Sardar Patel University,Balaghat(M.P)
School of Engineering & Technology
Department of CSE
Subject Name: Modern Information Retrival
Subject Code: CSE704(3)
Semester: 7th

UNIT – 1

1.1 Motivation of Information Retrieval

Information retrieval (IR) deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he is interested. Unfortunately, characterization of the user information need is not a simple problem. Consider, for instance, the following hypothetical user information need in the context of the World Wide Web (or just the Web):

Find all the pages (documents) containing information on college tennis teams which:

1. Are maintained by a university in the USA.
2. Participate in the NCAA tennis tournament. To be relevant, the page must include information on the national ranking of the team in the last three years and the email or phone number of the team coach.

Clearly, this full description of the user information need cannot be used directly to request information using the current interfaces of Web search engines. Instead, the user must first translate this information need into a query which can be processed by the search engine (or IR system).

In its most common form, this translation yields a set of keywords (or index terms) which summarizes the description of the user information need. Given the user query, the key goal of an IR system is to retrieve information which might be useful or relevant to the user. The emphasis is on the retrieval of information as opposed to the retrieval of data.

1.2 Information versus Data Retrieval

Data retrieval, in the context of an IR system, consists mainly of determining which documents of a collection contain the keywords in the user query which, most frequently, is not enough to satisfy the user information need. In fact, the user of an IR system is concerned more with retrieving information about a subject than with retrieving data which satisfies a given query. A data retrieval language aims at retrieving all objects which satisfy clearly defined conditions such as those in a regular expression or in a relational algebra expression. Thus, for a data retrieval system, a single erroneous object among a thousand retrieved objects means total failure. For an information retrieval system, however, the retrieved objects might be inaccurate and small errors are likely to go unnoticed. The main reason for this difference is that information retrieval usually deals with natural language text which is not always well structured and could be semantically ambiguous. On the other hand, a data retrieval system (such as a relational database) deals with data that has a well defined structure and semantics.

Data retrieval, while providing a solution to the user of a database system, does not solve the problem of retrieving information about a subject or topic. To be effective in its attempt to satisfy the user information need, the IR system must somehow `interpret' the contents of the information items (documents) in a collection and rank them according to a degree of relevance to the user query. This `interpretation' of a document content involves extracting syntactic and semantic information from the document text and using this information to match the user information need. The difficulty is not only knowing how to extract this information but also knowing how to use it to decide relevance. Thus, the notion of relevance is at the center of information retrieval. In fact, the primary goal of an IR system is to retrieve all the documents which are relevant to a user query while retrieving as few non-relevant documents as possible.

1.3 Basic Concepts

The effective retrieval of relevant information is directly affected both by the user task and by the logical view of the documents adopted by the retrieval system.

1.3.1 The User Task

The user of a retrieval system has to translate his information need into a query in the language provided by the system. With an information retrieval system, this normally implies specifying a set of words which convey the semantics of the information need. With a data retrieval system, a query expression (such as, for instance, a regular expression) is used to convey the constraints that must be satisfied by objects in the answer set. In both cases, we say that the user searches for useful information executing a retrieval task.

Consider now a user who has an interest which is either poorly defined or which is inherently broad. For instance, the user might be interested in documents about car racing in general. In this situation, the user might use an interactive interface to simply look around in the collection for documents related to car racing. For instance, he might find interesting documents about Formula 1 racing, about car manufacturers, or about the `24 Hours of Le Mans.' Furthermore, while reading about the `24 Hours of Le Mans', he might turn his attention to a document which provides directions to Le Mans and, from there, to documents which cover tourism in France. In this situation, we say that the user is browsing the documents in the collection, not searching. It is still a process of retrieving information, but one whose main objectives are not clearly defined in the beginning and whose purpose might change during the interaction with the system.

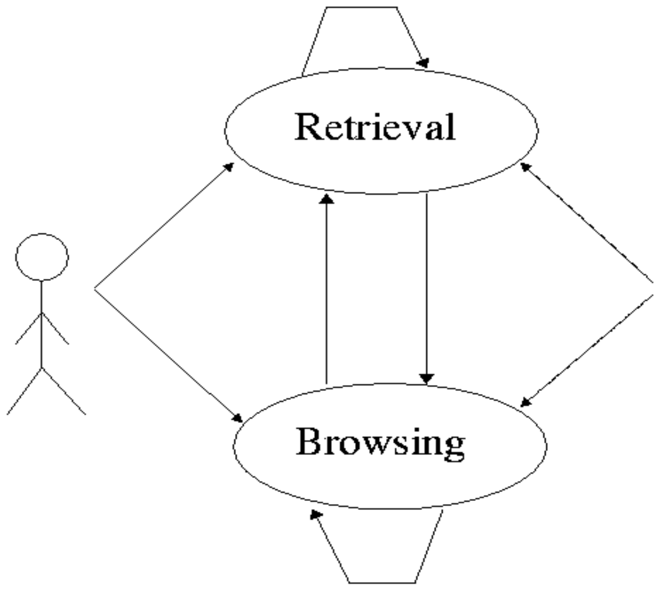


Figure-1 Interaction of the user with the retrieval system through distinct tasks.

Classic information retrieval systems normally allow information or data retrieval. Hypertext systems are usually tuned for providing quick browsing. Modern digital library and Web interfaces might attempt to combine these tasks to provide improved retrieval capabilities. However, combination of retrieval and browsing is not yet a well established approach and is not the dominant paradigm.

