1.1 INTRODUCTION:

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).

As defined in this way, information retrieval used to be an activity that only a few people engaged in: reference librarians, paralegals, and similar professional searchers. Now the world has changed, and hundreds of millions of people engage in information retrieval every day when they use a web search engine or search their email. Information retrieval is fast becoming the dominant form of information access, overtaking traditional database-style searching.

IR can also cover other kinds of data and information problems beyond that specified in the core definition above. The term “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, of the sort companies usually use to maintain product inventories and personnel records. In reality, almost no data are truly “unstructured”. This is definitely true of all text data if you count the latent linguistic structure of human languages. But even accepting that the intended notion of structure is open structure, most text has structure, such as headings and paragraphs and footnotes, which is commonly represented in documents by explicit markup (such as the coding underlying web pages). IR is also used to facilitate “semi structured”
search such as finding a document where the title contains Java and the body contains threading.

The field of information retrieval also covers supporting users in **browsing or filtering** document collections or further processing a set of retrieved documents. Given a set of documents, **clustering** is the task of coming up with a good grouping of the documents based on their contents. It is similar to arranging books on a bookshelf according to their topic. Given a set of topics, standing information needs, or other categories (such as suitability of texts for different age groups), **classification** is the task of deciding which class(es), if any, each of a set of documents belongs to. It is often approached by first manually classifying some documents and then hoping to be able to classify new documents automatically.

Information retrieval systems can also be distinguished by the scale at which they operate, and it is useful to distinguish three prominent scales. In web search, the system has to provide search over billions of documents stored on millions of computers. Distinctive issues need to gather documents for indexing, being able to build systems that work efficiently at this enormous scale, and handling particular aspects of the web, such as the exploitation of hypertext and not being fooled by site providers manipulating page content in an attempt to boost their search engine rankings, given the commercial importance of the web.

In the last few years, consumer operating systems have integrated information retrieval (such as Apple’s Mac OS X Spotlight or Windows Vista’s Instant Search). Email programs usually not only provide search but also text classification: they at least provide a spam (junk mail) filter, and commonly also provide either manual or automatic means for classifying mail so that it can be placed directly into particular folders. Distinctive issues here include handling the broad range of document types on a typical personal computer, and making the search system maintenance free and sufficiently lightweight in terms of start-up, processing, and disk space usage that it can run on one machine without annoying its owner. In between is the space of enterprise, institutional, and domain-specific search, where retrieval might be provided for collections such as a corporation’s internal documents, a database of patents, or research articles on biochemistry. In this case, the
documents will typically be stored on centralized file systems and one or a handful of
dedicated machines will provide search over the collection.

1.2 History of IR:

The idea of using computers to search for relevant pieces of information was
popularized in the article “As We May Think” by Vannevar Bush in 1945. It would appear
that Bush was inspired by patents for a 'statistical machine' - filed by Emanuel Goldberg in
the 1920s and '30s - that searched for documents stored on film. The first description of a
computer searching for information was described by Holmstrom in 1948, detailing an early
mention of the Univac computer.

Automated information retrieval systems were introduced in the 1950s: one even
featured in the 1957 romantic comedy, Desk Set. In the 1960s, the first large information
retrieval research group was formed by Gerard Salton at Cornell. By the 1970s several
different retrieval techniques had been shown to perform well on small text corpora such as
the Cranfield collection (several thousand documents). Large-scale retrieval systems, such as
the Lockheed Dialog system, came into use early in the 1970s.

In 1992, the US Department of Defense along with the National Institute of
Standards and Technology (NIST), cosponsored the Text Retrieval Conference (TREC) as part
of the TIPSTER text program. The aim of this was to look into the information retrieval
community by supplying the infrastructure that was needed for evaluation of text retrieval
methodologies on a very large text collection. This catalyzed research on methods that scale
to huge corpora. The introduction of web search engines has boosted the need for very
large scale retrieval systems even further.

1.2.1. Timeline:

1950s:-

1950: The term "information retrieval" was coined by Calvin Mooers.

