PANIMALAR INSTITUTE OF TECHNOLOGY

DEPARTMENT OF CSE

R-2017

ACADEMIC YEAR	:2023-2024
BATCH	:2020-2024
YEAR/SEM	:IV/VIII
SUBJECTCODE/TITLE	:CS8080/INFORMATION RETRIEVAL TECHNIQUES

UNIT-4

WEB RETRIEVAL AND WEB CRAWLING

UNIT IV
WEB RETRIEVAL AND WEB CRAWLING

The Web – Search Engine Architectures – Cluster based Architecture – Distributed Architectures – Search Engine Ranking – Link based Ranking – Simple Ranking Functions – Learning to Rank – Evaluations -- Search Engine Ranking – Search Engine User Interaction – Browsing – Applications of a Web Crawler – Taxonomy – Architecture and Implementation – Scheduling Algorithms – Evaluation.

The Web

World Wide Web, which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to local computers through the internet. These websites contain text pages, digital images, audios, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc. The WWW, along with internet, enables the retrieval and display of text and media to your device.

The building blocks of the Web are web pages which are formatted in HTML and connected by links called "hypertext" or hyperlinks and accessed by HTTP. These links are electronic connections that link related pieces of information so that users can access the desired information quickly. Hypertext offers the advantage to select a word or phrase from text and thus to access other pages that provide additional information related to that word or phrase.

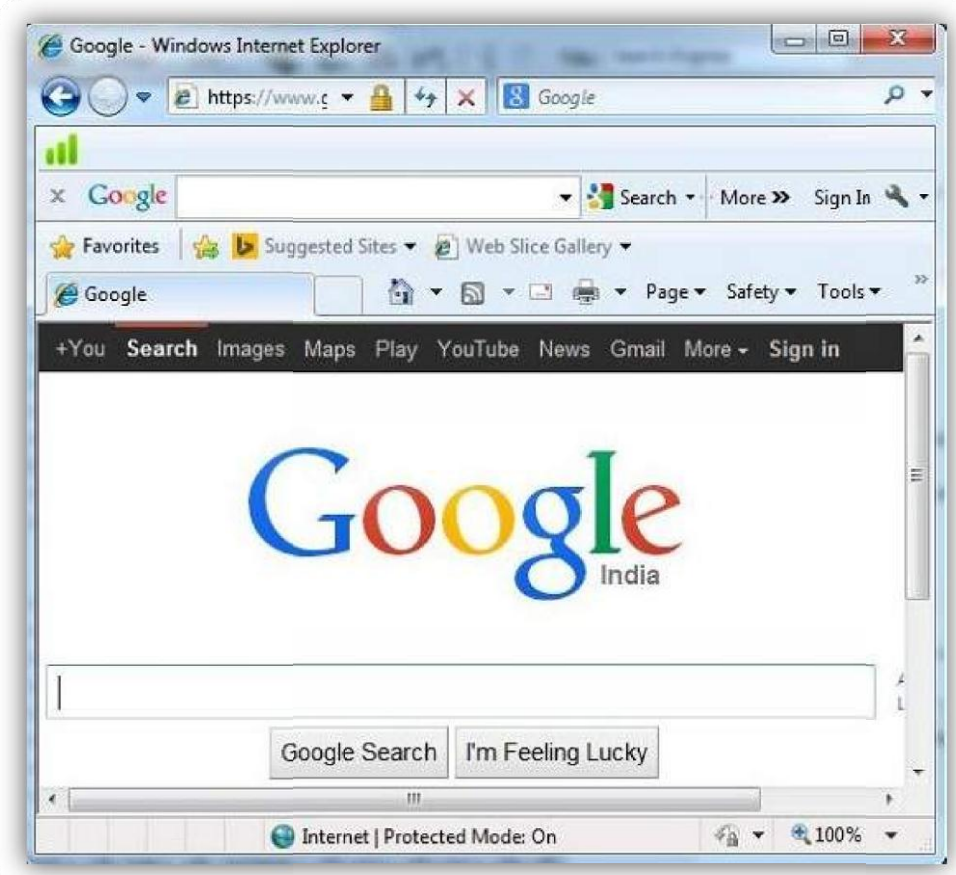
A web page is given an online address called a Uniform Resource Locator (URL). A particular collection of web pages that belong to a specific URL is called a website, e.g., *www.facebook.com*, *www.google.com*, etc. So, the World Wide Web is like a huge electronic book whose pages are stored on multiple servers across the world.

SEARCH ENGINE ARCHITECTURES

Components Of A Search Engine

Search Engine refers to a huge database of internet resources such as web pages, newsgroups, programs, images etc. It helps to locate information on World Wide Web.

User can search for any information by passing query in form of keywords or phrase. It then searches for relevant information in its database and return to the user.



Search Engine Components

Generally there are three basic components of a search engine as listed below:

1. Web Crawler
2. Database
3. Search Interfaces

Web crawler

It is also known as **spider** or **bots.** It is a software component that traverses the web to gather information.

Database

All the information on the web is stored in database. It consists of huge web resources.

Search Interfaces

This component is an interface between user and the database. It helps the user to search through the database.

Search Engine Working

Web crawler, database and the search interface are the major component of a search engine that actually makes search engine to work. Search engines make use of Boolean expression AND, OR, NOT to restrict and widen the results of a search. Following are the steps that are performed by the search engine:

- The search engine looks for the keyword in the index for predefined database instead of going directly to the web to search for the keyword.
- It then uses software to search for the information in the database. This software component is known as web crawler.
- Once web crawler finds the pages, the search engine then shows the relevant web pages as a result. These retrieved web pages generally include title of page, size of text portion, first several sentences etc.

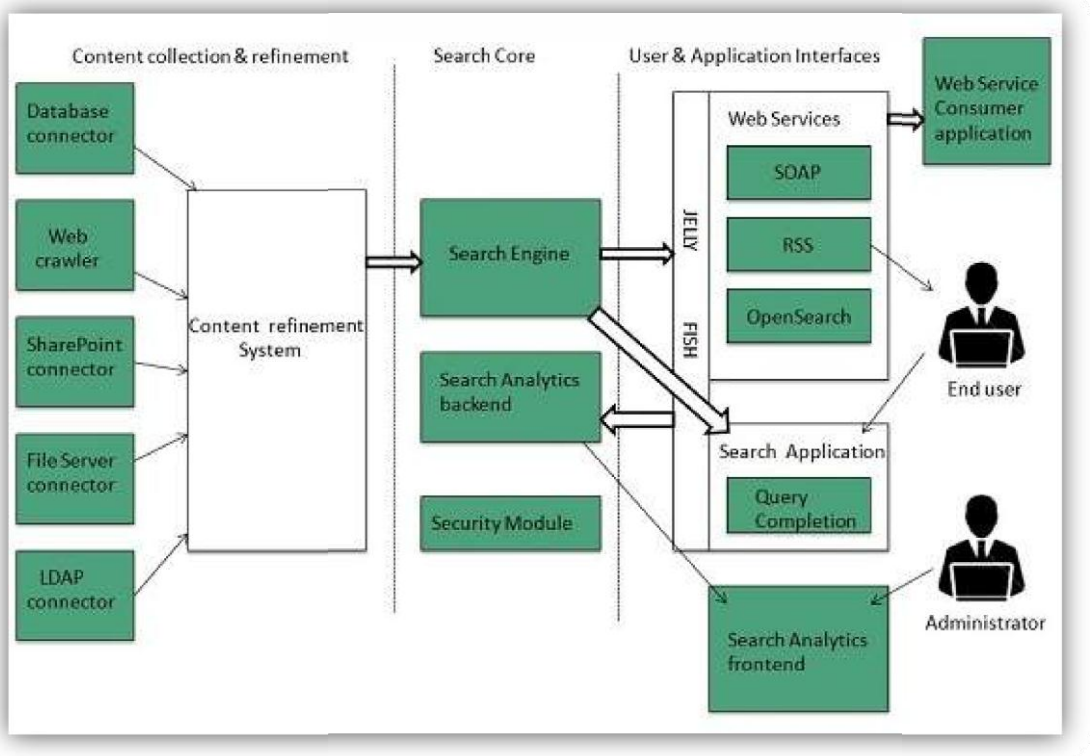
These search criteria may vary from one search engine to the other. The retrieved information is ranked according to various factors such as frequency of keywords, relevancy of information, links etc.