Figure 1 illustrates the interaction of the user through the different tasks we identify. Information and data retrieval are usually provided by most modern information retrieval systems (such as Web interfaces). Further, such systems might also provide some (still limited) form of browsing. While combining information and data retrieval with browsing is not yet a common practice, it might become so in the future.

Both retrieval and browsing are, in the language of the World Wide Web, `pulling' actions. That is, the user requests the information in an interactive manner. An alternative is to do retrieval in an automatic and permanent fashion using software agents which push the information towards the user. For instance, information useful to a user could be extracted periodically from a news service. In this case, we say that the IR system is executing a particular retrieval task which consists of filtering relevant information for later inspection by the user.

1.3.2 Logical View of the Documents

Due to historical reasons, documents in a collection are frequently represented through a set of index terms or keywords. Such keywords might be extracted directly from the text of the document or might be specified by a human subject (as frequently done in the information sciences arena). No matter whether these representative keywords are derived automatically or generated by a specialist, they provide a logical view of the document.

Modern computers are making it possible to represent a document by its full set of words. In this case, we say that the retrieval system adopts a full text logical view (or representation) of the documents. With very large collections, however, even modern computers might have to reduce the set of representative keywords. This can be accomplished through the elimination of stop words (such as articles and connectives), the use of stemming (which reduces distinct words to their common grammatical root), and the identification of noun groups (which eliminates adjectives, adverbs, and verbs). Further, compression might be employed. These operations are called text operations (or transformations). Text operations reduce the complexity of the document representation and allow moving the logical view from that of a full text to that of a set of index terms .

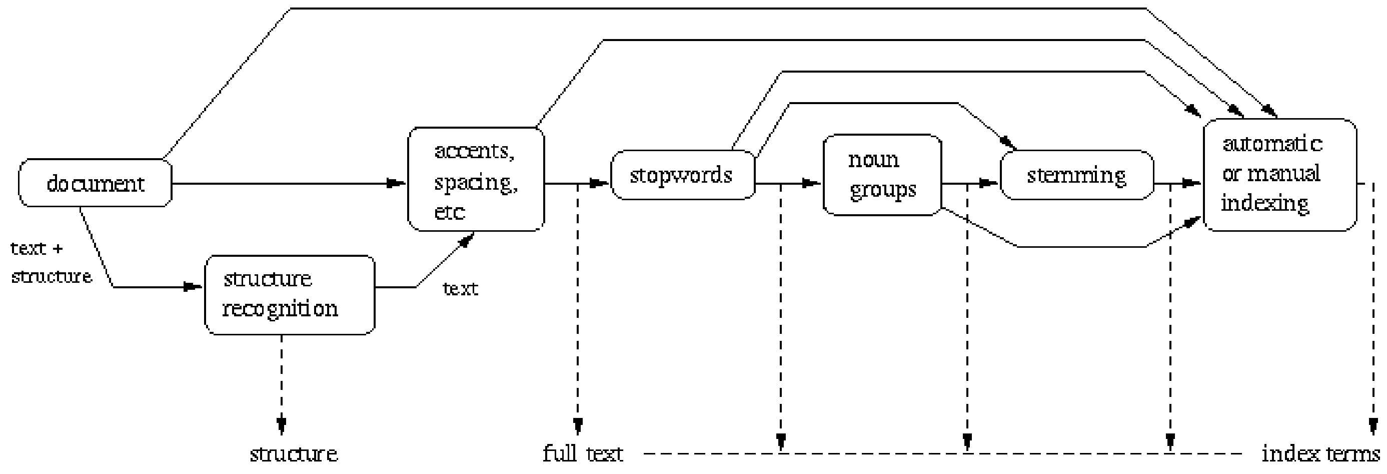


Figure 2: Logical view of a document: from full text to a set of index terms.

The full text is clearly the most complete logical view of a document but its usage usually implies higher computational costs. A small set of categories (generated by a human specialist) provides the most concise logical view of a document but its usage might lead to retrieval of poor quality. Several intermediate logical views (of a document) might be adopted by an information retrieval system as illustrated in **Figure 2**. Besides adopting any of the intermediate representations, the retrieval system might also recognize the internal structure normally present in a document (e.g., chapters, sections, subsections, etc.). This information on the structure of the document might be quite useful and is required by structured text retrieval models.

As illustrated in **Figure 2**, we view the issue of logically representing a document as a continuum in which the logical view of a document might shift (smoothly) from a full text representation to a higher level representation specified by a human subject.

1.4 The Retrieval Process

To describe the retrieval process, we use simple and generic software architecture as shown in **Figure 3**. 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 (using the DB Manager Module) 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 as indicated in **Figure 3**. 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.