1951: Philip Bagley conducted the earliest experiment in computerized document retrieval
in a master thesis at MIT.
1955: Allen Kent joined from Western Reserve University published a paper in American Documentation describing the **precision and recall** measures as well as detailing a proposed "framework" for evaluating an IR system which included statistical sampling methods for determining the number of relevant documents not retrieved.

1959: Hans Peter Luhn published "Auto-encoding of documents for information retrieval."

**1960s:**

early 1960s: **Gerard Salton** began work on IR at Harvard, later moved to **Cornell.**

1963:


1964:

- Karen Spärck Jones finished her thesis at Cambridge, Synonymy and Semantic Classification, and continued work on computational linguistics as it applies to IR.
- **The National Bureau of Standards sponsored a symposium titled "Statistical Association Methods for Mechanized Documentation."** Several highly significant papers, including G. Salton’s first published reference (we believe) to the SMART system.

**mid-1960s:**

- National Library of Medicine developed MEDLARS Medical Literature Analysis and Retrieval System, the first major machine-readable database and **batch-retrieval system.**
- Project Intrex at MIT.


late 1960s: F. Wilfrid Lancaster completed evaluation studies of the MEDLARS system and published the first edition of his text on information retrieval.

John W. Sammon, Jr.'s RADC Tech report "Some Mathematics of Information Storage and Retrieval..." outlined the vector model.

1969: Sammon's "A nonlinear mapping for data structure analysis" (IEEE Transactions on Computers) was the first proposal for visualization interface to an IR system.

**1970s**

early 1970s:

First online systems—NLM’s AIM-TWX, MEDLINE; Lockheed's Dialog; SDC’s ORBIT.

1971: Nicholas Jardine and Cornelis J. van Rijsbergen published "The use of hierarchic clustering in information retrieval", which articulated the "cluster hypothesis."

1975: Three highly influential publications by Salton fully articulated his vector processing framework and term discrimination model:

A Theory of Indexing (Society for Industrial and Applied Mathematics)

A Theory of Term Importance in Automatic Text Analysis (JASIS v. 26)

A Vector Space Model for Automatic Indexing (CACM 18:11)


1979: C. J. van Rijsbergen published Information Retrieval (Butterworths). Heavy emphasis on probabilistic models.

1979: Tamas Doszkocs implemented the CITE natural language user interface for MEDLINE at the National Library of Medicine. The CITE system supported free form query input, ranked output and relevance feedback.

**1980s**

1982: Nicholas J. Belkin, Robert N. Oddy, and Helen M. Brooks proposed the ASK (Anomalous State of Knowledge) viewpoint for information retrieval. This was an important concept, though their automated analysis tool proved ultimately disappointing.

mid-1980s: Efforts to develop end-user versions of commercial IR systems.

1989: First World Wide Web proposals by Tim Berners-Lee at CERN.

1990s


1997: Publication of Korfhage's Information Storage and Retrieval with emphasis on visualization and multi-reference point systems.

late 1990s: Web search engines implementation of many features formerly found only in experimental IR systems. Search engines become the most common and maybe best instantiation of IR models.

2000s-present:

More applications, especially Web search and interactions with other fields like Learning to rank, Scalability (e.g., MapReduce), Real-time search

1.3 COMPONENTS OF IR:

The Following figure shows the architecture of IR System
Components:

- Text operations
- Indexing
- Searching
- Ranking
- User Interface
- Query operations

Text operation:

Text Operations forms index words (tokens).

- Stop word removal, Stemming

Indexing:

Indexing constructs an inverted index of word to document pointers.

Searching:

Searching retrieves documents that contain a given query token from the inverted index.

Ranking:

Ranking scores all retrieved documents according to a relevance metric.

User Interface:

User Interface manages interaction with the user:

- Query input and document output.
- Relevance feedback.
- Visualization of results.

Query Operations:

Query Operations transform the query to improve retrieval:

- Query expansion using a thesaurus.
Query transformation using relevance feedback.