- User can click on any of the search results to open it.

Architecture

The search engine architecture comprises of the three basic layers listed below:

- Content collection and refinement.
- Search core
- User and application interfaces



Search Engine Processing

Indexing Process

Indexing process comprises of the following three tasks:

- Text acquisition
- Text transformation
- Index creation

Text acquisition

It identifies and stores documents for indexing.

Text Transformation

It transforms document into index terms or features.

Index Creation

It takes index terms created by text transformations and create data structures to support fast searching.

Query Process

Query process comprises of the following three tasks:

- User interaction
- Ranking
- Evaluation

User interaction

It supports creation and refinement of user query and displays the results.

Ranking

It uses query and indexes to create ranked list of documents.

Evaluation

It monitors and measures the effectiveness and efficiency. It is done offline.

CLUSTER BASED ARCHITECTURE

- ✓ The task for the retrieval system is to match the query against clusters of documents instead of individual documents, and rank clusters based on their similarity to the query.
- ✓ Any document from a cluster that is ranked higher is considered more likely to be relevant than any document from a cluster ranked lower on the list.
- ✓ This is in contrast to most other cluster search methods that use clusters primarily as a tool to identify a subset of documents that are likely to be relevant, so that at the time of retrieval, only those documents will be matched to the query.
- ✓ This approach has been the most common for cluster-based retrieval.

- ✓ The second approach to cluster-based retrieval is to use clusters as a form of document smoothing.
- ✓ Previous studies have suggested that by grouping documents into clusters, differences between representations of individual documents are, in effect, smoothed out.

Current search engines use a massive parallel and cluster-based architecture. Due to the large size of the document collection, the inverted index does not fit in a single computer and must be distributed across the computers of a cluster. The large volume of queries implies that the basic architecture must be replicated in order to handle the overall query load, and that each cluster must handle a subset of the query load. In addition, as queries originate from all around the world and Internet latency is appreciable across continents, cluster replicas are maintained in different geographical locations to decrease answer time. This allows search engines to be fault-tolerant in most typical worst-case scenarios, such as power outages or natural disasters.

There are many crucial details to be carefully addressed in this type of architecture:

1. It is particularly important to achieve a good balance between the internal (answering queries and indexing) and external (crawling) activities of the search engine. This is achieved by assigning dedicated clusters to crawling, to document serving, to indexing, to user interaction, to query processing, and even to the generation of the result pages.
2. In addition, a good load balancing among the different clusters needs to be maintained. This is achieved by specialized servers called (quite trivially) load balancers.
3. Finally, since hardware breaks often, fault tolerance is handled at the software level. Queries are routed to the most adequate available cluster and CPUs and disks are routinely replaced upon failure, using inexpensive exchangeable hardware components.

Figure 11.7 shows a generic search cluster architecture with its key components.

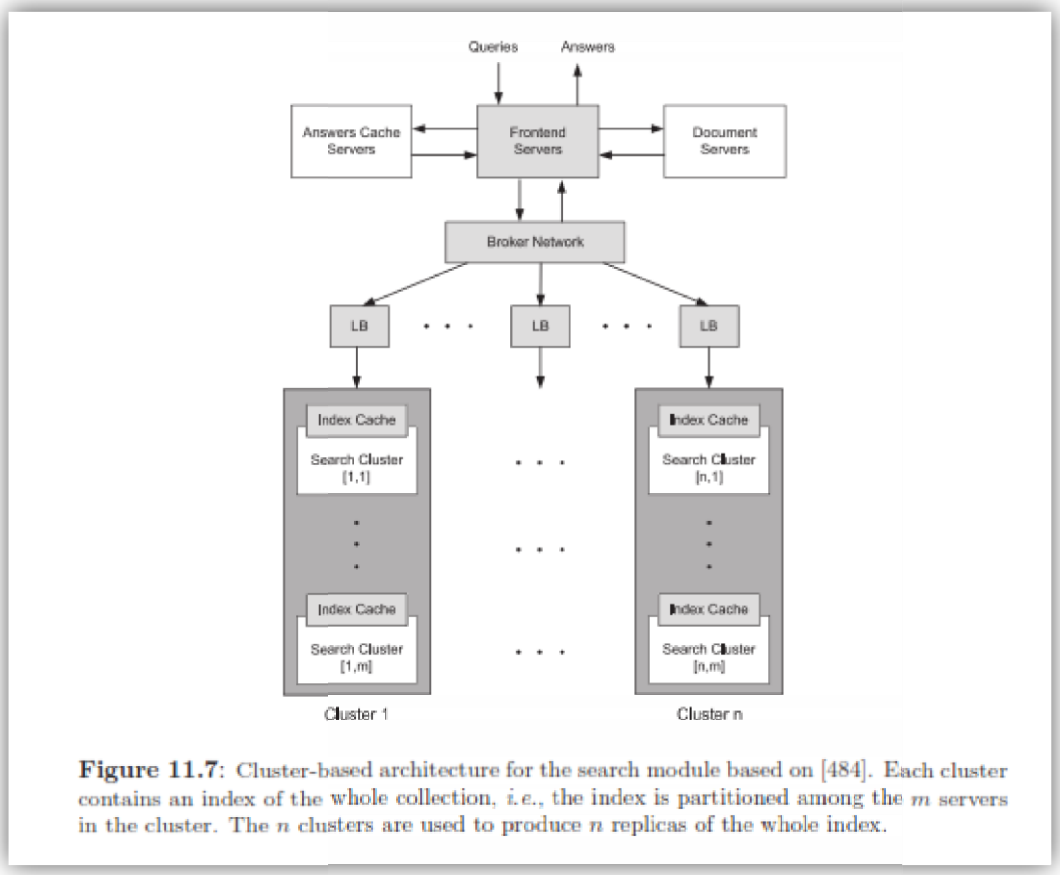


Figure 11.7: Cluster-based architecture for the search module based on [484]. Each cluster contains an index of the whole collection, *i.e.*, the index is partitioned among the m servers in the cluster. The n clusters are used to produce n replicas of the whole index.

The front-end servers receive queries and process them right away if the answer is already in the "answer cache" servers. Otherwise they route the query to the search clusters through a hierarchical broker network. The exact topology of this network can vary but basically, it should be designed to balance traffic so as to reach the search clusters as fast as possible. Each search cluster includes a load balancing server (LB in the figure) that routes the query to all the servers in one replica of the search cluster. In this figure, we show an index partitioned into n clusters with m replicas. Although partitioning the index into a single cluster is conceivable, it is not recommended as the cluster would turn out to be very large and consequently suffer from additional management and fault tolerance problems.

Each search cluster also includes an index cache, which is depicted at the top, as a flat rectangle. The broker network merges the results coming from the

search clusters and sends the merged results to the appropriate front-end server that will use the right document servers to generate the full results pages, including snippet and other search result page artifacts. This is an example of a more general trend to consider a whole data center as a computer.

DISTRIBUTED ARCHITECTURES

There exist several variants of the crawler-indexer architecture and we describe here the most important ones. Among them, the most significant early example is Harvest.

Harvest

Harvest uses a distributed architecture to gather and distribute data, which is more efficient than the standard Web crawler architecture. The main drawback is that Harvest requires the coordination of several Web servers. Interestingly, the Harvest distributed approach does not suffer from some of the common problems of the crawler-indexer architecture, such as:

- increased servers load caused by the reception of simultaneous requests from different crawlers,
- increased Web traffic, due to crawlers retrieving entire objects, while most content is not retained eventually, and
- lack of coordination between engines, as information is gathered independently by each crawler.