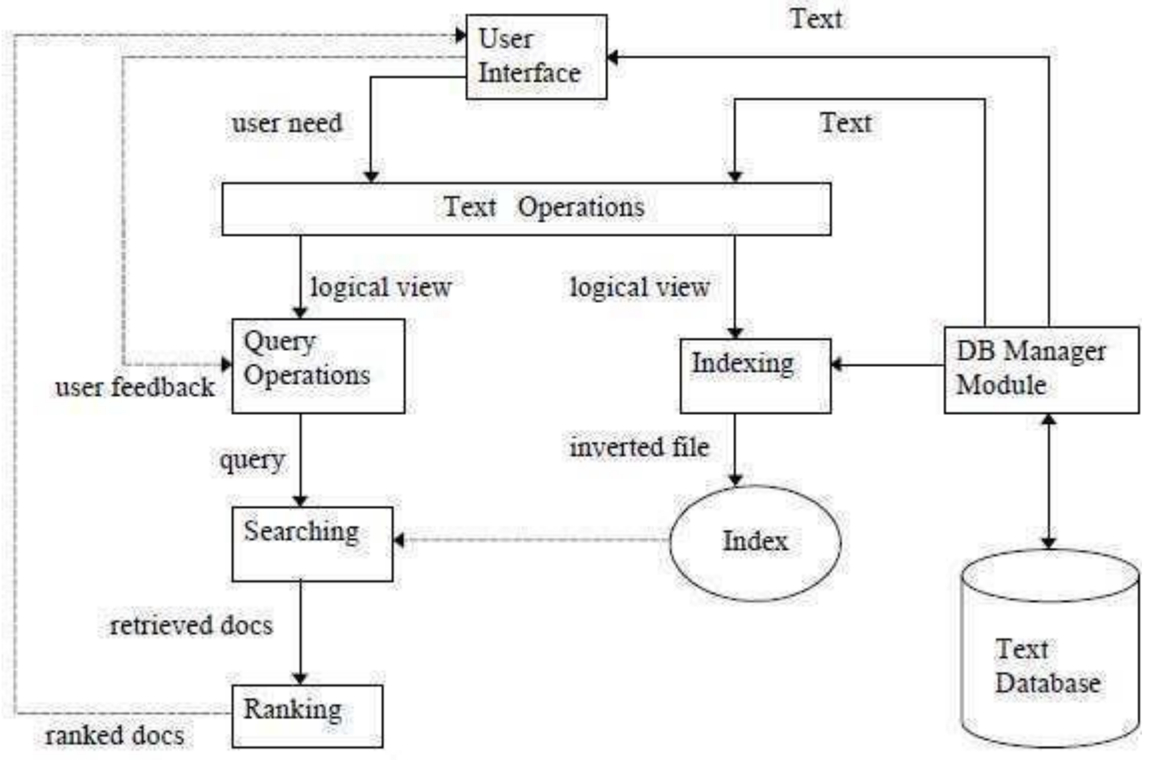


Figure 3: The Process of Retrieving Information

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 of the real user need.

1.5 Taxonomy of Information Retrieval Model

The 3 classic models in information retrieval are called Boolean, vector and probability. In the Boolean model documents and queries are represented as sets of index terms. So we can say that this model is set theoretic.

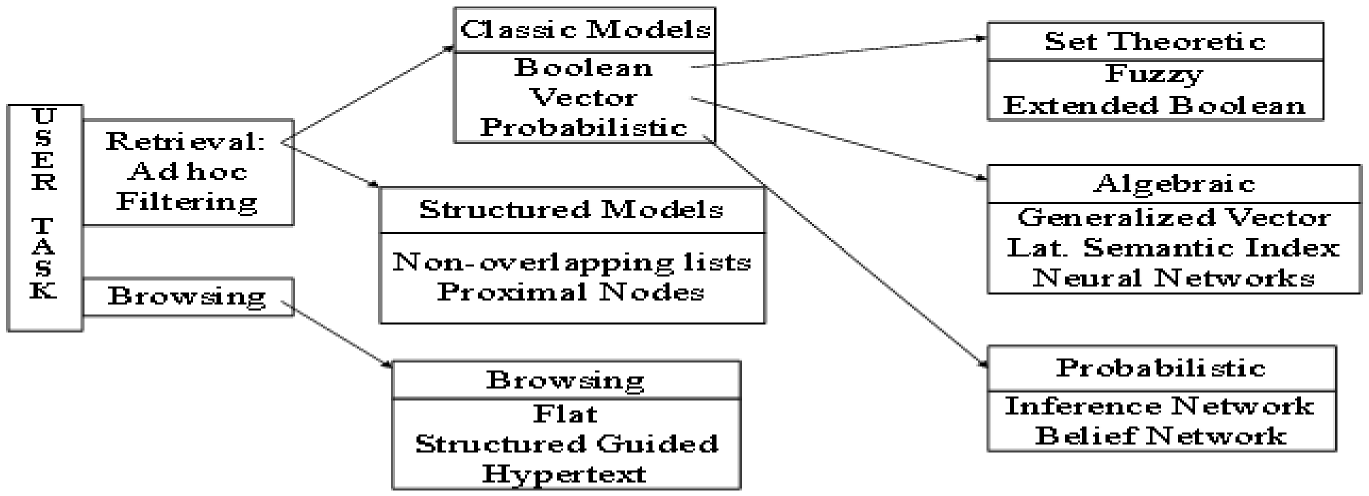


Figure 4: Taxonomy of Information Retrieval Model

In the vector model documents and queries are represented as vectors in t-dimensional space. Thus we can say that model is algebraic. In the probabilistic model, the framework for modelling document and query represe4ntation is based on probability theory. So we can say that the model is probabilistic.
Over the years alternative modelling paradigms for each type of classic model (i.e. set theoretic, algebraic and probabilistic) have been proposed. Regarding alternative set theoretic models, we distinguish the fuzzy and extended boolean models. Regarding alternative algebraic models, we distinguish the generalized vector, latent semantic indexing and the neural network models. Regarding alternative probabilistic models, we distinguish the inference network and belief network models.
Besides references to the text content, the model might also allow references to the structure normally present in written text. In this case, we say that we have structured model. We distinguish two models for structured text retrieval namely, the non overlapping lists model and proximal nodes model.
The user task might be one of browsing. We distinguish three models for browsing namely, flat, structure guided and hypertext.

