – this is an implicit process that is based on a set of rules and heuristics. For example the canonical form of a word with a hyphen is the same word without one and we will therefore create an equivalence class between those words (e.g. co-operation, cooperation -> cooperation*)*. These equivalence classes will be applied to the document terms during indexing as well as to the queries at runtime so that a query that contains one of the words in the equivalence class will return documents that contain any of the words.
- Synonym lists – this is a more explicit and labour intensive process, but can give more control. Synonym lists are constructed by hand (e.g. think – chair, stool, seat, throne). These synonym lists can then be applied in one of two ways. At indexing time you can index the same document under all the terms that are synonyms of each other, this requires a bit more space, but at query time any of the synonym terms will match the document. Also applying the synonym lists at runtime whereby expand the query to contain more terms (i.e. all the synonyms) which means the query takes a bit longer to execute.

There are other normalization steps that can be applied to terms, such as case folding, whereby lower-case all terms. This will of course need to be applied to the query as well, but can come back to bite us, especially where proper names are concerned. For example Crunchy Nut Cornflakes is a breakfast serial, crunchy nut cornflakes are just three random words. Despite this, case folding is normally used by most search engines [12, 64].

5. Stemming and Lemmatization

Another optimization approach is to reduce the average number of lists processed per query is stemming [Frakes, 1992]. Typically, English stemming iteratively removes suffixes from words to find the morphological root, allowing plurals to match singulars and so on. For example, stemming reduces "reducing", "reduced", and "reduction" to the common root "reduce-". Using this approach, all words that stem to "reduce-" are indexed as postings in one list, with the overall effect that the list is typically much longer than the list of any of the unstemmed variants. These improve compression — since the differences between adjacent values are smaller — and fewer disk seeks (but longer reads) are required on average to answer each query. From a search perspective, the process permits a query such as "reducing words" to match a document containing the terms "word" and "reduction".

As with stopping, effectiveness varies between tasks. Stemming is generally applied at index construction time, and reduces the size of the on-disk index as less terms are stored in the vocabulary [12].

In the other words,

- Stemming basically concerns itself with chopping off suffixes using various heuristics to find the more of less correct base form of the word. There are several well known stemming algorithms. The Lovins algorithm is one of the earliest stemmers and the Porter stemmer is one of the best known. Stemming tends to increase recall while harming precision**.**
- Lemmatization is a more complex process using morphological analysis and vocabularies to arrive at the base form [1, 64].

2.9 Ranking

Every search engine strives to return the relevant web pages that will satisfy the user requests. Each search engine uses a proprietary (ranking algorithm) that attempts to instantly build a list of highly appropriate responses to the given query [65].

Ranking is the heart of the search engine. In order to produce a good search engine, we need to know how to rank pages properly for the result documents. There is not much information available about this in the public. Today, most search engines use variations of the Boolean or vector model to do ranking. Recall that search engines do not allow access to the text, but only the indices, because it is too expensive in terms of time and space. So, when searching, ranking must use indices while not accessing the text. Besides that, there are also other difficulties as well. There might be too many relevant pages for a simple query. Also, it is difficult to compare two search engines, because of their continuous improvement [8].

Since users usually look only at the very first pages returned by a Web search engine, it is very important to effectively rank the results returned for the submitted queries. The two main techniques used in ranking algorithms for Web pages are: *Statistical* (Content-based ranking algorithm) force on words occurrence, keyword ratios, word relevance content, page title, domain, meta tags in the pages), and *Link Based* (link-based or link-sensitive ranking algorithm) based on importance inferred from informations on the structure of the Web graph (i.e. hyperlink structure among web pages) [10, 66].

2.9.1 Link analysis

Links connecting pages are a key componentnof the web. Links are a powerful navigational aid for people browsing the Web, but they also help search engines understand the relationships between the pages. These detected relationships help search engines rank web pages more effectively. A link in a web page is encoded in HTML with a statement such as:

For more information on this topic, please go to <a herf="http://www:somewhere.com"> the somewhere page</a>.

when this page appears in your web browser, the words "the somewhere page" will be displayed differently than regular text, usually underlined or in a different color(or both). When you click on that link, your browser will then load the web page http://www.somewhere.com. In this link, "the somewhere page" is called the *anchor text,* and http://www.somewhere.com is the *destination.* Both component are useful in the ranking process [1].