First of all, before the retrieval process can even be initiated, it is necessary to define the text database. This is usually done by the manager of the database, which specifies the following: (a) the documents to be used, (b) the operations to be performed on the text, and (c) the text model (i.e., the text structure and what elements can be retrieved). The text operations transform the original documents and generate a logical view of them.

Once the logical view of the documents is defined, the database manager builds an index of the text. An index is a critical data structure because it allows fast searching over large volumes of data. Different index structures might be used, but the most popular one is the inverted file. The resources (time and storage space) spent on defining the text database and building the index are amortized by querying the retrieval system many times.

Given that the document database is indexed, the retrieval process can be initiated. The user first specifies a user need which is then parsed and transformed by the same text operations applied to the text. Then, query operations might be applied before the actual query, which provides a system representation for the user need, is generated. The query is then processed to obtain the retrieved documents. Fast query processing is made possible by the index structure previously built.

Before been sent to the user, the retrieved documents are ranked according to a likelihood of relevance. The user then examines the set of ranked documents in the search for useful information. At this point, he might pinpoint a subset of the documents seen as definitely of interest and initiate a user feedback cycle. In such a cycle, the system uses the documents selected by the user to change the query formulation. Hopefully, this modified query is a better representation
1.4. **ISSUES:**

1. To process large document collections quickly. The amount of online data has grown at least as quickly as the speed of computers, and we would now like to be able to search collections that total in the order of billions to trillions of words.

2. To allow more flexible matching operations. For example, it is impractical to perform the query Romans NEAR countrymen with `grep`, where NEAR might be defined as “within 5 words” or “within the same sentence”.

3. To allow ranked retrieval: in many cases you want the best answer to an information need among many documents that contain certain words.

**The Big Issues**

Information retrieval researchers have focused on a few key issues that remain just as important in the era of commercial web search engines working with billions of web pages as they were when tests were done in the 1960s on document collections containing about 1.5 megabytes of text. One of these issues is relevance.

**Relevance** is a fundamental concept in information retrieval. Loosely speaking, a relevant document contains the information that a person was looking for when she submitted a query to the search engine. Although this sounds simple, there are many factors that go into a person’s decision as to whether a particular document is relevant. These factors must be taken into account when designing algorithms for comparing text and ranking documents. Simply comparing the text of a query with the text of a document and looking for an exact match, as might be done in a database system or using the `grep` utility in Unix, produces very poor results in terms of relevance. One obvious reason for this is that language can be used to express the same concepts in many different ways, often with very different words. This is referred to as the **vocabulary mismatch problem** in information retrieval. It is also important to distinguish between **topical relevance** and **user relevance**. A text document is topically relevant to a query if it is on the same topic. For example, a news story about a cyclone in Nagercoil would be topically relevant to the query “severe weather events”. The person who asked the question (often called the user) may not consider the story relevant, however, if she has seen that story before, or if the story is five years old, or
if the story is in Chinese from a Chinese news agency. User relevance takes these additional features of the story into account. To address the issue of relevance, researchers propose retrieval models and test how well they work. A retrieval model is a formal representation of the process of matching a query and a document. It is the basis of the ranking algorithm that is used in a search engine to produce the ranked list of documents. A good retrieval model will find documents that are likely to be considered relevant by the person who submitted the query. Some retrieval models focus on topical relevance, but a search engine deployed in a real environment must use ranking algorithms that incorporate user relevance. An interesting feature of the retrieval models used in information retrieval is that they typically model the statistical properties of text rather than the linguistic structure. This means, for example, that the ranking algorithms are typically far more concerned with the counts of word occurrences than whether the word is a noun or an adjective. More advanced models do incorporate linguistic features, but they tend to be of secondary importance. The use of word frequency information to represent text started with another information retrieval pioneer, H.P. Luhn, in the 1950s. This view of text did not become popular in other fields of computer science, such as natural language processing, until the 1990s.

