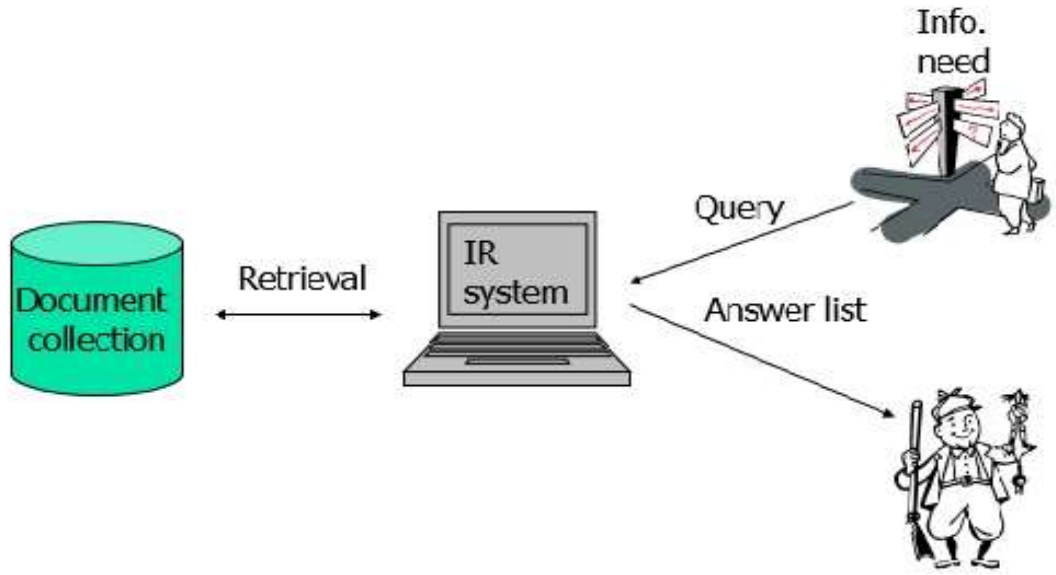
1.1. Introduction:

- Text mining refers to data mining using text documents as data.
- Most text mining tasks use Information Retrieval (IR) methods to pre-process text documents.
- These methods are quite different from traditional data pre-processing methods used for relational tables.
- Web search also has its root in IR.
- **Web**: A huge, widely-distributed, highly heterogeneous, semi structured, interconnected, evolving, hypertext/hypermedia information repository.

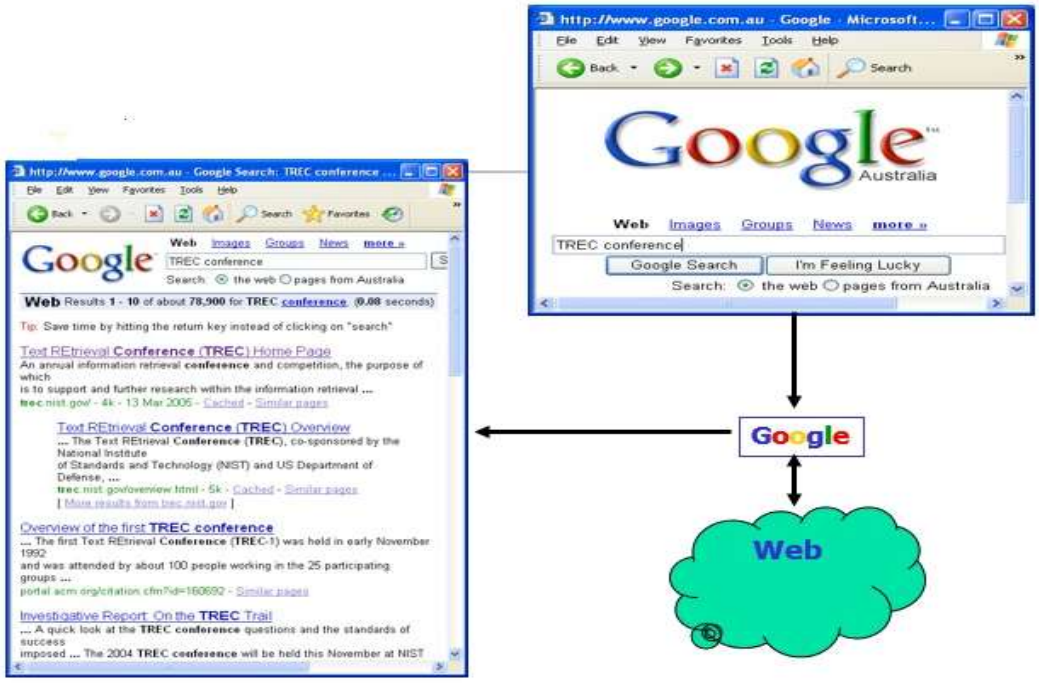
The Goal of IR:

Goal to find documents **relevant** to an information need from a large documents set.



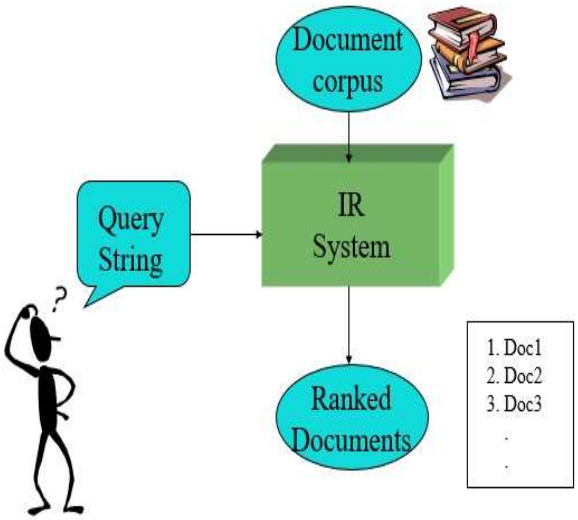
Information retrieval and web search architecture

IR & WS Example:



1.2. Definitions:

- **Information retrieval (IR):** subfield of computer science that deals with automated retrieval of documents (especially text) based on their content and context.



- **Searching:** seeking for specific information within a body of information. The result of a search is a set of **hits** (looking for very specific term)
- **Browsing:** unstructured exploration of a body of information. (nothing is special in your find)
- **Linking:** moving from one item to another following links, such as citations, references, etc.

1.3. The basics of Information Retrieval:

- **Query:** A string of text, describing the information that the user is seeking. each word of the query is called **a search term**.
- A query can be a single search term, a string of terms, a phrase in natural language, or a stylized expression using special symbols.
- **Full text searching**: methods that compare the query with every word in the text, without distinguishing the function of the various words.
- **Fielded searching**: methods that search on specific bibliographic or structural fields such as author or heading.