	Index Terms	Full Text	Full Text+ Structure
Retrieval	Classic Set Theoretic Algebraic Probabilistic	Classic Set Theoretic Algebraic Probabilistic	Structured
Browsing	Flat	Flat Hypertext	Structure Guided Hypertext

Table 1: Retrieval models most frequently associated with distinct combinations of a document logical view and user task.

1.6 Retrieval: Ad hoc and Filtering

In conventional IR system, the documents in the collection remain relatively static while new queries are submitted to the system. This operational mode has been term ad hoc retrieval and is the most common form of user task.

A similar but different task is one in which the queries remain relatively static while new documents come into the system (and leave). This is the case with the stock market and with news wiring services. This operational mode has been termed filtering.

In a filtering task, a user profile describing the user's preferences is constructed. Such a profile is then compared to the incoming documents in an attempt to determine those which might be of interest to this particular user.

A variation of this procedure is to rank the filtered documents and show this ranking to the user. The motivation is that the user can examine a smaller number of documents if he assumes that the ones at the top of the ranking are more likely to be relevant. This variation of filtering is called routing but it is not popular.

UNIT – 2

Syllabus

Classic Information Retrieval Techniques: Boolean Model, Vector Model, Probabilistic Model, comparison of classical models. Introduction to Alternative Algebraic Models: Latent Semantic Indexing Model etc.

A Formal Characterization of IR Models

Definition: An information retrieval model is a quadruple [D, Q, F, R (qi, dj)] where

1. D is a set of logical views (or representations) for the documents in the collection.
2. Q is a set of logical views (or representations) for the user information needs. Such representations are calledqueries.
3. F is a framework for modeling document representations, queries, and their relationships.
4. R (qi, dj) is a ranking function which associates a real number with a query qi ε Q and a document dj ε D. Such ranking defines an ordering among the documents with regards to the query qi.

To build a model, we think first of representations for the documents and for the user information need. Given these representations, we then conceive the framework in which they can be modeled. This framework should also provide the intuition for constructing a ranking function. For instance, for the classic Boolean model, the framework is composed of sets of documents and the standard operations on sets. For the classic vector model, the framework is composed of a t-dimensional vector space and standard linear algebra operations on vectors.For the classic probabilistic model, the framework is composed of sets, standard operations, and bayes' theorem.

Classic Information Retrieval

In this section we briefly present the three classic models in information retrieval namely, the Boolean, the vector and the probabilistic models.

Basic Concepts

The classic model in IR involves the following:

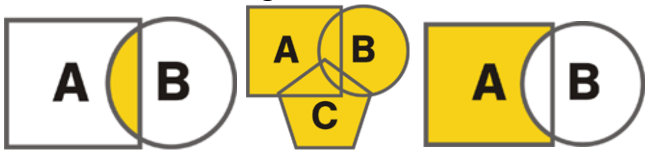
1. Each document is described by a set of representative keywords called index terms. An index term is simply a document word whose semantics help in remembering the main theme of the documents. Thus, index terms are used to index and summarize the document contents.
2. In general, index terms are mainly nouns because nouns have meaning by themselves and thus, their semanticsis easier to identify and to grasp.
3. Adjectives, adverbs, and connectives are less useful as index terms because they work mainly as complements.
4. How to decide which index terms are more important? Think about domain-specific, application- specific terms. A word that appears in all documents is completely useless. But a word that appears many times insome of the documents but only a few in others may be useful!
5. Thus we use weights for each index term of a document. This weight quantifies the importance of the index term for describing the document semantic contents.

IR Models:

1. Boolean Model

The Boolean model is the first model of information retrieval and probably also the most criticized model. The

Boolean model is the first model of information retrieval and probably also the most criticized model. The model can be explained by thinking of a query term as an unambiguous definition of a set of documents. For instance, the query term economic defines the set of all documents that are indexed with the term economical. Using the operators of George Boole's mathematical logic, query terms and their corresponding sets of documents can be combined to form new sets of documents. The Boolean model allows for the use of operators of Boolean algebra, AND, OR and NOT, for query formulation, but has one major disadvantage: a Boolean system is not able to rank the returned list of documents. In the Boolean model, a document is associated with a set of keywords. Queries are also expressions of keywords separated by AND, OR, or NOT/BUT. The retrieval function in this model treats a document as either relevant or irrelevant. In below figure, the retrieved sets are visualized by the shaded areas.



2. **Vector Space Model** Gerard Salton and his colleagues suggested a model based on Luhn's similarity criterion that has a stronger theoretical motivation (Salton and McGill 1983). They considered the index representations and the query as vectors embedded in a high dimensional Euclidean space, where each term is assigned a separate dimension. The vector space model can best be characterized by its attempt to rank documents by the similarity between the query and each document [10].In the Vector Space Model(VSM), documents and query are represent as a Vector, and the angle between the two vectors is computed using the similarity cosine
3. function. Similarity Cosine function can be defined as: Where, , = · ·|||| = ∑ ,, ∑ , ∑ , (1) Documents and queries are represented as vectors. = ,, ,, … , , = ,, ,, … , , Vector Space Model has been introducing a term weight scheme known as if-idf weighting. These weights have a term frequency (tf) factor measuring the frequency of occurrence of the terms in the document or query texts and an inverse document frequency (idf) factor measuring the inverse of the number of documents that contain a query or document term.