Another core issue for information retrieval is evaluation. Since the quality of a document ranking depends on how well it matches a person’s expectations, it was necessary early on to develop evaluation measures and experimental procedures for acquiring this data and using it to compare ranking algorithms. Cyril Cleverdon led the way in developing evaluation methods in the early 1960s, and two of the measures he used, precision and recall, are still popular. Precision is a very intuitive measure, and is the proportion of retrieved documents that are relevant. Recall is the proportion of relevant documents that are retrieved. When the recall measure is used, there is an assumption that all the relevant documents for a given query are known. Such an assumption is clearly problematic in a web search environment, but with smaller test collections of documents, this measure can be useful. A test collection for information retrieval experiments consists of a collection of text documents, a sample of typical queries, and a list of relevant documents for each query (the relevance judgments). The best-known test collections are those associated with the TREC6 evaluation forum. Evaluation of retrieval models and search engines is a very active area, with much of the current focus on using large volumes
of log data from user interactions, such as click through data, which records the documents that were clicked on during a search session. Click through and other log data is strongly correlated with relevance so it can be used to evaluate search, but search engine companies still use relevance judgments in addition to log data to ensure the validity of their results.

The third core issue for information retrieval is the **emphasis on users and their information needs**. This should be clear given that the evaluation of search is user centered. That is, the users of a search engine are the ultimate judges of quality. This has led to numerous studies on how people interact with search engines and, in particular, to the development of techniques to help people express their information needs. **An information need is the underlying cause of the query that a person submits to a search engine.** In contrast to a request to a database system, such as for the balance of a bank account, text queries are often poor descriptions of what the user actually wants. A one-word query such as “cats” could be a request for information on where to buy cats or for a description of the Broadway musical. Despite their lack of specificity, however, one-word queries are very common in web search. Techniques such as query suggestion, query expansion, and relevance feedback use interaction and context to refine the initial query in order to produce better ranked lists.

**1.5 THE IMPACT OF THE WEB ON IR**

With the rapid growth of the Internet, more information is available on the Web and Web information retrieval presents additional technical challenges when compared to classic information retrieval due to the **heterogeneity and size of the web.**

Web information retrieval is unique due to the **dynamism, variety of languages used, duplication, high linkage, ill formed query and wide variance in the nature of users.**

Another issue is the **rising number of inexperienced users**, they the majority of Web users are not very sophisticated searchers.

Many software tools are available for web information retrieval such as search engines (Google and Alta vista), hierarchical directories (Yahoo), many other software agents and collaborative filtering systems. The commonly cited problems in search engines are the slow speed of retrieval, communication delays, and poor quality of retrieved results.
In addition, in the current scenario, multimedia information is increasingly becoming available on the Web and modem IR systems must be capable of tackling not only the text but also multimedia information like sound, image and video.

1.6 OPEN SOURCE SEARCH ENGINE FRAMEWORKS

When deciding to install a search engine in a website, there exists the possibility to use a commercial search engine or an open source one. For most of the websites, using a commercial search engine is not a feasible alternative because of the fees that are required and because they focus on large scale sites. On the other hand, open source search engines may give the same functionalities (some are capable of managing large amount of data) as a commercial one, with the benefits of the open source philosophy: no cost, software maintained actively, possibility to customize the code in order to satisfy personal needs, etc.

Nowadays, there are many open source alternatives that can be used, and each of them have different characteristics that must be taken into consideration in order to determine which one to install in the website. These search engines can be classified according to the programming language in which it is implemented, how it stores the index (inverted file, database, other file structure), its searching capabilities (boolean operators, fuzzy search, use of stemming, etc), way of ranking, type of files capable of indexing (HTML, PDF, plain text, etc), possibility of on-line indexing and/or making incremental indexes.

Example:

There are several open source search engines available.