1.4. Sorting and Ranking Hits:

- When a **user** submits a **query** to a **search system**, the system returns a set of **hits**. With a large collection of documents, the set of hits maybe very large.
- The value of the use depends on the order in which the hits are presented.
- Three main methods:
  - **Sorting** the hits, e.g., by date.
  - **Ranking** the hits by **similarity** between query and documents
  - **Ranking** the hits by the **importance** of the documents

Possible approaches:

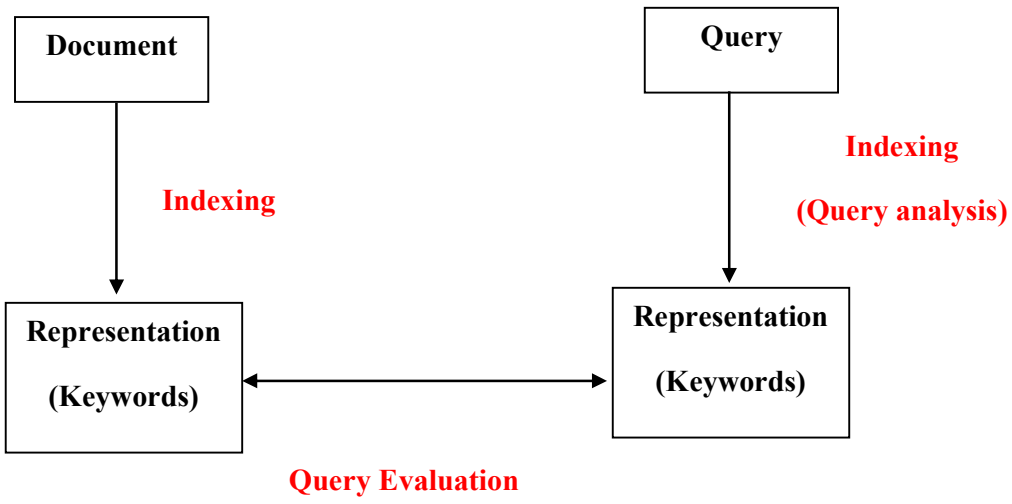
1- String matching (linear search in documents)
- Slow
- Difficult to improve

2- Indexing
- Fast
- Flexible to further improvement.

Examples of Search Systems:

- **Find file** on a computer system (Spotlight for Macintosh).
- **Library catalog** for searching bibliographic records about books and other objects (library of congress catalog).
- **Abstracting and indexing system** for finding research information about specific topics (Medline for medical information).
- **Web search service** for finding web pages (google).

1.5. Indexing-based IR:



1.6. Main problems in IR:

- Document and query indexing.
  - How to best represent their contents?
- Query evaluation (or retrieval process).
  - To what extent does a document correspond to a query?
- System evaluation
  - How good is a system?
  - Are the retrieved documents relevant? (precision).
  - Are all the relevant documents retrieved? (recall).

1.7. Document indexing:

- **Goal** to Find the important meanings and create an internal representation
- Factors to consider:
  - Accuracy to represent meanings (semantics).
  - Exhaustiveness (cover all the contents).
  - Facility for computer to manipulate.
- What is the best representation of contents?
  - Char. string (char trigrams): not precise enough.
  - Word: good coverage, not precise.
  - Phrase: poor coverage, more precise.

Information Retrieval (IR):

Is finding material (usually documents) of *an unstructured nature* (usually text) that satisfies an information need from within *large collections* (usually stored on computers).

unstructured data refers to data which does not have clear, semantically overt, easy-for-a-computer structure. It is the opposite of structured data, the canonical example of which is a relational database.

Relevance: the relevance of results is assessed relative to the information need,

not the query.

Relevance is a concept with interesting properties. First, it is **subjective**: two users may have the same information need and give different judgments about the same retrieved document. Another aspect is its **dynamic nature**, both in space and in time: documents retrieved and displayed to the user at a given time may influence relevance judgments on the documents that will be displayed later. Another aspect is its **multifaceted**, as it is determined not just by the content of a retrieved result but also by aspects such as the authoritativeness, credibility, specificity, exhaustiveness, recency, and clarity of its source.

Formal Characterization of IR Model:

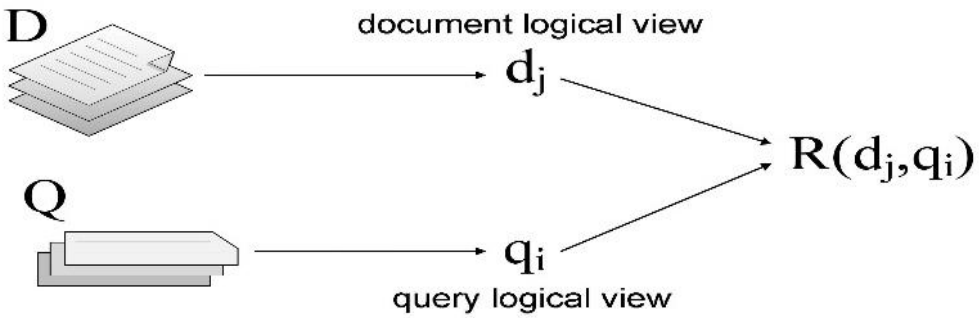
An information retrieval model (IRM) can be defined as a quadruple:

$$IRM = \{D, Q, F, R(qi, dj\}$$

Where:

- D is a set of logical views (or representations) of the documents in the collection (referred to as dj).
- Q is a set of logical views (or representations) of the user's information needs, called queries (referred to as qi).
- F is a framework (or strategy) for modeling the representation of documents, queries, and their relationships.
- R (qi, dj) is a ranking function that associates a real number to a document representation.

  dj, denoting its relevance to a query qi.