Avoiding these issues was achieved by introducing two main components in the architecture: gatherers and brokers. A gatherer collects and extracts indexing information from one or more Web servers. Gathering times are defined by the system and are periodic (i.e., there are harvesting times as the name of the system suggests). A broker provides the indexing mechanism and the query interface to the data gathered. Brokers retrieve information from one or more gatherers or other brokers, updating incrementally their indexes. Depending on the configuration of gatherers and brokers, different improvements on server load and network traffic

can be achieved. For example, a gatherer can run on a Web server, generating no external traffic for that server. Also, a gatherer can send information to several brokers, avoiding work repetition. Brokers can also filter information and send it to other brokers. This design allows the sharing of work and information in a very flexible and generic manner. An example of the Harvest architecture is shown in Figure 11.9.

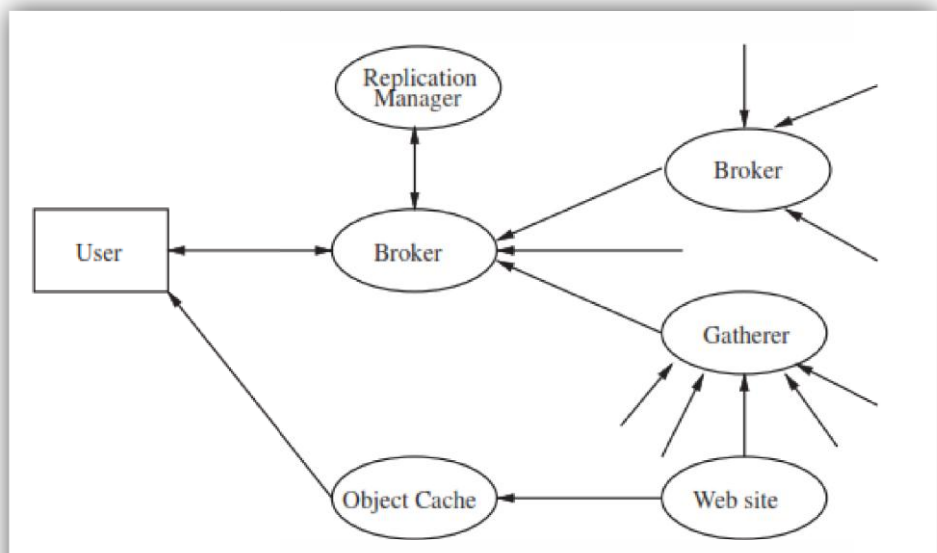


Figure 11.9: Harvest architecture.

One of the goals of Harvest is to build topic-specific brokers, focusing the index contents and avoiding many of the vocabulary and scaling problems of generic indexes. Harvest includes a dedicated broker that allows other brokers to register information about gatherers and brokers. This is mostly useful for identifying an appropriate broker or gatherer when building a new system. The Harvest architecture also provides replicators and object caches. A replicator can be used to replicate servers, enhancing user-base scalability. For example, the registration broker can be replicated in different geographic regions to allow faster access. Replication can also be used to divide the gathering process among many Web servers. Finally, the object cache reduces network and server load, as well as response latency when accessing Web pages.

SEARCH ENGINE RANKING

Ranking is the hardest and most important function search engines have to execute. A first challenge is to devise an adequate evaluation process that allows judging the efficacy of a ranking, in terms of its relevance to the users. Without such evaluation process it is close to impossible to fine tune the ranking function, which basically prevents achieving high quality results. There are many possible evaluation techniques and measures. We cover this topic in the context of the Web, paying particular attention to the exploitation of user's clicks.

A second critical challenge is the identification of quality content in the Web. Evidence of quality can be indicated by several signals such as domain names (i.e., .edu is a positive signal as content originating from academic institutions is more likely to be reviewed), text content and various counts (such as the number of word occurrences), links (like Page Rank), and Web page access patterns as monitored by the search engine. Indeed, as mentioned before, clicks are a key element of quality. The more traffic a search engine has, the more signals it will have available. Additional useful signals are provided by the layout of the Web page, its title, metadata, font sizes, as discussed later.

The economic incentives of the current advertising based business model adopted by search engines have created a third challenge, avoiding Web spam. Spammers in the context of the Web are malicious users who try to trick search engines by artificially inflating the signals mentioned in the previous paragraph. This can be done, for instance, by repeating a term in a page a great number of times, using link farms, hiding terms from the users but keeping them visible to the search engines through weird coloring tricks, or even for the most sophisticated ones, deceiving Javascript code. Finally, the fourth issue lies in defining the ranking function and computing it (which is different from evaluating its quality as mentioned above). While it is fairly difficult to compare different search engines as they evolve and operate on different Web corpora, leading search engines have to

constantly measure and compare themselves, each one using its own measure, so as to remain competitive.

Ranking Signals

We distinguish among different types of signals used for ranking improvements according to their origin, namely content, structure, or usage, as follows. Content signals are related to the text itself, to the distributions of words in the documents as has been traditionally studied in IR. The signal in this case can vary from simple word counts to a full IR score such as BM25. They can also be provided by the layout, that is, the HTML source, ranging from simple format indicators (more weight given to titles/headings) to sophisticated ones such as the proximity of certain tags in the page. Structural signals are intrinsic to the linked structure of the Web. Some of them are textual in nature, such as anchor text, which describe in very brief form the content of the target Web page. In fact, anchor text is usually used as surrogate text of the linked Web page. That implies that Web pages can be found by searching the anchor texts associated with links that point to them, even if they have not been crawled. Other signals pertain to the links themselves, such as the number of in-links to or outlinks from a page.

The next set of signals comes from Web usage. The main one is the implicit feedback of the user through clicks. In our case the main use of clicks are the ones in the URLs of the results set.

LINK-BASED RANKING

Given that there might be thousands or even millions of pages available for any given query, the problem of ranking those pages to generate a short list is probably one of the key problems of Web IR; one that requires some kind of relevance estimation. In this context, the number of hyperlinks that point to a page provides a measure of its popularity and quality. Further, many links in common among pages and pages referenced by a same page are often indicative of page relations with potential value for ranking purposes. Next, we present several

examples of ranking techniques that exploit links, but differ on whether they are query dependent or not.

Early Algorithms

- TF-IDF
- Boolean spread, vector spread, and most-cited
- WebQuery

HITS

A better idea is due to Kleinberg and used in HITS (Hypertext Induced Topic Search). This ranking scheme is query-dependent and considers the set of pages S that point to or are pointed by pages in the answer. Pages that have many links pointing to it in S are called authorities because they are susceptible to contain authoritative and thus, relevant content. Pages that have many outgoing links are called hubs and are susceptible to point to relevant similar content. A positive two-way feedback exists: better authority pages come from incoming edges from good hubs and better hub pages come from outgoing edges to good authorities. Let H(b) and A(p) be the hub and authority values of page p. These values are defined such that the following equations are satisfied for all pages p:

$$H(p) = \sum_{u \in S \mid p \to u} A(u)\ , \qquad A(p) = \sum_{v \in S \mid v \to p} H(v) \qquad (11.5)$$

where H(p) and A(p) for all pages are normalized (in the original paper, the sum of the squares of each measure is set to one). These values can be determined through an iterative algorithm, and they converge to the principal eigenvector of the link matrix of S. In the case of the Web, to avoid an explosion on the size of S, a maximal number of pages pointing to the answer can be defined. This technique does not work with non-existent, repeated, or automatically generated links. One solution is to weigh each link based on the surrounding content. A second problem is that of topic diffusion, because as a consequence of link weights, the result set might include pages that are not directly related to the query (even if they have got

high hub and authority values). A typical case of this phenomenon is when a particular query is expanded to a more general topic that properly contains the original answer. One solution to this problem is to associate a score with the content of each page, like in traditional IR ranking, and combine this score with the link weight. The link weight and the page score can be included in the previous formula multiplying each term of the summation. Experiments show that the recall and precision for the first ten results increase significantly. The appearance order of the links on the Web page can also be used by dividing the links into subgroups and using the HITS algorithm on those subgroups instead of the original Web pages. In Table 11.2, we show the exponent of the power law of the distribution for authority and hub values for different countries of the globe adapted from.