4. **Probabilistic Model** Whereas Maron and Kuhns introduced ranking by the probability of relevance; it was Stephen Robertson who turned the idea into a principle. He formulated the probability ranking principle, which he attributed to William Cooper, as follows (Robertson 1977). The most important characteristic of the probabilistic model is its attempt to rank documents by their probability of relevance given a query. Documents and queries are represented by binary vectors ~d and ~q, each vector element indicating whether a document attribute or term occurs in the document or query, or not. Instead of probabilities, the probabilistic model uses odds O(R), where O(R) = P(R)/1 − P(R), R means ‟document is relevant‟ and ¯R means ‟document is not relevant‟. Probability theory can be used to compute a measure of relevance between a query and a document.
   - Simple Term Weights.
   - Non binary independent model.
   - Language model

   3.1 Simple Term Weights: The use of term weights is based on the Probability Ranking Principle (PRP), which assumes that optimal effectiveness occurs when documents are ranked based on an estimate of the probability of their relevance to a query The key is to assign probabilities to components of the query and then use each of these as evidence in computing the final probability that a document is relevant to the query. The terms in the query are assigned weights which correspond to the probability that a particular term, in a match with a given query, will retrieve a relevant document. The weights for each term in the query are

2

combined to obtain a final measure of relevance. Most of the papers in this area incorporate probability theory and describe the validity of independence assumptions, so a brief review of probability theory is in order.

3.2 Non-Binary Independence Model: The non-binary independence model term frequency and document length, somewhat naturally, into the calculation of term weights. Once the term weights are computed, the vector space model is used to compute an inner product for obtaining a final similarity coefficient. The simple term weight approach estimates a term's weight based on whether or not the term appears in a relevant document. Instead of estimating the probability that a given term will identify a relevant document, the probability that a term which appears if times will appear in a relevant document is estimated.

3.3 Language Models: A statistical language model is a probabilistic mechanism for "generating" a piece of text. It thus defines a distribution over all the possible word sequences. The simplest language model is the unigram language model, which is essentially a word distribution. More complex language models might usemore context information (e.g., word history) in predicting the next word if the speaker were to utter the words in a document, what is the likelihood they would then say the words in the query.

5. **Inference Network Model** In this model, document retrieval is modeled as an inference process in an inferencenetwork Most techniques used by IR systems can be implemented under this model. In the simplest implementation of this model, a document instantiates a term with certain strength, and the credit from multiple
6. terms is accumulated given a query to compute the equivalent of a numeric score for the document. From an operational perspective, the strength of instantiation of a term for a document can be considered as the weightof the term in the document, and document ranking in the simplest form of this model becomes similar to ranking in the vector space model and the probabilistic models described above. The strength of instantiation ofa term for a document is not defined by the model, and any formulation can be used.

Comparison of classical models

IR Models (IR mod)/ attributes (A)	**Boolean Model**	**Vector space Model**	**Probabilistic Model**	**Latent semantic Indexing**
Concept (A)	Based on set theory andBoolean algebra	Based on the conceptof vectors	Based on probability ranking principle	It is an extension of vector space model
Representation(A)	Documents are represented by the index terms extracted from documents, and queries are Boolean expressions on terms.	Represented in the form of weighted-term vectors. Cosine measure is used to find the similarities	Documents andqueries are represented in binary vectors	Documents are represented in the form of term-document matrix.

Information Type (A)	Docs not consider semantic information	It consider semantic information	Considers the semantic informatio n	Considers the semantic informatio n
Word occurrence (A)	Number of occurrence arc not	Tells about the number of occurrence	Occurrence based on the probability	Based on term-document matrix

	mentioned		relevance	
Output(A)	Exact match of the output to the query	Best match of the query	It gives best match of output	Best match of the query
Advantages(A)	Easy to implement	Simple model, weights are not in binary	Theoretical adequacy: ranks by probabilities	Synonymy and polysemy
Disadvantages(A)	Does not rank documents, retrieves too many ortoo few	Suffers from synonymy and polysemy. It theoretically assumesthat terms are statistically independent	Binary weights ignore frequenciesand independence assumption.	Not clear about similarity betweenwords

Table 1: Comparison of Information Model

Introduction to Alternative Algebraic Models:

Latent semantic analysis (LSA) is a technique in natural language processing, in particular distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis). A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Paragraphs are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two columns. Values close to 1 represent very similar paragraphs while values close to 0 represent very dissimilar paragraphs.

LSI is a proximity model, a mathematical technique for spatially grouping similar objects together. This is achieved by first a) identifying keyword co-occurrences that exist across a set of documents, then b) organizing them into a multi-dimensional term-by document matrix, and finally c) algebraically decomposing the matrix to reduce the matrix dimensionality. As the dimensional space is reduced, related documents draw closer to one another. The relative distances between these points in the reduced vector space have been shown to represent semantic similarity between documents, and can be used as a basis for the final stage of d) fulfilling information queries. The keywords input by a user are mapped onto the same reduced vector space, and the information retrieval systems present documents to the user that are located in the same relative neighbourhood. These retrieved resources are semantically related to the user's search criteria, even if they do not share the exact same keywords. Testing has demonstrated that these semantically related documents are also related conceptually, thereby satisfying the user's information need. This is the fundamental appeal and promise of LSI; that it circumvents the need for direct word to-word mapping between user and system, and replaces onerous, manual indexing techniques with an automated approach.