**Nutch: (A Flexible and Scalable Open-Source Web Search Engine)**

Nutch is an open-source Web search engine that can be used at global, local, and even personal scale. Its initial design goal was to enable a transparent alternative for global Web search in the public interest — one of its signature features is the ability to “explain”
its result rankings. Recent work has emphasized how it can also be used for intranets; by local communities with richer data models, such as the Creative Commons metadata-enabled search for licensed content; on a personal scale to index a user's files, email, and web-surfing history;

**Architecture:**

Nutch has a highly modular architecture that uses plug-in APIs for media-type parsing, HTML analysis, data retrieval protocols, and queries. The core has four major components.

a) **Searcher:**

Given a query, it must *quickly find a small relevant subset of a corpus of documents*, then present them. Finding a large relevant subset is normally done with an inverted index of the corpus; ranking within that set to produce the most relevant documents, which then must be summarized for display.

b) **Indexer:**

Creates the inverted index from which the searcher extracts results. It uses *Lucene storing indexes*.

c) **Database:**

Stores the document contents for indexing and later summarization by the searcher, along with information such as the link structure of the document space and the time each document was last fetched.

d) **Fetcher:**

Requests web pages, parses them, and extracts links from them. *Nutch’s robot* has been written entirely from scratch.

**Figure:** Layer diagram of Nutch package dependencies.
Crawling:

An intranet or niche search engine might only take a single machine a few hours to crawl, while a whole-web crawl might take many machines several weeks or longer. A single crawling cycle consists of generating a fetch list from the webdb, fetching those pages, parsing those for links, then updating the webdb. It also uses a uniform refresh policy; all pages are refetched at the same interval (30 days, by default).

Indexing Text:

Lucene meets the scalability requirements for text indexing in Nutch. Nutch also takes advantage of Lucene’s multi-field case-folding keyword and phrase search in URLs, anchor text, and document text.

Indexing Hypertext:

Lucene provides an inverted-file full-text index, which suffices for indexing text but not the additional tasks required by a web search engine. In addition to this, Nutch implements a link database to provide efficient access to the Web’s link graph, and a page database that stores crawled pages for indexing, summarizing, and serving to users, as well as supporting other functions such as crawling and link analysis.

Removing Duplicates
The nutch **dedup command** eliminates duplicate documents from a set of Lucene indices for Nutch segments, so it inherently requires access to all the segments at once. It's a batch-mode process that has to be run before running searches to prevent the search from returning duplicate documents.

**Link Analysis:**

Nutch includes a link analysis algorithm similar to PageRank. Distributed link analysis is a bulk synchronous parallel process. At the beginning of each phase, the list of URLs whose scores must be updated is divided up into many chunks; in the middle, many processes produce score-edit files by finding all the links into pages in their particular chunk. At the end, an updating phase reads the score-edit files one at a time, merging their results into new scores for the pages in the web database.

**Searching:**

Nutch's search user interface runs as a Java Server Page (JSP) that parses the user's textual query and invokes the search method of a NutchBean. *If Nutch is running on a single server, this translates the user's query into a Lucene query and gets a list of hits from Lucene, which the JSP then renders into HTML. If Nutch is instead distributed across several servers, the NutchBean's search method instead remotely invokes the search methods of other NutchBeans on other machines, which can be configured either to perform the search locally as described above or farm pieces of the work out to yet other servers.*

**Summarizing:**

Summaries on a results page are designed to avoid click throughs. By providing as much relevant information as possible in a small amount of text, they help users improve precision. *Nutch's summarizer works by retokenizing a text string containing the entire original document, extracting a minimal set of excerpts containing five words of context on each side of each hit in the document, then deciding which excerpts to include in the final summary.* It orders the excerpts with the best ones first, preferring...
longer excerpts over shorter ones, and excerpts with more hits above excerpts with fewer, and then it truncates the total summary to a maximum of twenty words.

**1.7 The role of artificial intelligence (AI) in IR**

In the early days of computer science, information retrieval (IR) and artificial intelligence (AI) developed in parallel. In the 1980s, they started to cooperate and the term intelligent information retrieval was coined for AI applications in IR. In the 1990s, information retrieval has seen a shift from set based Boolean retrieval models to ranking systems like the vector space model and probabilistic approaches.

Artificial intelligence methods are employed throughout the standard information retrieval process and for novel value added services. The text preprocessing processes for indexing like stemming are from Artificial intelligence.