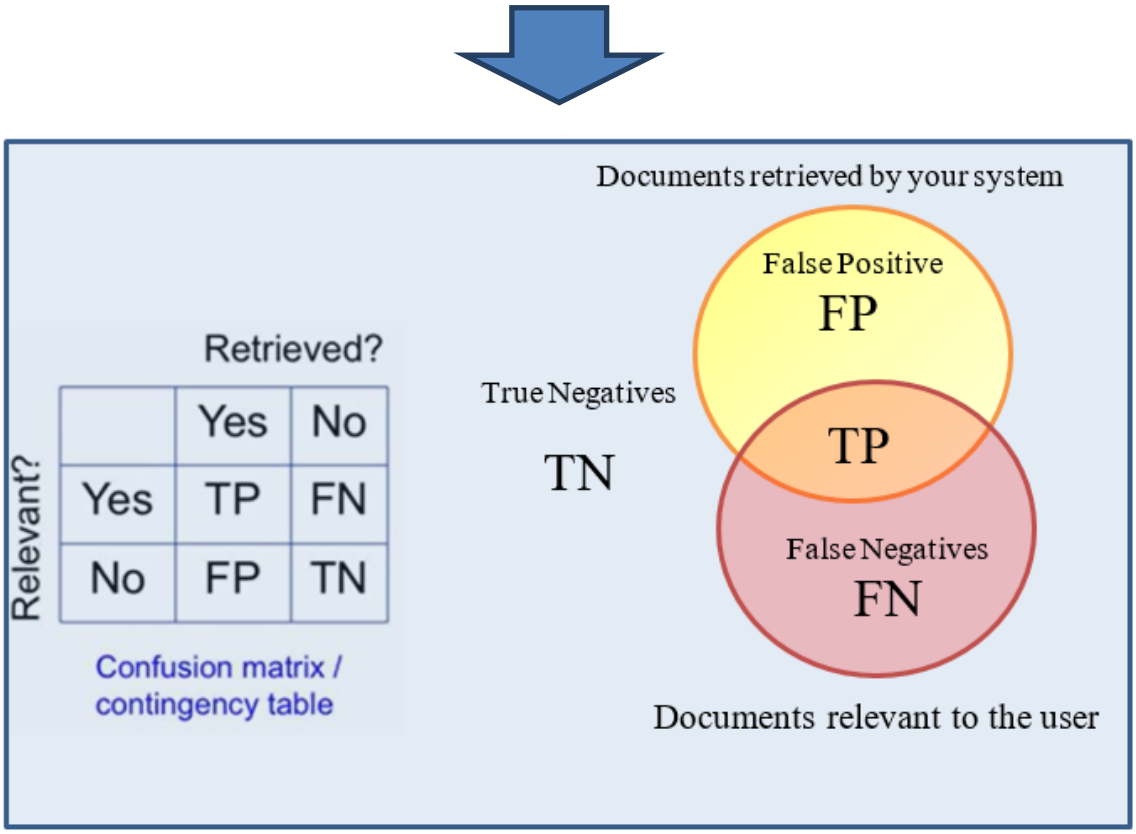
Formal Characterization of IR Model

Evaluating an Information Retrieval:

Precision (P): is the fraction of retrieved documents that are relevant to a query and provides a measure of the "soundness" of the system.

Recall (R): is the fraction of "truly" relevant documents that are effectively retrieved and thus provides a measure of the "completeness" of the system.

All Documents in the Test Corps



Confusion Matrix / Contingency Table:

Relevant: Yes, Retrieved: Yes =(TP) True Positive. معلومات ذات صلة وتم إسترجاعها

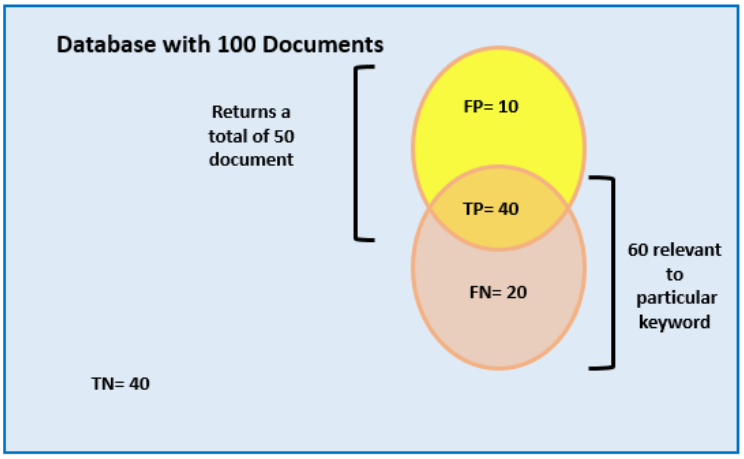
Relevant: Yes, Retrieved: No = (FN) True Negative.معلومات ذات صلة ولكن لم يتم إسترجاعها

Relevant: No, Retrieved: Yes = (FP) False Positive. معلومات ليست ذات صلة وتم إسترجاعها

Relevant: No, Retrieved: No = (TN) True Negative. معلومات ليست ذات صلة ولم يتم إسترجاعها

$$Precision = \frac{TP}{TP+FP} \quad , \quad Recall = \frac{TP}{TP+FN}$$

Example1: If I have a database with 100 documents, out of which 60 are relevant to a particular keyword. If my IR system returns a total of 50 documents, out of which 40 are relevant. *find the precision and the recall* to the system. P= 0.8, R= 0.66



Solution:

$$Precision = \frac{40}{40+10} = 0.8$$

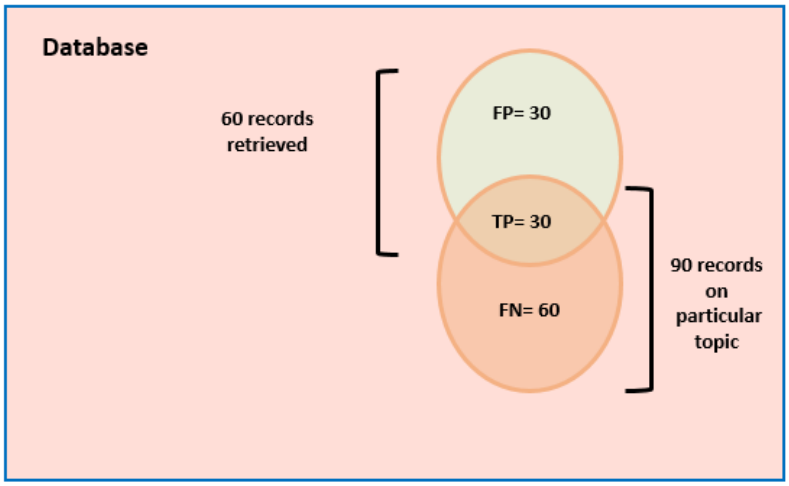
$$Recall = \frac{40}{40+20} = 0.66$$

Example2: Assume the following:

A database contains 90 records on a particular topic. A search was conducted on that topic and 60 records were retrieved, 30 were relevant.

1- Calculate the **precision** and **recall** for the search.

2- State (illustrate) the value of (TP, FP, FN) in a figure.

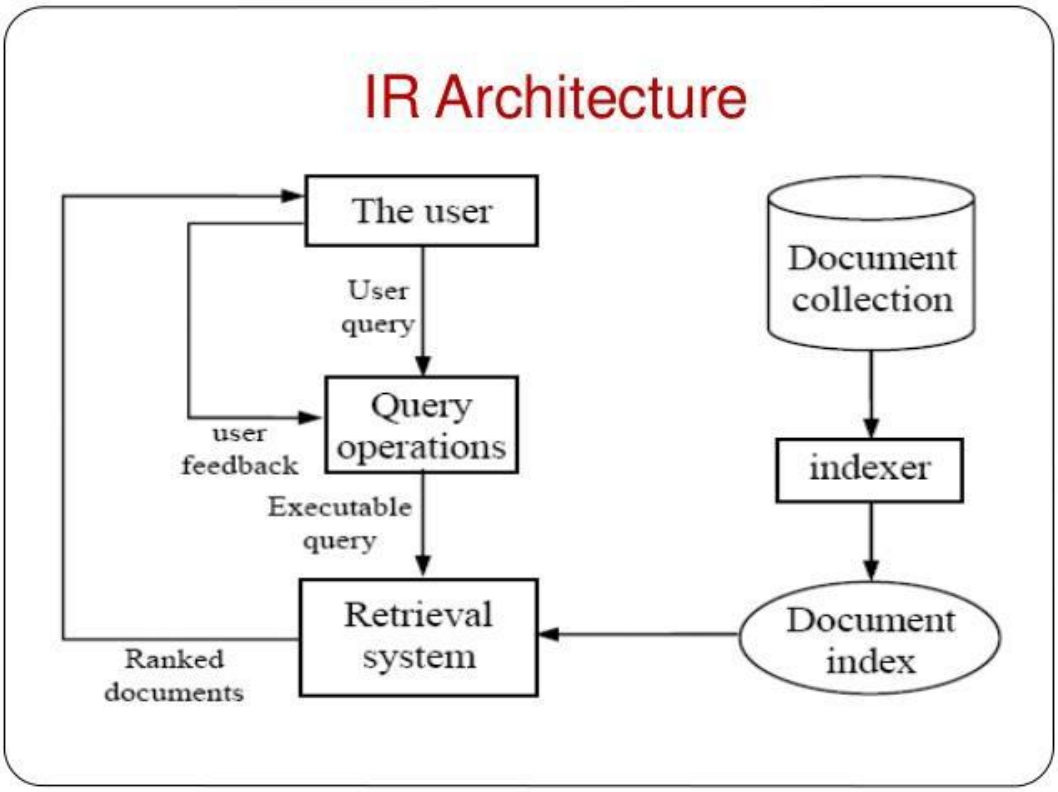


Solution:

$$Precision = \frac{30}{30+30} = 0.5$$

$$Recall = \frac{30}{30+60} = 0.33$$

Information Retrieval Architecture:



Indexing Process:

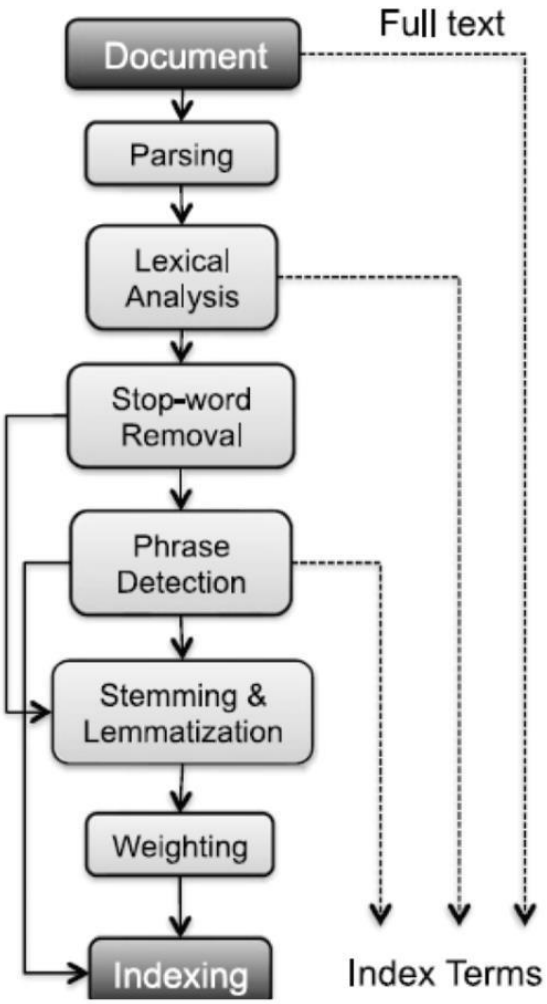
The index is a logical view where documents in a collection are represented through a set of index terms or keywords.

The indexing process consists of three basic steps:

1. **defining the data source database manager module,** which specifies the documents, the operations to be performed on them, the content structure, and what elements of a document can be retrieved (e.g. the full text, the title, the authors).
2. Transforming document content to generate a logical view.
3. Building an index of the text on the logical view.

Textual Operations:

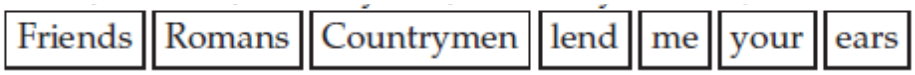
Figure below sketches the textual preprocessing phase typically performed by an IR engine, taking as input a document and yielding its index terms.



1- **Document Parsing**: Documents come in all sorts of languages, character sets, and formats often, the same document may contain multiple languages or formats. Document parsing deals with the recognition and "breaking down" of the document structure into individual components. In this preprocessing phase, unit documents are created. example emails with attachments are split into one document representing the email and as many documents as there are attachments.

2- **Lexical Analysis**: After parsing, lexical analysis tokenizes a document, seen as an input stream, into words.

Input: Friends, Romans, Countrymen, lend me your ears.

Output: 

Issues related to lexical analysis include the correct identification of accents, abbreviations, dates.

3- **Stop-Word Removal**: A subsequent step optionally applied to the results of lexical analysis is stop-word removal, i.e., ***the removal of high-frequency words***. For example, "*search engines are the most visible information retrieval applications*" the effect of stop-word removal would be: "*search engine most visible information retrieval applications*".

However, as this process may decrease recall (prepositions are important to disambiguate queries), most search engines do not remove them. The subsequent phases take the full-text structure derived from the initial phases of parsing and lexical analysis and process it in order to identify relevant keywords to serve as index terms.

4- **Phrase Detection:** This step captures text meaning beyond what is possible with pure bag-of-word approaches. Phrase detection may be approached in several ways, including rules (e.g., retaining terms that are not separated by punctuation marks). For example, scanning our example sentence "*search engines are the most visible information retrieval applications"* for noun phrases would probably result in identifying "*search engines*" and "*information retrieval*".

5- **Stemming and Lemmatization**: Following phrase extraction, stemming and lemmatization down aim at stripping to word suffixes in order normalize the word. In particular, stemming is a heuristic process that "chops off" the ends of words in the hope of achieving the goal correctly. According to the Porter stemmer, our example sentence "*Search engines are the most visible information retrieval applications*" would result in: "*Search engine are the most visibl inform retriev applic".*

Lemmatization: is a process that typically uses dictionaries and morphological analysis of words in order to return the base or dictionary form of a word, thereby collapsing its inflectional forms for example our sentence would result in *"Search engine are the most visible information retrieval application".*

6- **Weighting**: The final phase of text pre-processing deals with term weighting. As previously mentioned, words in a text have different descriptive power; hence, index terms can be weighted differently to account for their significance within a document and /or a document collection. Such a weighting can be binary, example assigning 0 for term absence and 1 for presence

Index Compression:

1. Use less disk space.
2. Saves a little money.
3. Keep more stuff in memory.
4. Increases speed.
5. Increase speed of data transfer from disk to memory.
6. [read compressed data | decompress] is faster than [read uncompressed data].

Basic concepts:

- Each documents are represented by a set of representative **keywords** or **index terms**.
- An index term is a word or group of consecutive words in a document.
- **A pre-selected set** of index terms can be used to summarize the document contents.
- We usually assume that all words are index terms **(full text representation)**.
- **Vocabulary** V={k_i,…,k_t}.
- The set of all distinct index terms in the collection**.**
- The occurrence of a term k_i in a document dj establishes a relation between k_i and d_j.
- Term-document matrix: term-document relation in matrix form

$$\begin{array}{c} \\ k_1 \\ k_2 \\ k_3 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} f_{1,1} & f_{1,2} \\ f_{2,1} & f_{2,2} \\ f_{3,1} & f_{3,2} \end{bmatrix} \end{array}$$

Where each $f_{i,j}$ represents the frequency of term k_i in a document d_j.