The accuracy of LSI model performance in experimental studies – up to 30% better retrieval performance overtraditional lexical matching techniques (Dumais, 1991) - is impressive. At a time when electronic textual resources

4

are burgeoning beyond the capacity of manual indexing efforts, LSI's promise of accurate, automated content representation holds great appeal. Despite 15 years of active research into ways to improve the latent semantic indexing model, though, field applications of LSI remains limited to specialized resource collections of limited size.

Benefits of LSI

1. LSI helps overcome synonymy by increasing recall, one of the most problematic constraints of Boolean keyword queries and vector space models. Synonymy is often the cause of mismatches in the vocabulary used by the authors of documents and the users of information retrieval systems. As a result, Boolean or keyword queries often return irrelevant results and miss information that is relevant.
2. LSI is also used to perform automated document categorization. In fact, several experiments have demonstrated that there are a number of correlations between the way LSI and humans process and categorize text. Document categorization is the assignment of documents to one or more predefined categories based on their similarity to the conceptual content of the categories. LSI uses example documents to establish the conceptual basis for each category. During categorization processing, the concepts contained in the documents being categorized are compared to the concepts contained in the example items, and a category (or categories) is assigned to the documents based on the similarities between the concepts they contain and the concepts that are contained in the example documents.
3. Dynamic clustering based on the conceptual content of documents can also be accomplished using LSI. Clustering is a way to group documents based on their conceptual similarity to each other without using example documents to establish the conceptual basis for each cluster. This is very useful when dealing with an unknown collection of unstructured text.
4. Because it uses a strictly mathematical approach, LSI is inherently independent of language. This enables LSI to elicit the semantic content of information written in any language without requiring the use of auxiliary structures, such as dictionaries and thesauri. LSI can also perform cross-linguistic concept searching and example-based categorization. For example, queries can be made in one language, such as English, and conceptually similar results will be returned even if they are composed of an entirely different language or of multiple languages.
5. LSI is not restricted to working only with words. It can also process arbitrary character strings. Any object that can be expressed as text can be represented in an LSI vector space. For example, tests with MEDLINE abstracts have shown that LSI is able to effectively classify genes based on conceptual modeling of the biological information contained in the titles and abstracts of the MEDLINE citations.
6. LSI automatically adapts to new and changing terminology, and has been shown to be very tolerant of noise (i.e., misspelled words, typographical errors, unreadable characters, etc.). This is especially important for applications using text derived from Optical Character Recognition (OCR) and speech-to-text conversion. LSI also deals effectively with sparse, ambiguous, and contradictory data.
7. Text does not need to be in sentence form for LSI to be effective. It can work with lists, free-form notes, email, Web-based content, etc. As long as a collection of text contains multiple terms, LSI can be used to identify patterns in the relationships between the important terms and concepts contained in the text.

LSI has proven to be a useful solution to a number of conceptual matching problems. The technique has beenshown to capture key relationship information, including causal, goal-oriented, and taxonomic information

Applications:

1. Information discovery
2. Automated document classification (eDiscovery, Government/Intelligence community, Publishing)
3. Text summarization (eDiscovery, Publishing)
4. Relationship discovery (Government, Intelligence community, Social Networking)
5. Automatic generation of link charts of individuals and organizations (Government, Intelligence community)
6. Matching technical papers and grants with reviewers
7. Online customer support
8. Understanding software source codes(Software Engineering)
9. Filtering spam (System Administration)
10. Information visualization
11. Stock returns prediction

UNIT – 3

3.1 Query Language

Different kinds of queries normally posed to text retrieval systems. This is in part dependent on the retrieval model the system adopts, i.e., a full-text system will not answer the same kinds of queries as those answered by a system based on keyword ranking (as Web search engines) or on a hypertext model. In this topic we show which queries can be formulated. The type of query the user might formulate is largely dependent on the underlying information retrieval model. For query languages not aimed at information retrieval, the concept of ranking cannot be easily defined, so we consider them as languages for data retrieval. Furthermore, some query languages are not intended for final users and can be viewed as languages that a higher level software package should use to query an on-line database or a CD-ROM archive. In that case, we talk about protocols rather than query languages. Depending on the user experience, a different query language will be used. For example, if the user knows exactly what he wants, the retrieval task is easier and ranking may not even be needed.

An important issue is that most query languages try to use the content (i.e., the semantics) and the structure of the text (i.e., the text syntax) to find relevant documents. In that sense, the system may fail to find the relevant answers. For this reason, a number of techniques meant to enhance the usefulness of the queries exist Examples include the expansion of a word to the set of its synonyms or the use of a thesaurus and stemming to put together all the derivatives of the same word. Moreover, some words which are very frequent and do not carry meaning (such as 'the') called stopwords, may be removed. When we want to emphasize the difference between words that can be retrieved by a query and those which cannot, we call the former 'keywords.'

Orthogonal to the kind of queries that can be asked is the subject of the retrieval unit the information system adopts. The retrieval unit is the basic element which can be retrieved as an answer to a query (normally a set of such basic elements is retrieved, sometimes ranked by relevance or other criterion). The retrieval unit can be a file, a document, a Web page, a paragraph, or some other structural unit which contains an answer to the search query. From this point on, we will simply call those retrieval units 'documents,' although as explained this can have different meaning.