Neural networks have been applied widely in IR. Several network architectures have been applied for retrieval tasks, most often the so-called **spreading activation networks** are used. Spreading activation networks are simple Hopfield-style networks, however, they do not use the learning rule of Hopfield networks. They typically consist of two layers representing terms and documents. The weights of connections between the layers are bi-directional and initially set according to the results of the traditional indexing and weighting algorithms. Closer look at the models reveals that they very much resemble **the traditional vector space model** of Information Retrieval.

**1.8 IR Versus Web Search:**

Web search is just one important area of information retrieval, but not all Information retrieval also includes, Recommendation, Question Answering, Text mining, Online advertising, Enterprise search: web search and desktop search.
The first important difference is the scale of web search, as we have seen that the current size of the web is approximately 600 billion pages. This is well beyond the size of traditional document collections.

The Web is dynamic in a way that was unimaginable to traditional IR in terms of its rate of change and the different types of web pages ranging from static types to a growing number dynamic pages.

The Web also contains an enormous amount of duplication, estimated at about 30%. Such redundancy is not present in traditional corpora and makes the search engine’s task even more difficult.

The quality of web pages vary dramatically.

The range of topics covered on the Web is completely open, as opposed to the closed collections indexed by traditional IR systems, where the topics such as in library catalogues, are much better defined and constrained.

Another aspect of the Web is that it is globally distributed. This poses serious logistic problems to search engines in building their indexes, and moreover, in delivering a service that is being used from all over the globe.

Users also tend to submit short queries (between two to three keywords), avoid the use of anything but the basic search engine syntax, and when the results list is returned, most users do not look at more than the top 10 results, and are unlikely to modify their query. This is all contrary to typical usage of traditional IR.

The hypertextual nature of the Web is also different from traditional document collections, in giving users the ability to surf by following links.

Comparison:

<table>
<thead>
<tr>
<th>Sl. No</th>
<th>Differentiator</th>
<th>Web Search</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Languages</td>
<td>Documents in many different languages. Usually search engines use full text indexing; no additional subject analysis.</td>
<td>Databases usually cover only one language or indexing of documents written in different languages with the same vocabulary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>2</td>
<td>File types</td>
<td>Several file types, some hard to index because of a lack of textual information.</td>
<td>Usually all indexed documents have the same format (e.g. PDF) or only bibliographic information is provided.</td>
</tr>
<tr>
<td>3</td>
<td>Document length</td>
<td>Wide range from very short to very long. Longer documents are often divided into parts.</td>
<td>Document length varies, but not to such a high degree as with the Web documents</td>
</tr>
<tr>
<td>4</td>
<td>Document structure</td>
<td>HTML documents are semi structures.</td>
<td>Structured documents allow complex field searching</td>
</tr>
<tr>
<td>5</td>
<td>Spam</td>
<td>Search engines have to decide which documents are suitable for indexing.</td>
<td>Suitable document types are defined in the process of database design.</td>
</tr>
<tr>
<td>6</td>
<td>Amount of data, size of databases</td>
<td>The actual size of the Web is unknown. Complete indexing of the whole Web is impossible.</td>
<td>Exact amount of data can be determined when using formal criteria.</td>
</tr>
<tr>
<td>7</td>
<td>Type of queries</td>
<td>Users have little knowledge how to search; very short queries (2-3 words).</td>
<td>Users know the retrieval language; longer, exact queries.</td>
</tr>
<tr>
<td>8</td>
<td>User interface</td>
<td>Easy to use interfaces suitable for laypersons.</td>
<td>Normally complex interfaces; practice needed to conduct searches.</td>
</tr>
<tr>
<td>9</td>
<td>Ranking</td>
<td>Due to the large amount of hits relevance ranking is the norm.</td>
<td>Relevance ranking is often not needed because the users know how to constrain the amount of hits.</td>
</tr>
<tr>
<td>10</td>
<td>Search functions</td>
<td>Limited possibilities.</td>
<td>Complex query languages allow narrowing searches.</td>
</tr>
</tbody>
</table>
1.8. COMPONENTS OF A SEARCH ENGINE:

Search engine components support two major functions, which we call the indexing process and the query process. The indexing process builds the structures that enable searching, and the query process uses those structures and a person’s query to produce a ranked list of documents.