From full text to a set of index terms:

1- Structure recognition (ex: remove html formatting).
2- Correct for accents, spacing, etc.
3- Stop words.
4- Detect noun groups.
5- Stemming.
6- Filter for known vocabulary (if needed).

Boolean Model:

- Simple model based on **set theory** and **boolean algebra**.
- Queries specified as **boolean expressions**.
  - Quite intuitive and precise semantics.
  - Neat formalism.
- **Term-document frequencies** in the **term-document matrix** are all **binary**.

Boolean Retrieval:

Advantages:

- Results are predictable, relatively easy to explain.
- Many different features can be incorporated.
- Efficient processing since many documents can be eliminated from search.

Disadvantages:

- Effectiveness depends entirely on user.
- Simple queries usually don't work well.
- Complex queries are difficult.

Vector Space Model:

- Documents and query represented by a vector of term weights.
- Collection represented by a matrix of term weights.
- Vector space = all the keywords encountered

  $<t_1,t_2,t_3,\ldots..,t_n>$

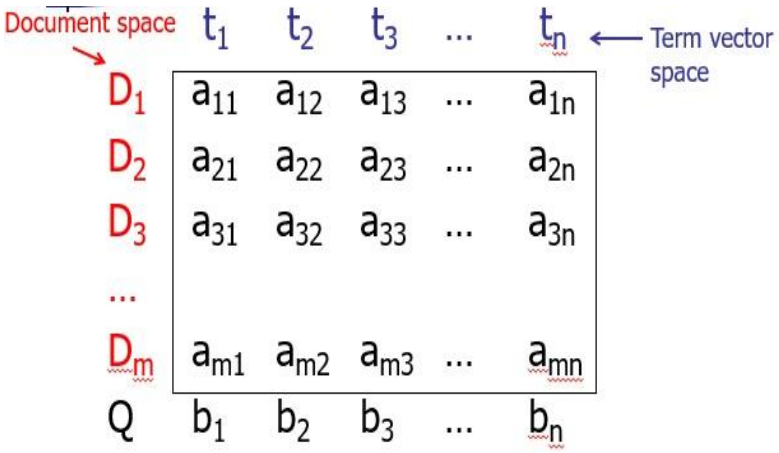
- Document $D = <a_1,a_2,a_3,\ldots.,a_n>$

  a_i= weight of t_i in D

- Query Q = $<b_1,b_2,b_3,\ldots b_n>$

  b_i =weight of t_i in Q

Matrix representation:



Advantages:

• Term-weighting improves quality of the answer set.

• Partial matching allows retrieval of docs that approximate the query conditions.

Disadvantages:

• It assumes independence of index terms.

Boolean Information Retrieval Models:

Matching process: is the method for determining the degree of relevance of the user's query with respect to the document representation, this process is expected to produce a ranked list of documents, where relevant documents should appear towards the top of the ranked list, in order to minimize the time spent by users in identifying relevant information.

Term-document incidence matrix: is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.

Example:

- DOC1 = "I like databases"
- DOC2 = "I hate databases"

then the document-term matrix would be:

	I	like	hate	databases
D1	1	1	0	1
D2	1	0	1	1

which shows which documents contain which terms and how many times they appear.

Boolean Model: is a simple retrieval model based on set theory and Boolean algebra, whereby queries are defined as Boolean expressions over index terms (using the Boolean operators AND, OR, and NOT), e.g., "photo AND mountain OR snow".

Find the answer by gripping (all rows with a property) all the works of Shakespeare Anthony and Cleopatra

	Julius Caesar	The Tempest (Sturm)	Hamlet	Othello	Macbeth	...
Anthony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser (Schlechtere)	1	0	1	1	1	0

To answer the query Brutus AND Caesar AND NOT Calpurnia, we take the vectors for Brutus, Caesar and Calpurnia, complement the last, and then do a bitwise AND:

110100 AND 110111 AND 101111 = 100100.

Note: we assume an average of 6 bytes per word including spaces and punctuation, then this is a document collection about 6 GB in size. Typically, there might be about M = 500,000 distinct terms in these documents.

general example:

- 1 million documents
- each document 1000 words (3 book pages)
- 6 bytes per word

document collection= 6 Gbyte

typically: 500 000 distinct terms (words)

500 000 * 1 million = 0.5 Terabits (60 Gbyte)

A system cannot handle this amount of data;

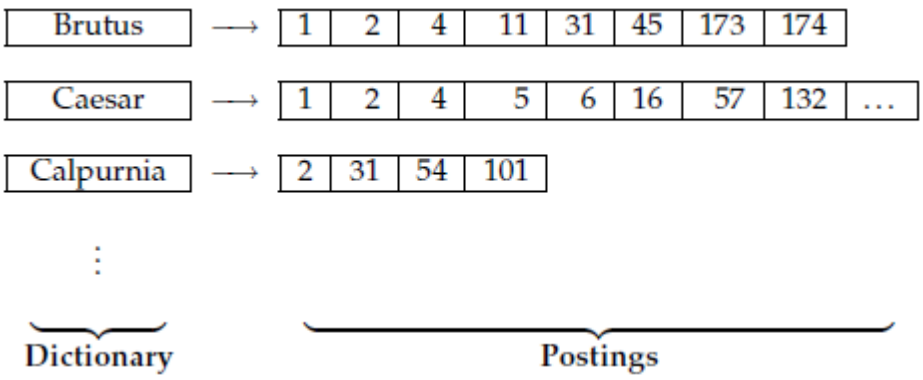
Critical observation: the matrix is extremely sparse: 99.8% of cells are 0

Inverted index: is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. The purpose of an inverted index is to allow fast full text searches.

To gain the speed benefits of indexing at retrieval time, we have to build the index in advance. The major steps in this are:

1. Collect the documents to be indexed:
   Friends, Romans, countrymen. | So let it be with Caesar | ...
2. Tokenize the text, turning each document into a list of tokens:
   Friends | Romans | countrymen | So | ...
3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms: friend | roman | countryman | so | ...
4. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

Example:



Indexing process:

1. Input: list of normalized tokens for each document.
2. Sort the terms alphabetically.
3. Merge multiple occurrences of the same term.
4. Record the frequency of occurrence of the term in the document.
5. Group instances of the same term and split dictionary and postings.

Example:

Doc 1
I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

Doc 2
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2

⟹

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2

⟹

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
I	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Exercises:

Draw the inverted index that would be built for the following document collection.

Doc 1: new home sales top forecasts

Doc 2: home sales rise in July

Doc 3: increase in home sales in July

Doc 4: July new home sales rise

Information Retrieval for the Web:

Information Seeking on the Web: There are three strategies for seeking information in the web:

1- Direct navigation: The simplest strategy is *direct navigation* when the web site address (or URL" Uniform Resource Locator ") is entered directly into the browser. This strategy is often successful for finding home pages of companies such as www.ibm.com, institutions such as www.mit.edu. Direct navigation is less successful for finding products such as Apple's iBook or HP's iPAQ, since these do not necessarily map directly to the required web addresses.