3.1.1 Keyword based querying

A query is the formulation of a user information need. In its simplest form, a query is composed of keywords and the documents containing such keywords are searched for. Keyword-based queries are popular because they are intuitive, easy to express, and allow for fast ranking. Thus, a query can be (and in many cases is) simply a word, although it can in general be a more complex combination of operations involving several words.

3.1.1.1 Single-word queries

The most elementary query that can be formulated in a text retrieval system is a word. Text documents are assumed to be essentially long sequences of words. Although some models present a more general view, virtually all models allow us to see the text in this perspective and to search words. Some models are also able to see the internal division of words into letters. These latter models permit the searching of other types of patterns. The set of words retrieved by these extended queries can then be fed into the word-treating machinery, say to perform thesaurus expansion or for ranking purposes.

A word is normally defined in a rather simple way. The alphabet is split into 'letters' and 'separators,' and a word is a sequence of letters surrounded by separators. More complex models allow us to specify that some characters are not letters but do not split a word, e.g. the hyphen in 'on-line.' It is good practice to leave the choice of what is a letter and what is a separator to the manager of the text database.

The division of the text into words is not arbitrary, since words carry a lot of meaning in natural language. Because of that, many models (such as the vector model) are completely structured on the concept of words, and words are the only type of queries allowed (moreover, some systems only allow a small set of words to be extracted from the documents). The result of word queries is the set of documents containing at least one of the words of the query. Further, the resulting documents are ranked according to a degree of similarity to the query. To support ranking, two common statistics on word occurrences inside texts are commonly used: 'term frequency' which counts the number of times a word appears inside a document and 'inverse document frequency' which counts the number of documents in which a word appears.

Additionally, the exact positions where a word appears in the text may be required for instance, by an interface which highlights each occurrence of that word.

3.1.1.2 Context queries

Many systems complement single-word queries with the ability to search words in a given context, that is, near other words. Words which appear near each other may signal a higher likelihood of relevance than if they appear apart. For instance, we may want to form phrases of words or find words which are proximal in the text. Therefore, we distinguish two types of queries:

- **Phrase** is a sequence of single-word queries. An occurrence of the phrase is a sequence of words. For instance, it is possible to search for the word `enhance,' and then for the word 'retrieval.' In phrase queries it is normally understood that the separators in the text need not be the same as those in the query (e.g., two spaces versus one space), and uninteresting words are not considered at all. For instance, the previous example could match a text such as `...enhance the retrieval...'. Although the notion of a phrase is a very useful feature in most cases, not all systems implement it.
- **Proximity** A more relaxed version of the phrase query is the proximity query. In this case, a sequence of single words or phrases is given, together with a maximum allowed distance between them. For instance, the above example could state that the two words should occur within four words, and therefore a match could be ' . . . enhance the power of retrieval ' This distance can be measured in characters or words depending on the system. The words and phrases may or may not be required to appear in the same order as in the query.
  Phrases can be ranked in a fashion somewhat analogous to single words. Proximity queries can be ranked in the same way if the parameters used by the ranking technique do not depend on physical proximity. Although it is not clear how to do better ranking, physical proximity has semantic value. This is because in most cases the proximity means that the words are in the same paragraph, and hence related in some way.

3.1.1.3 Boolean queries

The oldest (and still heavily used) form of combining keyword queries is to use Boolean operators. A Boolean query has a syntax composed of atoms (i.e., basic queries) that retrieve documents, and of Boolean operators which work on their operands (which are sets of documents) and deliver sets of documents. Since this scheme is in general compositional (i.e., operators can be composed over the results of other operators), a query syntax tree is naturally defined, where the leaves correspond to the basic queries and the internal nodes to the operators. The query syntax tree operates on an algebra over sets of documents (and the final answer of the query is also a set of documents). This is much as, for instance, the syntax trees of arithmetic expressions where the numbers and variables are the leaves and the operations form the internal nodes. Below **Figure 1** shows an example.

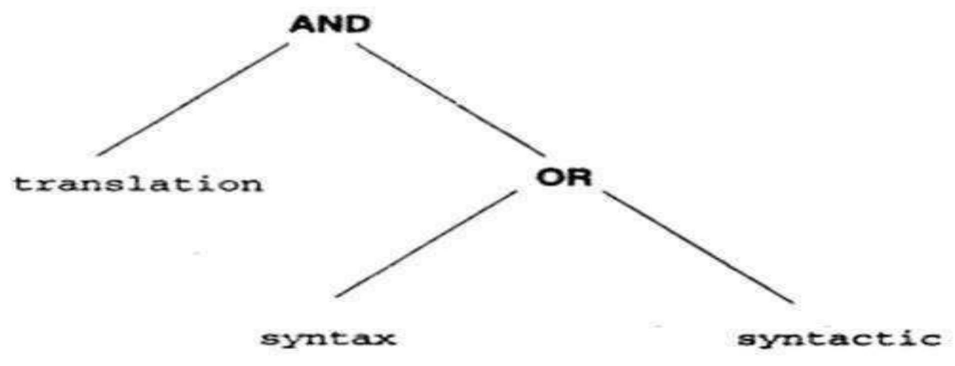


Figure 1: An example of a query syntax tree. It will receive all the documents which contain the word translation as well as either the word syntax or the word syntactic

The operators most commonly used, given two basic queries or Boolean sub expressions e_1 and e_2, are:

- **OR** The query (e_1 OR e_2) selects all documents which satisfy e_1 or e_2. Duplicates are eliminated.
- **AND** The query (e_1 AND e_2) selects all documents which satisfy both e_1 and e_2.
- **BUT** The query (e_1 BUT e_2) selects all documents which satisfy e_1 but not e_2. Notice that classical Boolean logic uses a NOT operation, where (NOT e_2) is valid whenever e_2 is not. In this case all documents not satisfying e_2 should be delivered, which may retrieve a huge amount of text and is probably not what the user wants. The BUT operator, instead, restricts the universe of retrievable elements to the result of e_1.