**Indexing Process:**

The following Figure 1.8.1 shows the high-level “building blocks” of the indexing process. These major components are **text acquisition, text transformation, and index creation**.

a) **Text acquisition:**

The task of the text acquisition component is to **identify and make available the documents that will be searched**. Although in some cases this will involve simply using an existing collection, text acquisition will more often require **building a collection by crawling or scanning the Web, a corporate intranet, a desktop, or other sources of information**. In addition to passing documents to the next component in the indexing process, the text acquisition component creates a **document data store, which contains the text and metadata for all the documents**. Metadata is information about a document that is not part of the text content, such as the document type (e.g., email or web page), document structure, and other features, such as document length.

b) **The text transformation**

The text transformation component transforms **documents into index terms or features**. Index terms, as the name implies, are the parts of a document that are stored in the index and used in searching. The simplest index term is a word, but not every word may be used for searching. A “feature” is more often used in the field of machine learning to refer to a part of a text document that is used to represent its content, which also describes an index term.

Examples of other types of index terms or features are phrases, names of people, dates, and links in a web page. **Index terms are sometimes simply referred to as**
“terms.” The set of all the terms that are indexed for a document collection is called the index vocabulary.

c) The index creation

The index creation component takes the output of the text transformation component and creates the indexes or data structures that enable fast searching. Given the large number of documents in many search applications, index creation must be efficient, both in terms of time and space.

Indexes must also be able to be efficiently updated when new documents are acquired. Inverted indexes, or sometimes inverted files, are by far the most common form of index used by search engines. **An inverted index, very simply, contains a list for every index term of the documents that contain that index term.** It is inverted in the sense of being the opposite of a document file that lists, for every document, the index terms they contain. There are many variations of inverted indexes, and the particular form of index used is one of the most important aspects of a search engine.

![Fig1.8.1: Indexing Process](image)

**Query Process:**

Figure 1.8.2 shows the building blocks of the query process. The major components are user interaction, ranking, and evaluation.
a) User interaction

The user interaction component provides the interface between the person doing the searching and the search engine. One task for this component is accepting the user’s query and transforming it into index terms. Another task is to take the ranked list of documents from the search engine and organize it into the results shown to the user.

This includes, for example, generating the snippets used to summarize documents. The document data store is one of the sources of information used in generating the results. Finally, this component also provides a range of techniques for refining the query so that it better represents the information need.

b) Ranking

The ranking component is the core of the search engine. It takes the transformed query from the user interaction component and generates a ranked list of documents using scores based on a retrieval model. Ranking must be both efficient, since many queries may need to be processed in a short time, and effective, since the quality of the ranking determines whether the search engine accomplishes the goal of finding relevant information. The efficiency of ranking depends on the indexes, and the effectiveness depends on the retrieval model.

c) Evaluation

The task of the evaluation component is to measure and monitor effectiveness and efficiency. An important part of that is to record and analyze user behaviour using log data. The results of evaluation are used to tune and improve the ranking component. Most of the evaluation component is not part of the online search engine, apart from logging user and system data. Evaluation is primarily an offline activity, but it is a critical part of any search application.
1.9 WEB CHARACTERISTICS

The essential feature that led to the explosive growth of the web decentralized content publishing with essentially no central control of authorship turned out to be the biggest challenge for web search engines in their quest to index and retrieve this content.

Web page authors created content in dozens of (natural) languages and thousands of dialects, thus demanding many different forms of stemming and other linguistic operations.

Trust of Web: The democratization of content creation on the web meant a new level of granularity in opinion on virtually any subject. This meant that the web contained truth, lies, contradictions and suppositions on a grand scale. This gives rise to the question: which web pages does one trust? In a simplistic approach, one might argue that some publishers are trustworthy and others not begging the question of how a search engine is to assign such a measure of trust to each website or web page.