Country	PageRank	HITS	
		Hubs	Auth.
Brazil	1.83	2.9	1.83
Chile	1.85	2.7	1.85
Greece	1.83	2.6	1.83
South Korea	1.83	3.7	1.83
Spain	1.96	n/a	n/a

Table 11.2: Summary of power-law exponents for link-based measures of various countries.

SIMPLE RANKING FUNCTIONS

The simplest ranking scheme consists of using a global ranking function such as PageRank. In that case, the quality of a Web page is independent of the query and the query only acts as a document filter. That is, for all Web pages that satisfy a query, rank them using their PageRank order.

A more elaborated ranking scheme consists of using a linear combination of different relevance signals. For example, combining textual features, say BM25, and link features, such as PageRank. To illustrate, consider the pages p that satisfy query Q. Then, the rank score R(p, Q) of page p with regard to query Q can be computed as:

$$R(p,Q) = \alpha\ BM25(p,Q)\ +\ (1-\alpha)PR(p) \qquad (11.7)$$

Further, R(p, Q)=0 if p does not satisfies Q. If we assume that all the functions are normalized and a ∈ [0, 1], then R(p, Q) ∈ [0, 1]. Notice that this linear function is convex in a. Also, while the first term depends on the query, the second term does not. If a = 1, we have a pure textual ranking, which was the typical case in the early search engines. If a = 0, we have a pure link-based ranking that is also independent of the query. Thus, the order of the pages is known in advance for pages that do contain q. We can tune the value of a experimentally using labeled data as ground truth or click through data. In fact, a might even be query dependent. For example, for navigational queries a could be made smaller than for informational queries.

Early work on combining text-based and link-based rankings was published by Silva. The authors used a Bayesian network to combine the different signals and showed that the combination leads to far better results than those produced by any of the combining ranking functions in isolation. Subsequent research by Calado discussed the efficacy of a global link-based ranking versus a local link based ranking for computing Web results. The local link-based ranking for a page p is computed considering only the pages that link to and are linked by page p. The authors compare results produced by a combination of a text-based ranking (Vector model) with global HITS, local HITS, and global PageRank, to conclude that a global link-based ranking produces better results at the top of the ranking, while a local link-based ranking produces better results later in the ranking.

LEARNING TO RANK

A rather distinct approach for computing a Web ranking is to apply machine learning techniques for learning to rank. For this, one can use their favorite machine learning algorithm, fed with training data that contains ranking information, to “learn” a ranking of the results, analogously to the supervised

algorithms for text classification. The loss function to minimize in this case is the number of mistakes done by the learned algorithm, which is similar to counting the number of misclassified instances in traditional classification. The evaluation of the learned ranking must be done with another data set (which also includes ranking information) distinct from the one used for training. There exist three types of ranking information for a query Q, that can be used for training:

- point wise: a set of relevant pages for Q.
- Pair wise: a set of pairs of relevant pages indicating the ranking relation between the two pages.

  That is, the pair [p1>p2], implies that the page p1 is more relevant than p2.
- List-wise: a set of ordered relevant pages: p1>p2⋯ >pm.

In any case, we can consider that any page included in the ranking information is more relevant than a page without information, or we can maintain those cases undefined. Also, the ranking information does not need to be consistent (for example, in the pair wise case). The training data may come from the so-called "editorial judgments" made by people or, better, from click through data. Given that users' clicks reflect preferences that agree in most cases with relevance judgments done by human assessors, one can consider using click through information to generate the training data. Then, we can learn the ranking function from click-based preferences. That is, if for query Q, p has more clicks than p2 , then [p1_ p21].

One approach for learning to rank from clicks using the pair wise approach is to use support vector machines (SVMs), to learn the ranking function. In this case, preference relations are transformed into inequalities among weighted term vectors representing the ranked documents. These inequalities are then translated into an SVM optimization problem, whose solution computes optimal weights for the document terms. This approach proposes the combination of different retrieval functions with different weights into a single ranking function.

The point wise approach solves the problem of ranking by means of regression or classification on single documents, while the pairwise approach transforms ranking into a problem of classification on document pairs. The advantage of these two approaches is that they can make use of existing results in regression and classification. However, ranking has intrinsic characteristics that cannot be always solved by the latter techniques. The list wise approach tackles the ranking problem directly, by adopting list wise loss functions, or directly optimizes IR evaluation measures such as average precision. However, this case is in general more complex. Some authors have proposed to use a multi-variant function, also called relational ranking function, to perform list wise ranking, instead of using a single-document based ranking function.

QUALITY EVALUATION

To be able to evaluate quality, Web search engines typically use human judgments that indicate which results are relevant for a given query, or some approximation of a "ground truth" inferred from user's clicks, or finally a combination of both, as follows.

Precision at 5, 10, 20

One simple approach to evaluate the quality of Web search results is to adapt the standard precision-recall metrics to the Web. For this, the following observations are important:

on the Web

it is almost impossible to measure recall, as the number of relevant pages for most typical queries is prohibitive and ultimately unknown. Thus, standard precision-recall figures cannot be applied directly.

Most Web users inspect only the top 10 results and it is relatively uncommon that a user inspects answers beyond the top 20 results. Thus, evaluating the quality of Web results beyond position 20 in the ranking is not indicated as does not reflect common user behavior. Since Web queries tend to be short and vague,

human evaluation of results should be based on distinct relevance assessments for each query-result pair. For instance, if three separate assessments are made for each query-result pair, we can consider that the result is indeed relevant to the query if at least two of the assessments suggest so. The compounding effect of these observations is that

(a) precision of Web results should be measured only at the top positions in the ranking, say P@5, P@10, and P@20 and

(b) each query-result pair should be subjected to 3-5 independent relevant assessments.

Click-through Data as an Evaluation Metric

One major advantage of using click through data to evaluate the quality of answers derives from its scalability. Its disadvantage is that it works less well in smaller corpora, such as countries with little Web presence, Intranet search, or simply in the long tail of queries. Note that users' clicks are not used as a binary signal but in significantly more complex ways such as considering whether the user remained a long time on the page it clicked (a good signal) or jumped from one result to the other (a signal that nothing satisfying was found). These measures and their usage are complex and kept secret by leading search engines.

Evaluating the Quality of Snippets

A related problem is to measure the quality of the snippets in the results. Search snippets are the small text excerpts associated with each result generated by a search engine. They provide a summary of the search result and indicate how it is related to the query (by presenting query terms in boldface, for instance). This provides great value to the users who can quickly inspect the snippets to decide which results are of interest.

Web Spam

The Web contains numerous profit-seeking ventures, so there is an economic incentive from Web site owners to rank high in the result lists of search engines.

All deceptive actions that try to increase the ranking of a page in search engines are generally referred to as Web spam or spamdexing (a portmanteau of “spamming” and “index”). The area of research that relates to spam fighting is called Adversarial Information Retrieval, which has been the object of several publications and workshops.

SEARCH ENGINE RANKING

Assigning Identifiers to Documents

Document identifiers are usually assigned randomly or according to the ordering with which URLs are crawled. Numerical identifiers are used to represent URLs in several data structures. In addition to inverted lists, they are also used to number nodes in Web graphs and to identify documents in search engines repositories.