2.9.2 Anchor text

Anchor text has two properties that make it particulary useful for ranking web pages. First, it tend to be very short, perhaps two or three words, and those words often succinctly describe the topic of the linked page. For instance, links so www.ebay.com are highly likely to contain the word "eBay" in the anchor text. Many queries are very similar to anchor text in that they are also short topical descriptions of web pages. This suggests a very simple algorithm for ranking pages: search through all links in the collection, looking for anchor text that is an exact match for the users query. Each time there is aa match, add 1 to the score of the destination page. Pages would then be ranked in decreasing order of this score. This algorithm has some glaring faults, not the least of which is how to handle the query "click here". More generally, the collection of all the anchor text in the links pointing to a page can be used as an additional text field for that page, and incorporated into the ranking algorithm. Anchor text is usually written by people who are not the authors of the destination page. This means that the anchor text can describe a destination page from a different prespective, or emphasize the most important aspect of the page from a community viewpoint. The fact that the link exists at all is a vote of importance for the destination page. Although anchor text is not mentioned as often as link analysis algorithms (for example, PageRank) in discussions of web search engines, TRACE evaluations have shown that it is the most important part of the representation of a page for some types of web search. In particular, it is essential for searches where the user is trying to find home page for a particular topic, person, or organization [1].

2.9.3 Link-Based Ranking system

Its also called (Query-independent ranking) that aims to measure the intrinsic quality of a page. To this end, a score is assigned to each page independent of a specific user query [67]. In uses the link-structure of the Web to enhance search results of term-based search engines. This is done by concedering the potential hyper-information contained in each Web-page: the information that can be found when following hyperlinks which originate in the page. Brin and Page devised PageRank while desgining the ranking module of the prototype of Google, which has since become a highly popular commercial search engine [68].

1. PageRank

There are tens of billions of web pages, but most of them are not very interesting. The huge size of the Web makes this a difficult problem for search engines. How can the search engine choose the most popular(and probably the correct) one? One very effective approach is to use the links between web pages as a way to measure popularity. The most obvious measure is to count the number of *inlinks* (links pointing to a page) for each page and use this as a feature or piece of evidence in the ranking algorithm. Although this has been shown to be quite effective, it is very susceptible to spam. Mesures based on link analysis algorithms are designed to provide more reliable ratings of web pages. Of these measures, PageRank, which is associated with the Google search engine, is most often mentioned [1].

The original PageRank algorithm was described by Lawrence Page and Sergey Brin in 1998 [11]. Page Rank is based on the idea of *random surfer*(the PageRank calculation corresponds to finding what is known as the stationary probability distribution of a *random walk* on the graph of the Web. A random walk is a special case of a *Markov chain* in which the next state (the next page visited) depends solely on the current state (current page). The transitions that are allowed between states are all equally probable and are given by the links).

Suppose for the moment that the Web consists of just three pages, A, B, and C. We will suppose that page A links to pages B and C, page B links to page C, and page C links to page A, as shown in the Figure (2-7). The PageRank of page C, which is the probability of this page, will depend on the PageRank of pages A and B. If starting in page A, there is a 50% chance for going to page C (because there are two outgoing links). Another way of saying this is that the PageRank for a page is divided evenly between all the outgoing links.

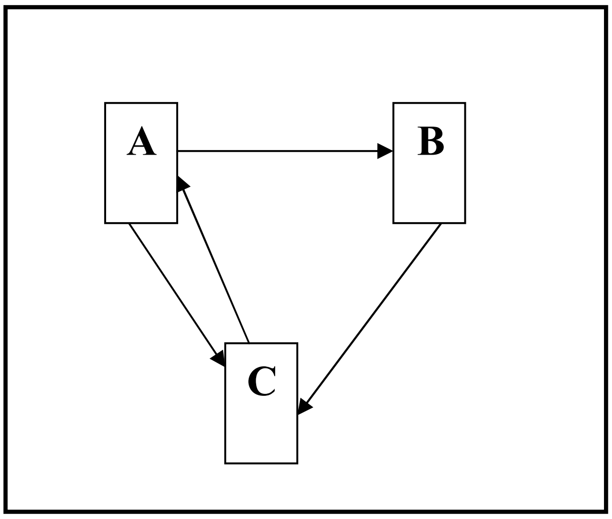


Fig.(2-7) A sample "Internet" consisting of just three web pages. The arrows denote links between the pages.

In the first step, assume that the PageRank is the same for each page, so assignning a PageRank of 1 to each page. PageRank is defined by Page and Sergey like the following [5]:

$$PR(A) = (1\text{-}d) + d * \left(\frac{PR(T1)}{C(T1)} + \ldots\ldots + \frac{PR(T_n)}{C(T_n)} \right) \qquad \ldots\ldots\ldots\ (2\text{-}1)$$

Where:

PR(A): is the PageRank of the page A.