2- Navigation within a directory: A *web portal* is a web site that provides a gateway, or an entry point, to other resources on the web. Examples of portals are www.msn.com, and www.yahoo.com. Web portals provide a broad range of features, services and content. of special interest to the information seeker are portals that are organized in the form of a subject directory such as Yahoo, Web directories consist of a topic categorization, including amongst other categories: Arts, Business, Computers and Internet, Entertainment, Government, News, and Science.

3- Navigation using a search engine: For the search engine strategy a user seeking information on the web will normally iterate through the following steps:

Step 1: Query formulation − the user submits a query to a search engine specifying his or her goal; normally a query consists of one or more input keywords.

Step 2: Selection − the user selects one of the web pages from the ranked results list returned by the search engine, clicks on the link to that page, and browses the page once it is loaded into the browser.

Step 3: Navigation (or surfing) − the user initiates a navigation session, which is the process of clicking on links and browsing the pages displayed. The user surfing the

web by link following will use various cues and tools to augment their navigational activity.

Step 4: Query modification − a navigation session may be interrupted for the purpose of query modification, when the user decides to reformulate the original query and resubmit it to the search engine. In this case the user returns to step (1).

Problems with web information seeking:

- Results are not stable and that users may need to vary their strategy over time to satisfy similar needs.
- The quality of information on the web is extremely variable and the user has to make a judgment.
- Factual knowledge on the web is not objective, so if you want to find out who is the president of the United States you may get several answers. In this case you may trust the White House web site to give you a correct answer but other sites may not be so trustworthy.

Informational, Navigational and Transactional Queries:

Depending on the specification of the query and the quality of the search engine, the user issuing an informational or transactional query may satisfy his or her information need with minimal navigation and query modification. There are three categories of queries:

1- **Informational** − when the user's intent is to acquire some information about a topic presumed to be present in one or more web pages.

2- **Navigational**−when the user's intent is to find a particular site from which to start surfing.

3- **Transactional** − when the user's intent is to perform some activity which is mediated by a web site, e.g. online shopping, or another web service such as news delivery, online library browsing, or some other specialist service.

Search Engines:

Search engines are programs that search documents for specified keywords and returns a list of the documents where the keywords were found.

web search engine: is a software system that is designed to search for information on the World Wide Web. The search results are generally presented in a line of results often referred to as search engine results pages (SERPs).

How search engine works?

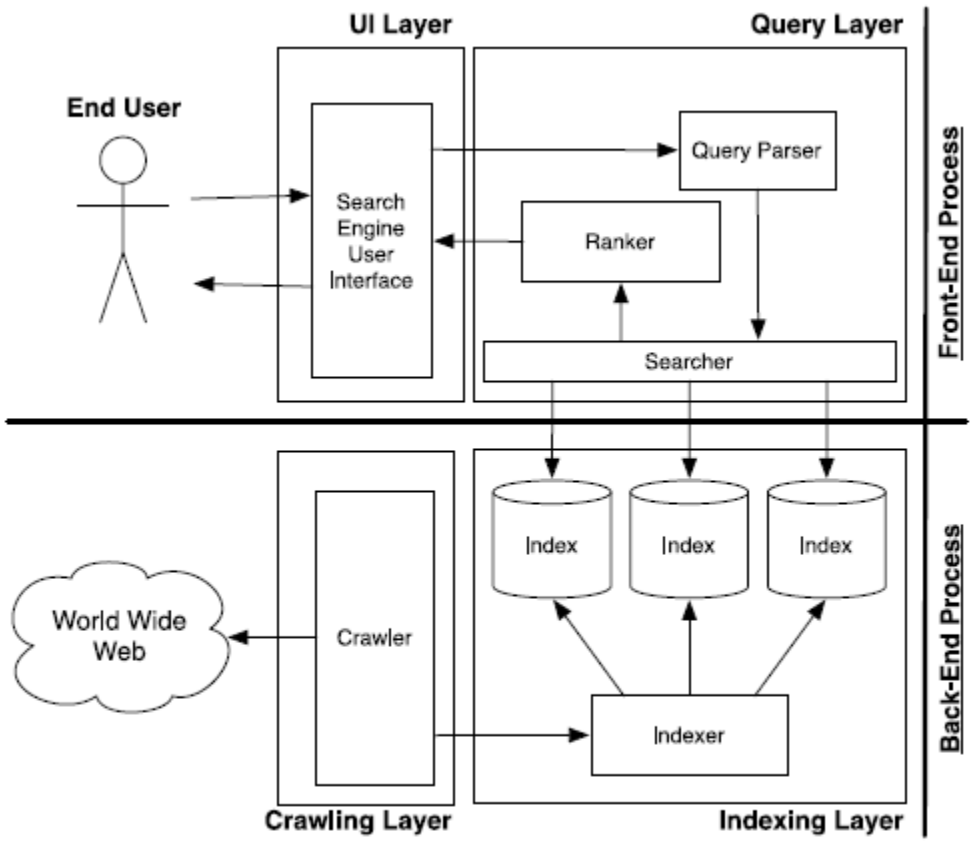
A search engine operates in the following order:

1. Web crawling.
2. Indexing.
3. Searching.

Web search engines work by storing information about many web pages. These pages are retrieved by a Web crawler (sometimes also known as a spider) — an automated Web crawler which follows every link on the site. The search engine then analyses the contents of each page to determine how it should be indexed (for example, words can be extracted from the titles, page content, headings, or special fields called meta tags). The engine examines its index and provides a listing of best- matching web pages according to its criteria, usually with a short summary containing the document's title and sometimes parts of the text. The index is built from the information stored with the data. Most search engines support the use of the Boolean operators AND, OR and NOT to further specify the search query.

The architecture of a web search engine:

Web search engines are roughly composed of a *front-end process* and a *back-end process*. As show in figure below:



The architecture of a Web search engine

The front-end process involves the ***user interface*** and ***searcher*** **layers**. Upon the submission of a user query, it is devoted to:

(i) parsing and understanding of the query.

(ii) retrieval of relevant results from the search engine indexes

(iii) ranking of the matching results.

(iv) interaction with the user.