It has been shown in the literature that a careful ordering of documents leads to an assignment of identifiers from which both index and Web graph storing methods can benefit. Also an assignment based on a global ranking scheme may simplify the ranking of answers (see section 11.5.3). Regarding the compression of inverted lists, a very effective mapping can be obtained by considering the sorted list of URLs referencing the Web documents of the collection. Assigning identifiers in ascending order of lexicographically sorted URLs improves the compression rate. The hypothesis that is empirically validated by Silvestri is that documents sharing correlated and discriminant terms are very likely to be hosted by the same site and will therefore also share a large prefix of their URLs. Experiments validate the hypothesis since compression rates can be improved up to 0.4 by using the URL sorting technique. Furthermore, sorting a few million URLs takes only tens of seconds and takes only a few hundreds megabytes of main memory.

search engine user interaction

Web search engines target hundreds of millions of users, most of which have very little technical background. As a consequence, the design of the interface has been heavily influenced by a extreme simplicity rule, as follows.

Extreme Simplicity Rule. The design of the user search experience, i.e., the patterns of user interaction with the search engine, must assume that the users have only minimal prior knowledge of the search task and must require as little learning on their part as possible. In fact, users are expected to read the "user manual" of a new refrigerator or DVD player more often than the help page of their favorite search engine. One immediate consequence of this state of affairs is that a user that does not "get it", while interacting with a search engine, is very likely to attempt to solve the problem by simply switching to another search engine. In this context, extreme simplicity has become the rule for user interaction in Web search.

In this section, we describe typical user interaction models for the most popular Web Search engines of today, their recent innovations, and the challenges they face to abide by this extreme simplicity rule. But we revisit them here in more depth, in the context of the Web search experience offered by major players such as Ask.com, Bing, Google and Yahoo! Search. We do not discuss here "vertical" search engines, i.e., search engines restricted to a specific domains of knowledge such as Yelp or Netflix, or major search engines verticals, such as Google Image Search or Yahoo! Answers.

The Search Rectangle Paradigm

Users are now accustomed with specifying their information needs by formulating queries in a search "rectangle". This interaction mode has become so popular that many Web homepages now feature a rectangle search box, visible in prominent area of the site, even if the supporting search technology is provided by a third partner. To illustrate, Figure 11.10 displays the search rectangle of the Ask, Bing, Google, and Yahoo! search engines. The rectangle design has remained

pretty much stable for some engines such as Google, whose main homepage has basically not changed in the last ten years. Others like Ask and Bing allow a more fantasy oriented design with colorful skins and beautiful photos of interesting places and objects (notice, for instance, the Golden Gate bridge background of Ask in Figure 11.10).

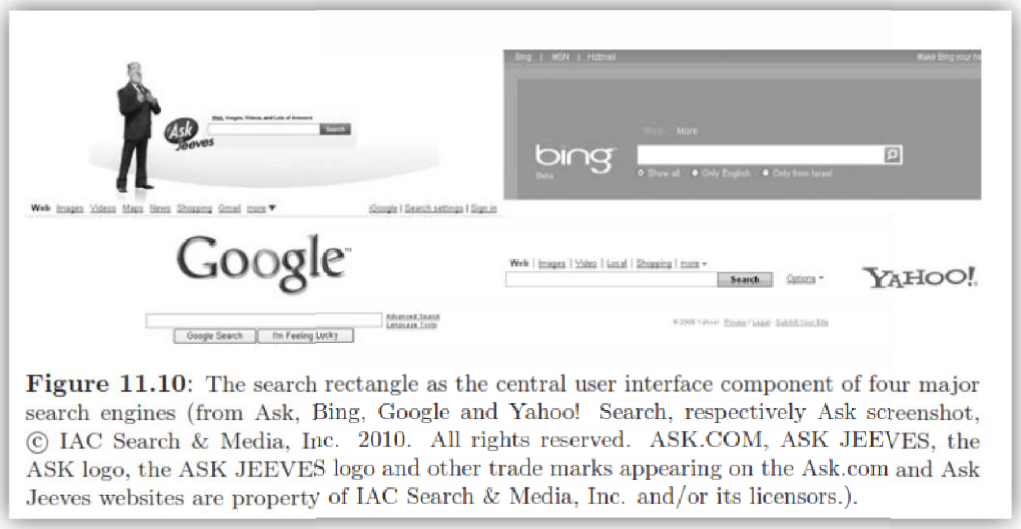


Figure 11.10: The search rectangle as the central user interface component of four major search engines (from Ask, Bing, Google and Yahoo! Search, respectively Ask screenshot, © IAC Search & Media, Inc. 2010. All rights reserved. ASK.COM, ASK JEEVES, the ASK logo, the ASK JEEVES logo and other trade marks appearing on the Ask.com and Ask Jeeves websites are property of IAC Search & Media, Inc. and/or its licensors.).

Despite these trends, the search rectangle remains the center piece of the action in all engines. While the display of a search rectangle at the center of the page is the favored layout style, there are alternatives:

Some Web portals embed the search rectangle in a privileged area of the homepage. Examples of this approach are provided by yahoo.com or aol.com.

• Many sites include an Advanced Search page, which provides the users with a form composed of multiple search "rectangles" and options (rarely used).

• The search toolbars provided by most search engines as a browser plug-in, or built-in in browsers like Firefox, can be seen as a leaner version of the central search rectangle. By being accessible at all times, they represent a more convenient alternative to the homepage rectangle, yet their requirement of a download for installation prevents wider adoption. Notice that, to compensate this overhead, many search engines negotiate costly OEM deals with PC distributors/manufacturers to preinstall their toolbars.

• The "ultimate" rectangle, introduced by Google's Chrome "omnibox", merges the functionality of the address bar with that of the search box. It becomes then

responsibility of the browser to decide whether the text inputted by the user aims at navigating to a given site or at conducting a search. Prior to the introduction of the Omnibox, Firefox already provided functionality to recognize that certain words cannot be part of a URL and thus, should be treated as part of a query. In these cases, it would trigger Google's "I feel lucky" function to return the top search result. Interestingly enough, this Firefox feature is customizable, allowing users to trigger search engines other than Google or to obtain a full page of results.

The Search Engine Result Page

Result Presentation -The Basic Layout

The classic presentation style of the Search Engine Result Page, often referred to as SERP, consists of a list of "organic" or "algorithmic" results, that appear on the left hand side of the results page, as well as paid/sponsored results (ads) that appear on the right hand side. Additionally, the most relevant paid results might appear on top of the organic results in the North area, as illustrated in Figure 11.12. By default, most search engines show ten results in the first page, even though some engines, such as Bing, allow users to customize the number of results to show on a page. Figure 11.12 illustrates the layout generally adopted by most engines, with common but not uniformly adopted locations being indicated by a dotted line framed box.

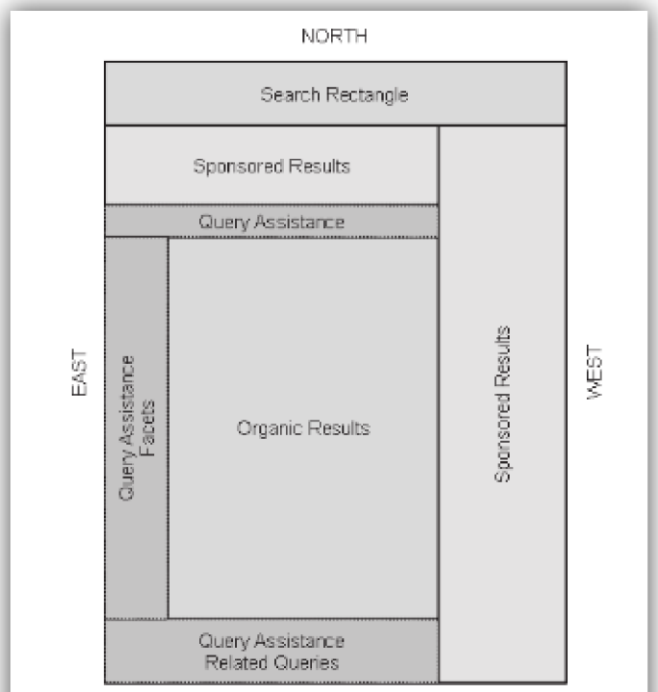


Figure 11.12: A typical SERP layout.