PR(Ti): is the PageRank of the page Ti which links to page A.

C(Ti): is the number of outbound links on page Ti.

D: is the damping factor which can be set between 0 and 1 (often d assigned at 0.85).

The sum of weighted PageRank of all pages Ti is multiplied by the damping factor d. Thereby, the extending of PageRank benefit for a page by another page linking to it will be reduced [11].

To summarize, PageRank is an important example of query-independent meta-data that can improve ranking for web search. Web pages have the same PageRank value regardless of what query is being processed. Search engines that use PageRank will prefer pages with high PageRank values instead of assuming that all web pages are equally likely to satisfy a query [1].

2. Improved PageRank Algorithm[11]

The ranker system consists of two subsystems, the link-based ranker subsystem and the term-based ranker subsystem. In this section, the link-based ranker will be presented. This subsystem is working in off-line time; it depends on the hyperlink structure of the web page. Several modifications are proposed to achieve a high quality ranking score by implementing the improved PageRank algorithm on the web pages. These are:

a. Additional Factors Influencing PageRank.

The original PageRank calculations are based on the number of inbound links and outbound links only. But by adding the additional factor to the original PageRank equation (2-1) the improved PageRank will achieve more accurate values. Therefore, the following modification of the PageRank algorithm is assumed [11]:

$$\boxed{PR(A)= (1-d)+d*[(\frac{PR(T_1)}{C(T_1)})* L(T_1,A)+...+(\frac{PR(T_n)}{C(T_n)})*L(T_n,A)]....(2\text{-}2)}$$

Where, L(Ti,A) represents the evaluation of a (inbound or outbound) link which points from page Ti to page A. L(Ti,A) may consist of several factors, each of which has to be determined only once. Each link has three importance factors that are extracted during the crawler running phase these factors are:

1. Visibility of a link which represented by Y in equation (2-4).
2. Position of a link within a web page which is represented by X in equation (2-4).
3. Distance between the web pages which is represented by Kd in equation (2-3).

To implement the evaluation of the linking page into PageRank, the evaluation factor of the modified algorithm must consist of several single factors. For a link that points from page Ti to page A, it can be given as follows [11]:

$$L(T_i, A) = K(T_i, A) * Kd(T_i) \qquad (2\text{-}3)$$

Where Kd(Ti) represents the distance between web pages, and K(Ti,A) is the weighting of a single link within a page by its visibility or position, it can be given as follows:

$$K(T_i, A) = \left(\frac{X(T_i, A) * Y(T_i, A)}{Z(T_i)}\right) \qquad (2\text{-}4)$$

Where X(Ti,A) represents the position of a link within a document. X(Ti,A) equals 1 if the link is on the lower one third of the page, 2 if the link is on the middle one third of the page, and 3 if the link is on the upper one third of the page. Y(Ti,A) represents the visibility of a link. Y(Ti,A) equals 1 if a link is not particulary emphasized, and 3 if the link is, for instance, bold or italic. Z(Ti) is the summation of the multiplicative correlation between X and Y, it can given as follows:

$$Z(T_i) = \sum_{k \neq i} (X(T_i, T_k) * Y(T_i, T_k)) \qquad (2\text{-}5)$$

Those factors reflect the probability for the random surfer clicking on a link on a specific web page. In the original PageRank algorithm, this probability is given by the term (1/C(Ti)), where by the probability is equal for each link on one page. Assigning different probabilities to each link on a page can, for instance, be realized as follows:

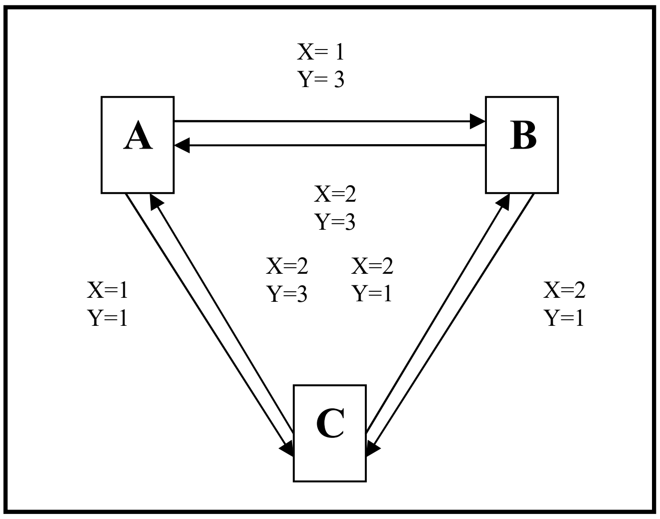


Fig (2-8) Web Pages with its Link Attributes

Consider a web consisting of three pages A, B, and C, where each of these pages has outbound links to both of other pages as in figure (2-8). The links are weighted by two evaluation factor X and Y. The multiplicative correlation between X and Y for the links in this example are evaluated as follows:

X(A,B)*Y(A,B)=1*3=3

X(A,C)*Y(A,C)=1*1=1

X(B,A)*Y(B,A)=2*3=6

X(B,C)*Y(B,C)=2*1=2

X(C,A)*Y(C,A)=2*3=6

X(C,B)*Y(C,B)=2*1=2

For the purpose of determining the single factors L, the evaluated links must not simply be weighted by the number of outbound links on one page, but in fact by the total of evaluated links on the page. The following weighting are quotients Z(Ti) for the single pages Ti:

$Z(A)=X(A,B)*Y(A,B)+X(A,C)*Y(A,C)=4$

$Z(B)=X(B,A)*Y(B,A)+X(B,C)*Y(B,C)=8$

$Z(C)=X(C,A)*Y(C,A)+X(C,B)*Y(C,B)=8$

The result of each Z(Ti)is stored in Hyperlink Table as will be presented in table (4-12). The evaluation factors K(T1,T2) for a link which is pointing from page T1 to page T2 are hence given by the equation (2-4). Their values regarding presented example are as follows:

$K(A,B)=0.75$

$K(A,C)=0.25$

$K(B,A)=0.75$

$K(B,C)=0.25$

$K(C,A)=0.75$

$K(C,B)=0.25$

If all pages A, B, and C are in the same site, the evaluation factors L(T1,T2) for a link which is pointing from page T1 to T2 are given by the equation (2-3). Since Kd(Ti) for all i is equal to (1), the value of L(T1,T2) will be equal to K(T1,T2). At a damping factor d of 0.85, the following equations for the calculation of PageRank values will be obtained:

$PR(A)=0.15+0.85*(0.375*PR(B)+0.375*PR(C))$

$PR(B)=0.15+0.85*(0.375*PR(A)+0.125*PR(C))$

$PR(C)=0.15+0.85*(0.125*PR(A)+0.125*PR(B))$

Solving these equations gives the following PageRank values for the presented example:

$PR(A)=819/693$

$PR(B)=721/693$

$PR(C)=539/693$

First of all, it is obvious that page A has the highest PageRank of all three pages, this is caused by page A receiving the relatively higher evaluated link from page B as well as from page C. Furthermore, even by the evaluation of single links, the sum of the PageRank values of all pages equals 3 (2079/693) and thereby the total number of pages.

b. Using most recent PageRank value of each web page.

The original PageRank equation (2-1) is implemented in several itrations to be as most as converge. So, in the first iteration all pages rank value is equal to one as an initial value, then in the second iteration pages' rank value at the first iteration is used as an input to the next iteration and so on. Instead of using $PR(t)_{i-1}$ (of i-1th iteration) as input to PageRank equation in the ith iteration, the most resent PageRank can be used as it is clear in this equation [11]:

$$PR(Pi)=(1-d)+d*[\sum_{J<i}\frac{PR(Pj)_{l+1}}{C(Pj)}+\sum_{j\geq i}\frac{PR(Pj)_{l}}{C(Pj)}] \text{ ....(2-6)}$$

By implementing these modifications on the original PageRank equation, the proposed PageRank equation will be as follows:

$$PR(Pi)=(1-d)+d*[\sum_{J<i}(\frac{PR(Pj)_{l+1}}{C(Pj)}*L(Pj,Pi))+\sum_{j\geq i}(\frac{PR(Pj)_{l}}{C(Pj)}*L(Pj,Pi))] \text{....(2-7)}$$

Where l is the iteration number.