The back-end process, involves the ***indexing*** and the ***crawling*** **layers**, *and it is in charge of guaranteeing that the Web search engines are able to handle hundreds of thousands of queries per second, while offering sub-second query processing time and an updated view on the current status of the available Web resources.*

Crawling:

In order for a search engine to provide accurate responses to users, Web resources must be identified, analyzed, and cataloged in a timely and exhaustive manner. This content exploration activity, called Web crawling**,** *is an automated process that analyzes resources in order to extract information and the link structure that interconnects them. The computer programs in charge of traversing the Web to discover and retrieve new or updated pages are typically referred to as crawlers*; they can also be called spiders, robots, worms, walkers, etc. Crawlers have many purposes (e.g., Web analysis, email address gathering, etc.), although their main one is related to the discovery and indexing of pages for Web search engines. The whole set of resources accessible from the Web can be roughly classified into three categories:

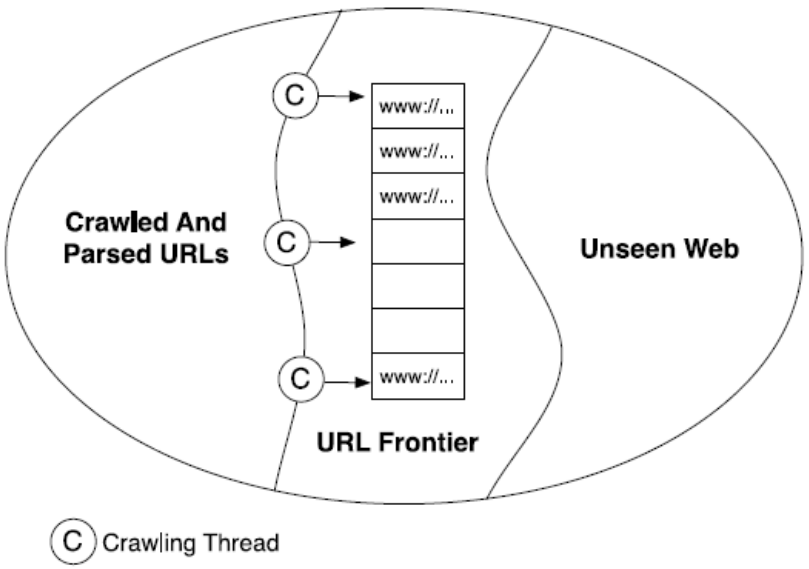
- ***Web pages***, textual documents written in the Hyper Text Mark-up Language (HTML), using a charset encoding system (US-ASCII, UTF-8, etc.), and containing information and hyper textual links;
- ***Multimedia resources***, which are binary objects such as images, videos, PDFs, Microsoft Word documents, etc., which require a dedicated player or viewer in order to access their contents;
- ***Scripts***, the source code of programs that can be executed by a Web client in order to provide dynamic behaviour to a Web page.

Crawling Process:

Crawlers discover new resources *by recursively traversing the Web*. Figure below sketches the typical steps of a crawling process. The crawler picks the resources to fetch from the URL frontier, i.e., a processing queue that contains the URLs corresponding to the resources that have yet to be processed by the crawler. When the crawling process starts up, the URL frontier contains an initial set of seed URLs; the crawler picks one URL and downloads the associated resource, which is then analyzed.

The analysis aims at extracting all the information which might be relevant for crawling and indexing purposes. For instance, when the fetched resource is a Web page, the analysis task.

parses the HTML page so as to extract the contained *text* to be later processed and provided to the textual indexer for textual analysis and indexing. The analysis also retrieves *links* pointing to other resources, which are then added to the URL *frontier*. Indeed, different resource types call for alternative analysis methods. When a resource is fetched and analyzed, it can be either removed from the frontier, or added back for future fetching.



There are mainly two types of crawling strategies as below:

1- **Breadth-First Crawling:** In order to build a wide Web archive like that of the Internet archive, a crawl is carried out from a set of Web pages (initial URLs or seeds). A breadth-first exploration is launched by following hypertext links leading to those pages directly connected with this initial set. In fact, Web sites are not really browsed breadth-first and various restrictions may apply, e.g. Limiting crawling processes to within a site, or downloading the pages deemed most interesting first.

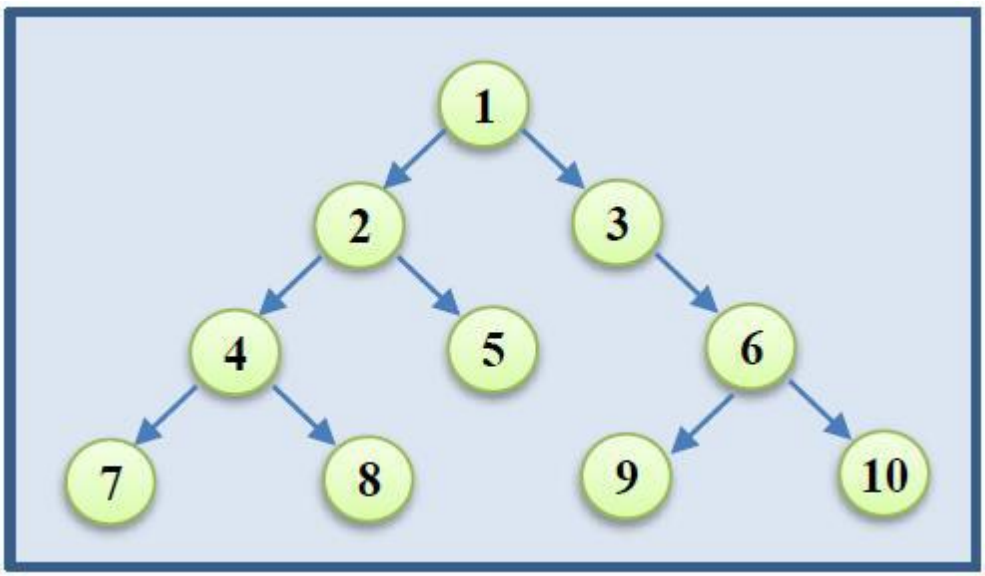


Figure1: Numbers showing path of Breadth-First Crawling

The Breadth-First search algorithm performs the unique search around the neighbor nodes(hyperlinks). It starts by following the root node (Hyperlink) and scans the all the neighbor nodes at the initial level. If the targeted search is achieved, then the scanning is stopped otherwise it leads to the next level.

Such types of algorithms are best suited where the branches are small and resultant objective is identical. When the branches or tree is very deep then this algorithm will not perform well, i.e. All path traversals lead to the same resultant node.

2- Depth-First Crawling:

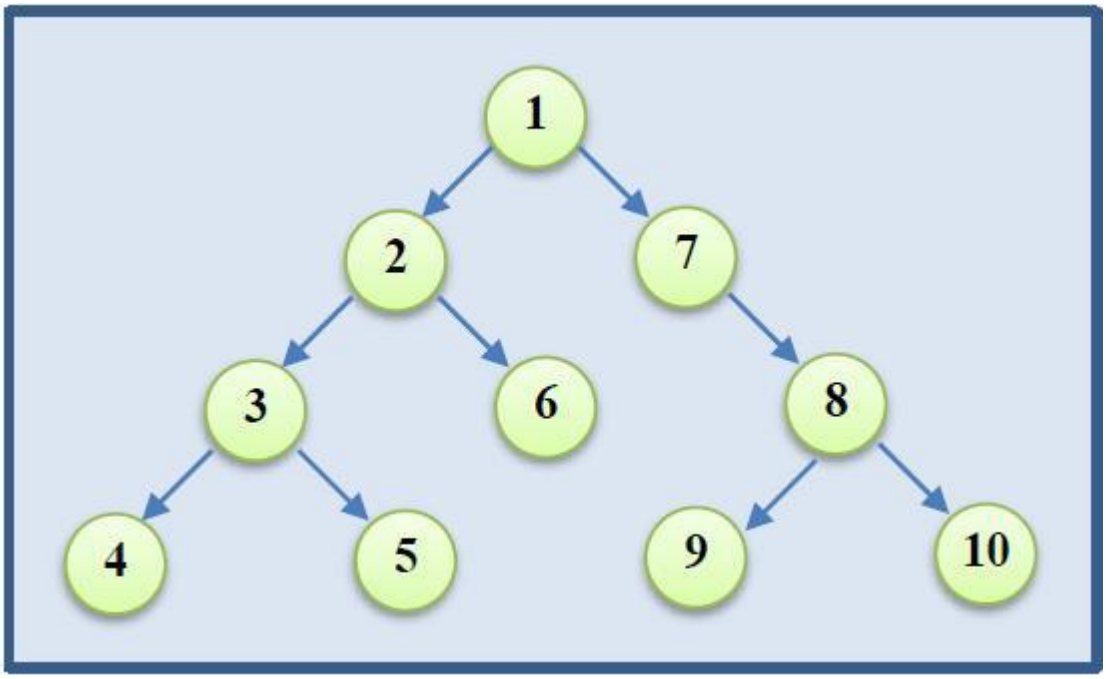


Figure2: Numbers showing path of Depth-First Crawling

The Depth First search algorithm starts searching the objective from the root node and traverse next to its child node, if there are more than one child node, then left most node is given highest priority and traverse deep until no more child node is present. Then it starts from the next unvisited node and then continues in a similar manner.

By using this algorithm, the assurance of scanning of all node is achieved but when the number of child node is large then this algorithm takes more time and might go in to infinite.

Architecture of a crawler:

1- **FRONTIER:**

The frontier is the to-do list or a vector of a crawler that **contains the unique URLs of unvisited pages**. In graph search terminology the frontier is an open list of unexpanded (unvisited) nodes.

The frontier may be implemented as **a FIFO queue** in which case we have a breadth first or depth first crawler that can be used to blindly crawl the Web. The URL to crawl next comes from the head of the queue and the new URLs are added to the tail of the queue.

2- **DNS resolution:**

Each web server (and indeed any host connected to the internet) has a unique IP address: a sequence of four bytes generally represented IP address as four integers separated by dots; for instance, 207.142.131.248 is the numerical IP address associated with the hostwww.wikipedia.org. Translating URL to an IP address is a process known as **DNS resolution** or **DNS lookup.**

During DNS resolution the web crawler contacts a DNS server that returns the translated IP address.

3- **FETCHING:** A fetch module that uses the **http protocol** to retrieve the web page at a URL.

4- **PARSING:** A parsing module that extracts the text and set of links from a fetched web page.

Once a page has been fetched, we need to parse its content to extract information that will feed and possibly guide the future path of the crawler. Parsing may imply *simple hyperlink/URL extraction* or it may involve the more complex process of *tidying up the HTML content in order to analyze the HTML tag tree.* Parsing might also involve *steps to convert the extracted URL to a canonical form,* remove stop words from the page's content and stem the remaining words.

5- **Duplicate Elimination:**

Many resources on the Web are available under multiple URLs or mirrored on multiple servers, thus causing any Web crawler to download (and analyze) the same resource's contents multiple times.

A duplicate elimination module determines whether an extracted link is already in the URL frontier or has recently been fetched and decide if the resource has already been processed by the system.

Crawling policy: The behavior of a Web crawler is the outcome of a combination of policies:

- a *selection policy* which states the pages to download,
- a *re-visit policy* which states when to check for changes to the pages,
- a *politeness policy* that states how to avoid overloading Web sites.
- a *parallelization policy* that states how to coordinate distributed web crawlers.

Metadata: data that provides information about other data, three distinct types of metadata exist

1- Descriptive metadata describes an information resource for identification and retrieval through elements such as title, author, and abstract.

2- ***Structural metadata*** is metadata about containers of data and indicates how compound objects are put together (e.g., how pages are put together to form chapters).

3- ***Administrative metadata*** provides information to help manage a resource, such as when and how it was created, file type and other technical information, and who can access it.

Navigating the Web:

Navigation tools assist users in their surfing tasks. Some are ***built in to the browser*** or ***provided through plug-ins***, and ***others are delivered through web sites*** in order to help users locate information local to the sites.

Browser: is the software component which requests and displays web pages for the user to inspect.

HTML and web site design:

HTML allows web site designers to create web pages containing text, graphics and hyperlinks (normally referred to simply as links), which are suitable for being displayed in a web browser. With HTML a web site designer can structure web pages, in analogy to a word processor being used to stylize and edit documents.

The basic browser tools:

1- ***The back and forward buttons:***

The mechanism underlying the back button is stack based. What this means is that *each time a user clicks on a link, the address of the web page loaded into the browser*.

2- ***Search engine toolbars:***

allows surfers to submit a query to the search engine directly from any page they are browsing, either by typing in the query in the toolbar's search box or by selecting text on the web page with the mouse.

3- ***The bookmarks tool:***

Is a standard browser tool, it allows surfers to create shortcuts in the form of links to web pages they wish to return to on demand. The bookmarks can be viewed on a popup menu or, alternatively, in a side window within the browser by clicking on the favourites button on the standard toolbar.

4- ***The history list:***

The history list contains the links of all the web pages you have visited up to a limited number of days according to your preference.

Web Data Mining:

Data mining is concerned with *finding useful and interesting patterns in large data sets*. Often the term data mining is accompanied with the phrase "*knowledge discovery*" as an indication of the intent to convert raw data or information into knowledge. which is more meaningful, compact and understandable. In web data mining the data source is the web. We can view web mining from three perspectives depending on the data that is being scrutinized.

Three perspectives on data mining:

1- ***Content mining*** deals with the information contained in web documents. Web pages may contain diverse types of data such as text, images, audio and video.

2- ***Structure mining*** is concerned with link analysis, and with the analysis of the structure of individual documents as carried out by search engines.

3- ***Usage mining*** attempts to discover patterns in web data, as collected from web server log data, from a proxy server log or from the user's browser.

Markup Language:

A markup language is a computer language that uses tags to define elements within a document. It is human-readable, meaning markup files contain standard words, rather than typical programming syntax. While several markup languages exist, the two most popular are HTML and XML.

HTML is a markup language used for creating webpages. The contents of each webpage are defined by HTML tags. Basic page tags, such as <head>, <body>, and define sections of the page, while tags such as <table>, <form>, <image>, and <a> define elements within the page. Most elements require a beginning and end tag, with the content placed between the tags. For example, a link to the TechTerms.com home page may use the following HTML code:

TechTerms.com

XML is used for storing structured data, rather than formatting information on a page. While HTML documents use predefined tags (like the examples above), XML files use custom tags to define elements. For example, an XML file that stores information about computer models may include the following section:

```
<computer>
<manufacturer>Dell</manufacturer>
<model>XPS 17</model>
<components>
<processor>2.00 GHz Intel Core i7</processor>
<ram>6GB</ram>
<storage>1TB</storage>
</components>
</computer>
```

XML is called the "Extensible Markup Language" since custom tags can be used to support a wide range of elements. *Each XML file is saved in a standard text format, which makes it easy for software programs to parse or read the data.* Therefore, XML is a common choice for exporting structured data and for sharing data between multiple programs.