These engines m ght differ on small details like the "query assistance" features, which might appear in the North, South or West region of the page, the position of the navigational tools, which might or might not be displayed on the West region, or the position of spelling correction recommendations, which might appear before of after the sponsored results in the North region. Search engines constantly experiment with small variations of layout, and it might be the case that drastically different layouts be adopted in the future, as this is a space that calls for innovative features. To illustrate, Cuil introduced a radically different layout that departs from the one dimensional ranking, but this is more the exception than the rule. In contrast, search properties other than the main engines, commonly adopt distinct, such as the image search in both Google and Yahoo! as an example, or Google Ads search results, all of which display results across several columns. In this section, we focus exclusively on the organic part of search results. We will refer to them from now as "search results", note the distinction with paid/sponsored search results.

Major search engines use a very similar format to display individual results composed basically of

(a) a title shown in blue and underlined,

(b) a short snippet consisting of two or three sentences extracted from the result page, and

(c) a URL, that points to the page that contains the full text. In most cases, titles can be extracted directly from the page.

When a page does not have a title, anchor texts pointing to it can be used to generate a title.

More Structured Results

In addition to results fed by the main Web corpus, search engines include additional types of results, as follows.

Oneboxes" results. These are very specific results, produced in response to very precise queries, that are susceptible of having one unique answer. They are displayed above regular Web results, due to their high relevance, and in a distinct format. Oneboxes are triggered by specific terms in the user's query that indicate a clear intent. They aim at either exposing the answer directly or exposing a direct link to the answer, which provides the ultimate search experience but can be achieved only in very specific cases, i.e., when relevance is guaranteed and the answer is short and unambiguous.

As an example, both Google and Yahoo! Search support a weather onebox, which is triggered by entering "weather <a location>" (see example in Figure 11.13).

Figure 11.13: Example of weather onebox from Google, where is no need to click to get the full answer.

Universal search results: Most Web search engines offer, in addition to core Web search, other properties, such as Images, Videos, Products, Maps, which come with their own vertical search. While users can go directly to these properties to conduct corpus-specific searches, the "universal" vision states that users should not have to specify the target corpus. The engine should guess their intent and automatically return results from the most relevant sources when appropriate. The key technical challenge here is to select these sources and to decide how many results from each sources to display.

BROWSING

In this section, we cover browsing as an additional discovery paradigm, with special attention to Web directories. Browsing is mostly useful when users have no idea of how to specify a query (which becomes rarer and rarer in the context of the global Web), or when they want to explore a specific collection and are not sure of its scope. Nowadays, browsing is no longer the discovery paradigm of choice on the Web. Despite that, it can still be useful in specific contexts such as that of an Intranet or in vertical domains, as we now discuss. In the case of browsing, users are willing to invest some time exploring the document space, looking for interesting or even unexpected references. Both with browsing and searching, the user is pursuing discovery goals. However, in search, the user's goal

is somewhat crisper. In contrast, with browsing, the user's needs are usually broader. While this distinction is not valid in all cases, we will adopt it here for the sake of simplicity. We first describe the three types of browsing namely, flat, structure driven (with special attention to Web directories), and hypertext driven. Following, we discuss attempts at combining searching and browsing in a hybrid manner.

Flat Browsing

In flat browsing, the user explores a document space that follows a flat organization. For instance, the documents might be represented as dots in a two-dimensional plane or as elements in a single dimension list, which might be ranked by alphabetical or by any other order. The user then glances here and there looking for information within the visited documents. Note that exploring search results is a form of flat browsing. Each single document can also be explored in a flat manner via the browser, using navigation arrows and the scroll bar.

One disadvantage is that in a given page or screen there may not be any clear indication of the context the user is in. For example, while browsing large documents, users might lose track of which part of the document they are looking at. Flat browsing is obviously not available in the global Web due to its scale and distribution, but is still the mechanism of choice when exploring smaller sets. Furthermore, it can be used in combination with search for exploring search results or attributes. In fact, flat browsing conducted after an initial search allows identifying new keywords of interest. Such keywords can then be added to the original query in an attempt to provide better contextualization.

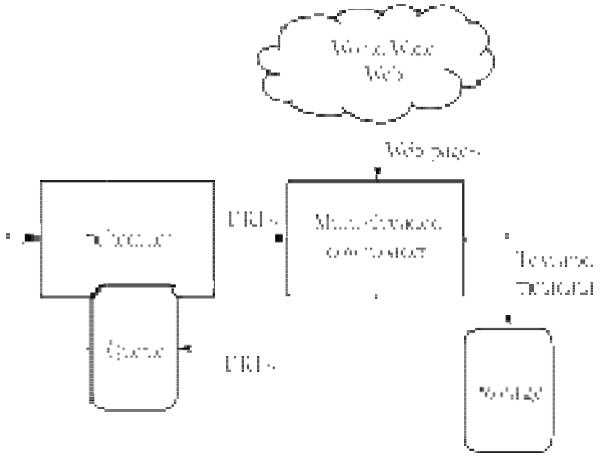
WEB CRAWLING

A **Web Crawler** is a software for downloading pages from the Web. Also known as Web Spider, Web Robot, or simply Bot.

A **Web crawler**, sometimes called a **spider** or **spiderwort** and often shortened to **crawler**, is an Internet bot that systematically browses the World

Wide Web, typically operated by search engines for the purpose of Web indexing (web spidering).

Web search engines and some other websites use Web crawling or spidering software to update their web content or indices of other sites' web content. Web crawlers copy pages for processing by a search engine, which indexes the downloaded pages so that users can search more efficiently. Crawlers consume resources on visited systems and often visit sites without approval. Issues of schedule, load, and "politeness" come into play when large collections of pages are accessed. Mechanisms exist for public sites not wishing to be crawled to make this known to the crawling agent. For example, including a robots.txt file can request bots to index only parts of a website, or nothing at all.



The number of Internet pages is extremely large; even the largest crawlers fall short of making a complete index. For this reason, search engines struggled to give relevant search results in the early years of the World Wide Web, before 2000. Today, relevant results are given almost instantly. Crawlers can validate hyperlinks and HTML code. They can also be used for web scraping and data-driven programming.

APPLICATIONS OF A WEB CRAWLER

A Web Crawler can be used to

- create an index covering broad topics (**general Web search**)
- create an index covering specific topics (**vertical Web search**)
- archive content (**Web archival**)
- analyze Web sites for extracting aggregate statistics (**Web characterization**)
- keep copies or replicate Web sites (**Web mirroring**)
- Web site analysis

Types of Web search

- **General Web search:** done by large search engines
- **Vertical Web search:** the set of target pages is delimited by a topic, a country or a language

Crawler for general Web search must balance coverage and quality.

- **Coverage**: It must scan pages that can be used to answer many different queries
- **Quality**: The pages should have high quality

Vertical Crawler: focus on a particular subset of the Web

✓ This subset may be defined geographically, linguistically, topically, etc.

Examples of vertical crawlers

1. **Shopbot**: designed to download information from on-line shopping catalogs and provide an interface for comparing prices in a centralized way
2. **News crawler**: gathers news items from a set of pre-defined sources
3. **Spambot**: crawler aimed at harvesting e-mail addresses inserted on Web pages

Vertical search also includes segmentation by a data format. In this case, the crawler is tuned to collect only objects of a specific type, as image, audio, or video objects. Example

Feed crawler: checks for updates in RSS/RDF files in Web sites.