Finally, the Improved Link-Based Ranker algorithm is as follows[11]:

```
Algorithm Name: Link-Based Ranking Algorithm
Input: webpage, and hyperlink
Output: PageRank for each webpages
Begin
  while no_error
    for all Pi in web retrieve_factors(Pi)
      for all Pj points to pi
        $PR(Pi)=(1-d)+d*[\sum_{J<i}(\frac{PR(Pj)_{l+1}}{C(Pj)}*L(Pj,Pi))+\sum_{j\geq i}(\frac{PR(Pj)_l}{C(Pj)}*L(Pj,Pi))]$
        err=abs[PR(Pi)l-PR(Pi)l+1]
        if err>0.000001 then
          no_error=false
        else no_error =true
  end [while]
End of Algorithm
```

Figure(2-9) Link-Based Ranking Algorithm

2.9.4 Keyword-Based Ranking Systems

Most traditional algorithms are keyword-based, also called (In query-dependent ranking), an algorithm assigns a score that measures the quality and relevance of a selected set of pages to a given user query. Use word frequencies, word importance, document length and other statistical cues to assign potential importance to a document. However, with the emergence of the web many new algorithms for web search have been proposed and are being used in various web search engines today. Many of these algorithms incorporate link-structure of pages in their ranking schemes, and are notably different from the traditional keyword-based document ranking algorithms [67].

1. Boolean Spreading Activation Algorithm

This algorithm is proposed by Yuwono and Lee in 1996. it is based on the Boolean retrieval model, where retrieval is based solely on the occurrence or absence of keywords in the documents without considering term frequencies. In this algorithm the Boolean model is extended so that documents can be ranked based on the number of the query words they contain. This strategy can be considered as a simple Fuzzy set retrieval model in contrast to the rigorous set membership of Boolean retrieval model. More formally, document *i* is assigned a relevence score, *R(i,q)*, with respect to the query q as follows [11]:

$$R(i,q) = \sum_{j=1}^{M} C(i,j) \qquad \text{....................} \quad (2\text{-}8)$$

where:

R(i,j): the relevence score of i-th page with respect to the query q

M: number of words in the query

C*(i,j)*: occurrence of the j-th query in the i-th page,

Where:

$$C(i,j) = \begin{cases} 1 \text{ if } P_i \text{ contains } Q_j \\ 0 \text{ otherwise} \end{cases}$$

Notice that all terms frequency is not used in the formula. It is assumed that the query does not contain any disjunctions or negations. Disjunctions can be removed by normalization or splitting the query into separate conjunctive clauses. Negations can be removed by disqualifying all documents containing the negated terms prior to the ranking [11].

2. TFxIDF Algorithm

The TFxIDF algorithm is based on the well-known vector space model, which typically uses the cosine of the angle between the document and the query vectors in a multi-dimensional space as the similarity measure. Vector length normalization can be applied when computing the relevance score, Ri,q, of page Pi with respect to query q [69]:

$$R_{i,q} = \frac{\sum_{term_j \in q}(0.5 + 0.5\frac{TF_{i,j}}{TFmax_i})IDF_j}{\sqrt{\sum_{term_j \in P_i}(0.5 + 0.5\frac{TF_{i,j}}{TFmax_i})^2 (IDF_j)^2}} \quad \ldots\ldots (2\text{-}9)$$

where:

$TF_{i,j}$	:	the term frequency of Q_j in P_i
$TF_{i,max}$	:	the maximum term frequency of a keyword in P_i
IDF_j	:	$\log(N/\sum_{i=1}^{N} C_{i,j})$

Generally speaking, the relevance score of document is a sum of the weights of the query terms that appear in the document, normalized by the Euclidean vector length of the document. The weight of a term is a function of a word's occurrence frequency (also called term frequency) in the document and the number of documents containing the word in the collection (i.e., the inverse document frequency). This weighting function gives higher weights to terms which occur frecuently in a small set of documents.

The full vector space model is very expensive to implement, because the normalization factor is very expensive to compute. In TFxIDF algorithm, the normalization factor is not used. That is the relevance score is computed by [69]:

$$R_{i,q} = \sum_{term_j \in q} (0.5 + 0.5 \frac{TF_{i,j}}{TF_{i,max}}) IDF_j \qquad \text{...... (2-10)}$$

The vector-length normalization does not work well for short documents. This is consistent with testing queries, if the average length in the test collection is only 48,00 words (not including stop words and other words removed during the indexing process). It may be case that vector-length normalization does not work for documents where the size of segment with a coherent topic is small, e.g., a few sentences.

By using the standered evaluation procedure to compute the average interpolated recall/precision for TFxIDF algorithm, it is decided that this algorithm is not the best one but is better than “Boolean Spreading Activation” algorithm [11].