Size: While the question “how big is the Web?” has no easy answer.

Static Vs Dynamic: Static web pages are those whose content does not vary from one request for that page to the next. For this purpose, a professor who manually updates his
home page every week is considered to have a static web page, but an airport’s flight status page is considered to be dynamic. **Dynamic pages** are typically mechanically generated by an application server in response to a query to a database.

![Dynamic web page generation](image)

**Fig:** Dynamic web page generation

**The web graph:** We can view the static Web consisting of static HTML pages together with the hyperlinks between them as a directed graph in which each web page is a node and each hyperlink a directed edge.

![Two nodes of web graph joined by link](image)

**Fig:** Two nodes of web graph joined by link

Figure shows two nodes A and B from the web graph, each corresponding to a web page, with a hyperlink from A to B. We refer to the set of all such nodes and directed edges as the web graph. This text is generally encapsulated in the href attribute of the `<a>` (for anchor) tag that encodes the hyperlink in the HTML code of page A, and is referred to as anchor text. As one might suspect, this directed graph is **not strongly connected**: there are pairs of pages such that one cannot proceed from one page of the pair to the other by following hyperlinks. We refer to the hyperlinks into a page as **in-links** and those out of a page as **out-links**. The number of in-links to a page (also known as its in-degree) has averaged from roughly 8 to 15, in a range of studies. We similarly define the **out-degree of a web page to be the number of links out of it**.
There is ample evidence that these links are not randomly distributed; this distribution is widely reported to be a power law, in which the total number of web pages with in-degree $i$ is proportional to $1/i^\alpha$; the value of $\alpha$ typically reported by studies ($\alpha=1$).

The directed graph connecting web pages has a bowtie shape: there are three major categories of web pages that are sometimes referred to as IN, OUT and SCC (Strongly Connected Component). A web surfer can pass from any page in IN to any page in SCC, by following hyperlinks. Likewise, a surfer can pass from page in SCC to any page in OUT. Finally, the surfer can surf from any page in SCC to any other page in SCC. However, it is not possible to pass from a page in SCC to any page in IN, or from a page in OUT to a page in SCC. The remaining pages form into tubes that are small sets of pages outside SCC that lead directly from IN to OUT, and tendrils that either lead nowhere from IN, or from nowhere to OUT.
Spam:

Web search engines were an important means for connecting advertisers to prospective buyers. A user searching for **maui golf real estate** is not merely seeking news or entertainment on the subject of housing on golf courses on the island of Maui, but instead likely to be seeking to purchase such a property. Sellers of such property and their agents, therefore, have a strong incentive to create web pages that rank highly on this query. In a **search engine whose scoring was based on term frequencies**, a **web page with numerous repetitions of maui golf real estate would rank highly**. This led to the first generation of **spam** in which is the manipulation of web page content for the purpose of appearing high up in search results for selected keywords.

**Spammers** resorted to such tricks as rendering these repeated terms in the same colour as the background. Despite these words being consequently invisible to the human user, a search engine indexer would parse the invisible words out of the HTML representation of the web page and index these words as being present in the page.

**Paid Inclusion**: in many search engines, it is possible to pay to have one’s web page included in the search engine’s index – a model known as **paid inclusion**. Different search engines have different policies on whether to allow paid inclusion, and whether such a payment has any effect on ranking in search results.

**Cloaking**: Spammers responded with a richer set of spam techniques like cloaking. Here, the spammer’s web server returns different pages depending on whether the http request comes from a web search engine’s crawler or from a human user’s browser.

**Fig**: Cloaking used by spammers
A doorway page contains text and metadata carefully chosen to rank highly on selected search keywords. When a browser requests the doorway page, it is redirected to a page containing content of amore commercial nature.

SEO: Search Engine Optimizers or SEOs to provide consultancy services for clients who seek to have their web pages rank highly on selected keywords. Web search engines frown on this business of attempting to decipher and adapt to their proprietary ranking techniques and indeed announce policies on forms of SEO behaviour they do not tolerate.