Focused crawlers: focus on a particular topic

Provides a more efficient strategy to avoid collecting more pages than necessary

A focused crawler receives as input the description of a topic, usually described by

- a driving query
- a set of example documents

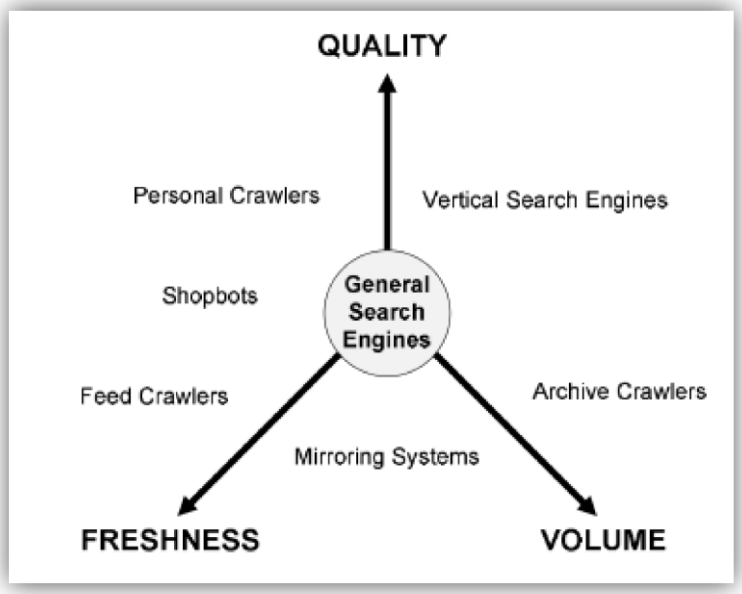
The crawler can operate in

batch mode, collecting pages about the topic periodically

on-demand, collecting pages driven by a user query

TAXONOMY

The crawlers assign different importance to issues such as **freshness**, **quality**, and **volume** The crawlers can be classified according to these three axes



A crawler would like to use all the available resources as much as possible crawling servers, Internet bandwidth). However, crawlers should also fulfill **politeness,** That is, a crawler cannot overload a Web site with HTTP requests. That implies that a crawler should wait a small delay between two requests to the same Web site Later we will detail other aspects of politeness

ARCHIT CTURE AND IMPLEMENTATION

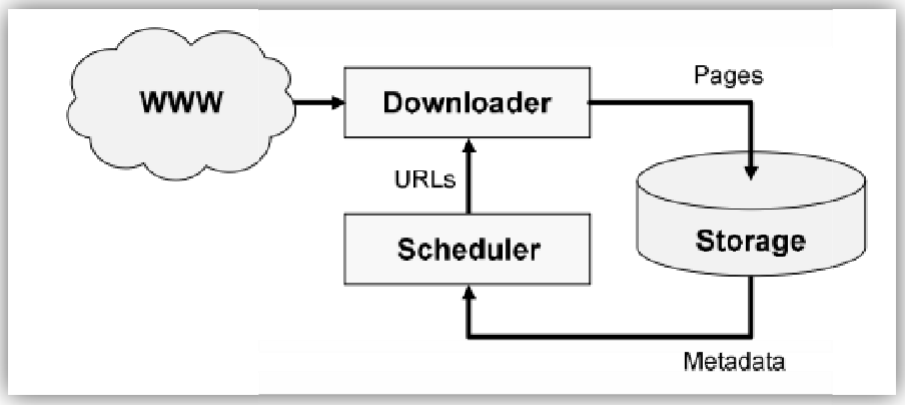
The crawler is composed of three main modules:

- downloader,
- storage, and
- scheduler

Scheduler: maintains a queue of URLs to visit

Downloader: downloads the pages

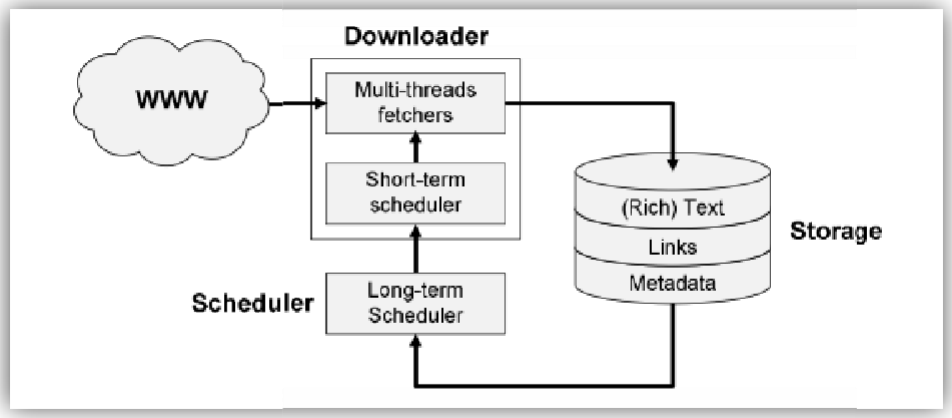
Storage: makes the indexing of the pages, and provides the scheduler with metadata on the pages retrieved.



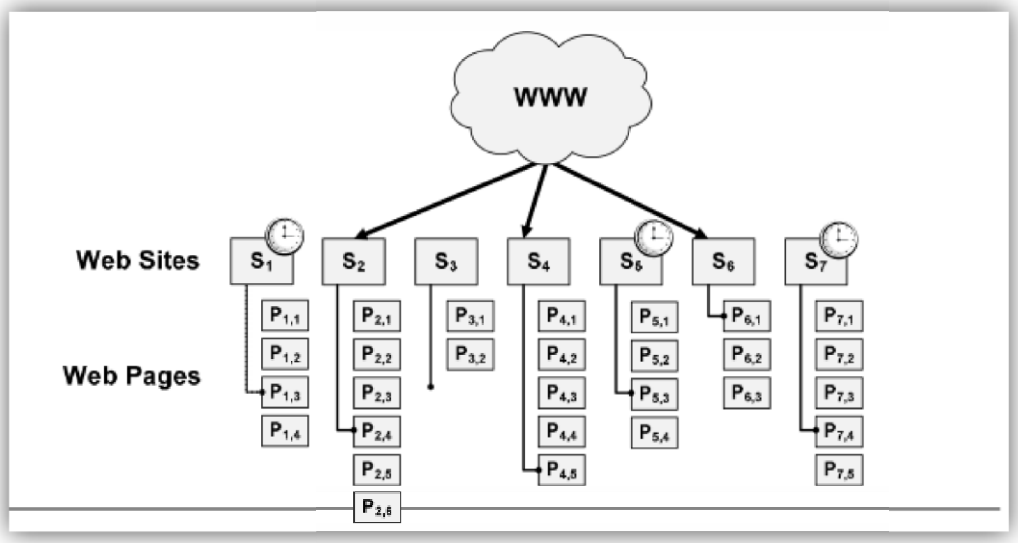
The scheduling can be further divided into two parts:

- **long-term scheduling**: decide which pages to visit next
- **short-term scheduling**: re-arrange pages to fulfill politeness

The storage can also be further subdivided into three parts: (rich) text, metadata, and links.



In the **short-term scheduler**, enforcement of the politeness policy requires maintaining several queues, one for each site, and a list of pages to download in each queue.



The implementation of a crawler involves many **practical issues.**

Most of them is due to the need to interact with many different systems

Example: How to download maintaining the traffic produced as uniform as possible?

The pages are from multiple sources DNS and Web server response times are highly variable Web server up-time cannot be taken for granted.