3. Vector Spreading Activation Algorithm

This algorithm is also proposed by Yuwono and Lee in 1996. It combines the vector space model and spreading activation model. Each document is first assigned a relevance score using TFxIDF algorithm, and then the score of a document is propagated to the documents it references. More formally, the algorithm assigns a relevance score to page *Pi* with respect to query *q* as follows [12]:

$$R_{i,q} = S_{i,q} + \sum_{j=1, j \neq i}^{N} \alpha Li_{i,j} \cdot S_{j,q} \qquad \text{........ (2-11)}$$

Where:

N: the number of WWW pages in the index database.

R(i,q): the relevance score of i-th page respect to the query *q*

S(i,q): the TFxIDF score of the i-th page

α: is a constant link weight where (0<α<1).

Through an experiment the value of α is identified in 0.2. The using of recall/precision curves of Vector Spreading Activation on specific testing queries with α parameter value of 0.0 (as TFxIDF without the spreading activation effect), 0.1, 0.2, 0.3, 0.4 and 0.5 proves that the best retrieval performence is achieved with α equals 0.2.

By using the standard evaluation procedure to compute the average interpolated recall/precision for Vector Spreading Activation algorithm, it becomes obvious that this algorithm has the best retrieval performence, followed by TFxIDF, Boolean Spreading Activation [11].

2.9.5 Improved Keyword-Based Ranking Systems

Term-Based Ranking process can be improved by using two algorithms to compute page's ranking, TFxIDF and Vector Spreading Activation algorithms. TFxIDF is enhanced by supplementing it with the attribute for each word occurrence. Then the obtained number from the enhanced TFxIDF is considered as a factor in Vector Spreading Activation equation (2-11)[11].

Important factors are retrieved from the Search Engine Indices by the searcher. One of them is page_id that for each page_id the relevancy is calculated depending on the occurrence of query terms by improved TFxIDF relevancy equation[11]:

$$\boxed{S(i,q) = \sum_{\text{termj} \in q} \left(\left(0.5 + 0.5 \cdot \frac{TF(i,j)}{TFmaxi} \right) \cdot IDFj \cdot attributej \right) \quad .... (2\text{-}12)}$$

The term **attribute** represents the summation of the following factors:

1. the first factor that is used in computing the attribute is word importance multiplied by the related word position:

F1 = importance * norpos
Where norpos = 1- (position / nows)

The absolute number that is used in representing a word position is a critical number because it depends greatly on the number of words (nows) in the entire page. For example, suppose there are two pages, the first one has 100 words as a content and the second has 300 words as a content and in both of them the word “rank” occures at position 90, it will be clear that the position in the first page is approximately at the end of the page, its relative position is:

norpos = 1 – (90/100) = 0.1

But the second page the position 90 is approximately at the end of the first one third of the page. Its relative posision is:

norpos = 1 – (90/300) = 0.7

From these two results, it is obvious that 0.7>0.1 that means the accurate word position depends on the page length (in words). Also, the word position is effected by its importance as presented in table (4-6). So in the first page, if the word “rank” occurs as a hypertext (link text), its importance will be equal to 6:

F1 = 6* 0.1= 0.6

And the second page, if the “rank” occurs as a normal text its importance will be equal to 2:

F1 = 2*0.7 = 1.4

2. The second factor represents the color factor: if the word's font color is matched with global page font color the value of F2 will be equal to 1, otherwise it will be equal to 2.
3. The third factor represents the font face factor: if the word's font face is matched with global page font face the value of F3 will be equal to 1, otherwise it will be equal to 2.

4. The fourth factor represents the style of each word, each style has its own value as presented in table (4-7).
5. And the last one represents the size of each word.

Finally, the attribute will be equal to:

$$\text{Attribute} = \sum F_k \quad \text{..............................} \quad (2\text{-}13)$$

Then the term is multiplied to the single itrated term. Then the result that yielded from the improved TFxIDF is considered as an input to the Vector Spreading Activation.

The improved Term-Based Ranking algorithm will be as follows[11]:

```
Algorithm Name: Term-Based Ranking Algorithm
Input : Set of attribute
Output: ranked webpages
Begin
   [1]  for all pages in Retrieved_pages_list
          for all terms in Query
            for each occurance of termj in pagei
              attributej = term_attribute(factors)
              $S(i,q) = \sum_{termj \in q} \left( \left(0.5 + 0.5 \cdot \frac{TF(i,j)}{TFmaxi}\right) \cdot IDFj \cdot attributej \right)$
              $R(i,q) = S(i,q) + \sum_{J=1, j \neq i}^{N} \alpha \cdot Li(i,j) \cdot S(i,q)$
        go to [1]
End of Algorithm
```

Figure (2-10) Term-Based Ranking Algorithm