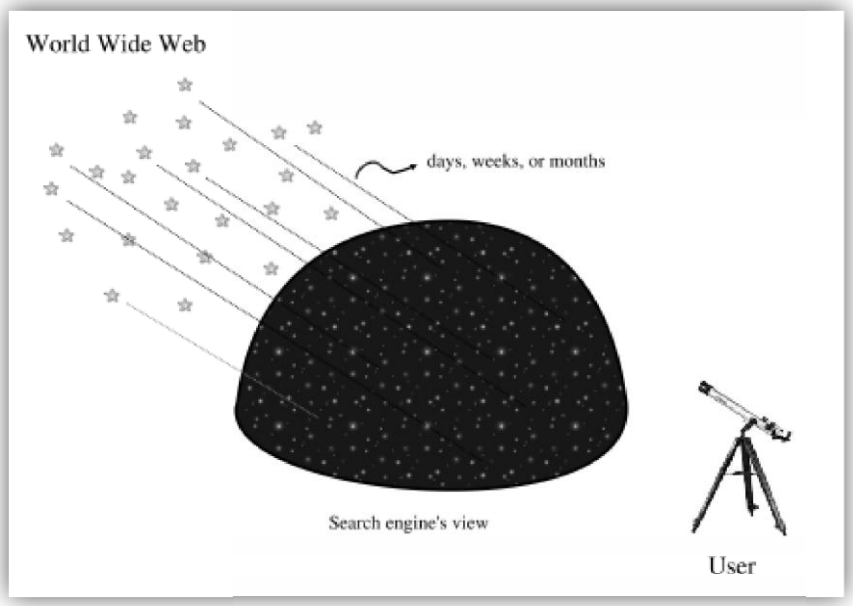
Other practical issues are related with:

types of Web pages, URL canonization, parsing, wrong implementation of HTML standards, and duplicates.

SCHEDULING ALGORITHMS

A Web crawler needs to balance various objectives that contradict each other It must download new pages and seek fresh copies of downloaded pages It must use network bandwidth efficiently avoiding to download bad pages However, the crawler cannot know which pages are good, without first downloading them To further complicate matters there is a huge amount of pages being added, changed and removed every day on the Web

Crawling the Web, in a certain way, resembles watching the sky in a clear night: the star positions that we see reflects the state of the stars at different times.



The simplest crawling scheduling is traversing Web sites in a breadth-first fashion. This algorithm increases the Web site coverage and is good to the politeness policy by not requesting many pages from a site in a row. However, can be useful to consider the crawler's behavior as a combination of a series of policies To illustrate, a crawling algorithm can be viewed as composed of three distinct policies

- **selection policy**: to visit the best quality pages, first
- **re-visit policy**: to update the index when pages change
- **politeness policy**: to avoid overloading Web sites

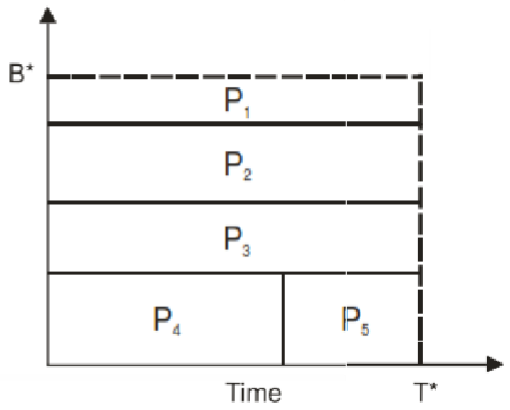
CRAWLING EVALUATION

The diagram below depicts an optimal Web crawling scenario for an hypothetical batch of five pages. The x-axis is time and the y-axis is speed, so the area of each page is its size (in bytes).

The downloader has maximum bandwidth B^* so the crawl can be completed in time:

$$T^* = \frac{\sum_i \text{size}(P_i)}{B^*}$$

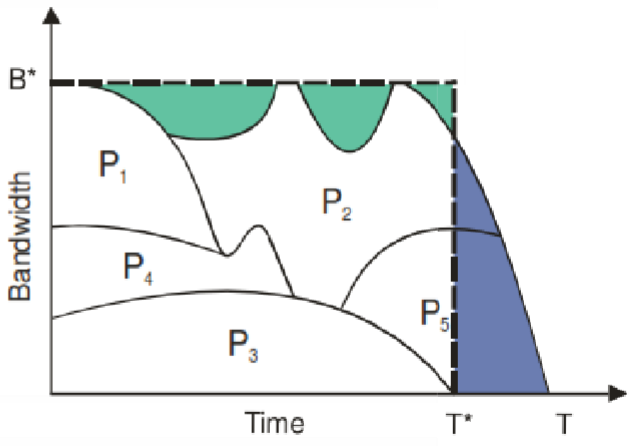
where $\text{size}(P_i)$ is the size of page P_i



Let us consider now a more realistic setting in which:

The speed of download of every page is variable and bounded by the effective bandwidth to a Web site (a fraction of the bandwidth that the crawler would like to use can be lost) Pages from the same site can not be downloaded right away one after the other (politeness policy).

- Under these assumptions, a different crawling timeline may occur, such as the one depicted on the figure below
- In the realistic case, the total time T is larger than the optimal case
- The bandwidth lost can be measured and is given by $B^* \times (T - T^*)$
- In the figure, green and blue areas are equal



By the end of the batch of pages, it is very likely that only a few hosts are active

Once a large fraction of the pages have been downloaded it is reasonable to stop the crawl, if only a few hosts remain at the end. Particularly, if the number of hosts remaining is very small then the bandwidth cannot be used completely.

A short-term scheduler may improve its finishing time by saturating the bandwidth usage, through the use of more threads However, if too many threads are used, the cost of switching control among threads becomes prohibitive The ordering of the pages can also be optimized by avoiding having a few sites to choose from at any point of the batch.

PART-A

1 Express the basics of web search with a neat diagram.
2 Explain Pay for Placement.
3 What is meant by Search Engine Optimization?
4 List the need of web search engine.
5 Demonstrate the architecture of search engine.
6 Compare parallel crawler and meta crawler.
7 List the SPAM Techniques.
8 Evaluate use of inversion in indexing process.
9 State the issues in search engines.
10 Design the Politeness policies used in web crawler.
11 Classify the ways to identify duplication.
12 How to Apply duplicate Deduction to web pages?
13 Assess the need for keyword stuffing.
14 What are the challenges in data traversing by search engines?
15 Identify the applications of web crawlers.
16 Classify the use of Web indexing and inversion of indexing process.
17 What is focused crawler?
18 Illustrate the hashing technique with example.
19 Classify the types of search engines.
20 Generalize on XML Retrieval.

PART-B

1 Demonstrate the Search Engine Optimization/SPAM in detail.(13)
2 i) Describe in detail about Vector space model for XML Retrieval. (9)
ii)What is Structured and Unstructured Retrieval.(4)
3 i) List the types of Search Engine and explain them. (7)
ii) Distinguish visual vs programmatic crawler. (6)
4 Design and develop a Web search Architecture and the components of search engine and its issues.(13)
5 i) What is P4P? Elaborate on Paid Placement. (7)
ii) Describe the structure of WEB and its characteristics (6)
6 i) Summarize on the working of WEB CRAWLER with its diagram. (8)
ii) Explain the working of Search Engine. (5)
7 i) Differentiate meta crawler and focused crawler. (8)
ii) Analyze on URL normalization.(5)
8 Recommend the need for Near-Duplication Detection by the way of finger print algorithm. (13)
9 i) Examine the behavior of web crawler and the outcome of crawling policies. (5)
ii) Illustrate the following (8)
a) Focused Crawling
b) Deep web
c) Distributed crawling
d) Site map
10 i) Explain the overview of Web search. (8)
ii) What is the purpose of Web indexing? (5)

11 Summarize the process of index compression in detail.(13)
12 (i) Examine the need for Web Search Engine. (6)
(ii) List the challenges in data traversing by search engine and how will you overcome it.(7)
13 Describe the following with example.
i) Bag of Words and Shingling (7)
ii) Hashing, Min Hash and Sim Hash (6)
14 (i) Based on the Application of Search Engines, How will you categorize them and what are the issues faced by them? (9)
(ii) Demonstrate about Search Engine Optimization. (4)

PART-C

1 Develop a web search structure for searching a newly hosted web domain by the naïve user with step by step procedure. (15)
2 i) Grade the optimization techniques available for search engine and rank them by your justification. (9)
ii) Explain Web Crawler Taxonomy in detail (6)
3 Estimate the web crawling methods and illustrate how do the various nodes of a distributed crawler communicate and share URLs? (15)
4 Formulate the application of Near Duplicate Document Detection techniques and also Generalize the advantages in Plagiarism checking. (15)