Besides selecting the appropriate documents, the IR system may also sort the documents by some criterion, highlight the occurrences within the documents of the words mentioned in the query, and allow feedback by taking the answer set as a basis to reformulate the query.
With classic Boolean systems, no ranking of the retrieved documents is normally provided. A document either satisfies the Boolean query (in which case it is retrieved) or it does not (in which case it is not retrieved). This is quite a limitation because it does not allow for partial matching between a document and a user query. To overcome this limitation, the condition for retrieval must be relaxed. For instance, a document which partially satisfies an AND condition might be retrieved.
In fact, it is widely accepted that users not trained in mathematics find the meaning of Boolean operators difficult to grasp With this problem in mind, a 'fuzzy Boolean' set of operators has been proposed. The idea is that the meaning of AND and OR can be relaxed, such that instead of forcing an element to appear in all the operands (AND) or at least in one of the operands (OR), they retrieve elements appearing in some operands (the AND may require it to appear in more operands than the OR). Moreover, the documents are ranked higher when they have a larger number of elements in common with the query.

3.1.1.4 Natural language

Pushing the fuzzy Boolean model even further, the distinction between AND and OR can be completely blurred, so that a query becomes simply an enumeration of words and context queries. All the documents matching a portion of the user query are retrieved. Higher ranking is assigned to those documents matching more parts of the query. The negation can be handled by letting the user express that some words are not desired, so that the documents containing them are penalized in the ranking computation. A threshold may be selected so that the documents with very low weights are not retrieved. Under this scheme we have completely eliminated any reference to Boolean operations and entered into the field of natural language queries. In fact, one can consider that Boolean queries are a simplified abstraction of natural language queries.
A number of new issues arise once this model is used, especially those related to the proper way to rank an element with respect to a query. The search criterion can be re-expressed using a different model, where documents and queries are considered just as a vector of 'term weights' (with one coordinate per interesting keyword or even per existing text word) and queries are considered in exactly the same way (context queries are not considered in this case). Therefore, the query is now internally converted into a vector of term weights and the aim is to retrieve all the vectors (documents) which are close to the query (where closeness has to be defined in the model). This allows many interesting possibilities, for instance a complete document can be used as a query (since it is also a vector), which naturally leads to the use of relevance feedback techniques (i.e., the user can select a document from the result and submit it as a new query to retrieve documents similar to the selected one). The algorithms for this model are totally different from those based on searching patterns (it is even possible that not every text word needs to be searched but only a small set of hopefully representative keywords extracted from each document).

3.2 Query Operations

Without detailed knowledge of the collection make-up and of the retrieval environment, most users find it difficult to formulate queries which are well designed for retrieval purposes. In fact, as observed with Web search engines, the users might need to spend large amounts of time reformulating their queries to accomplish effective retrieval. This difficulty suggests that the first query formulation should be treated as an initial (naive) attempt to retrieve relevant information. Following that, the documents initially retrieved could be examined for relevance and new improved query formulations could then be constructed in the hope of retrieving additional useful documents. Such query reformulation involves two basic steps:
1. Expanding the original query with new terms,
2. And reweighting the terms in the expanded query.

Several of approaches available for improving the initial query formulation through query expansion and term reweighting. These approaches are grouped in three categories:

1. Approaches based on feedback information from the user;
2. Approaches based on information derived from the set of documents initially retrieved (called the local set of documents);
3. Approaches based on global information derived from the document collection.

3.2.1 User relevance feedback

Relevance feedback is the most popular query reformulation strategy. In a relevance feedback cycle, the user is presented with a list of the retrieved documents and, after examining them, marks those which are relevant In practice, only the top 10 (or 20) ranked documents need to be examined. The main idea consists of selecting important terms, or expressions, attached to the documents that have been identified as relevant by the user, and of enhancing the importance of these terms in a new query formulation. The expected effect is that the new query will be moved towards the relevant documents and away from the non-relevant ones.

Early experiments using the Smart system and later experiments using the probabilistic weighting model have shown good improvements in precision for small test collections when relevance feedback is used. Such improvements come from the use of two basic techniques:

a. Query expansion (addition of new terms from relevant documents)
b. Term reweighting (modification of term weights based on the user relevance judgment).

Relevance feedback presents the following main advantages over other query reformulation strategies: (a) It shields the user from the details of the query reformulation process because all the user has to provide is a relevance judgment on documents (b) It breaks down the whole searching task into a sequence of small steps which are easier to grasp; and (c) It provides a controlled process designed to emphasize some terms (relevant ones) and de-emphasize others (non-relevant ones).

The usage of user relevance feedback to (a) expand queries with the vector model, (b) reweight query terms with the probabilistic model, and (c) reweight query terms with a variant of the probabilistic model.

3.2.1.1 Query Expansion and Term Reweighting for the Vector Model

The application of relevance feedback to the vector model considers that the term-weight vectors of the documents identified as relevant (to a given query) have similarities among themselves (i.e., relevant documents resemble each other). Further, it is assumed that non-relevant documents have term-weight vectors which are dissimilar from the ones for the relevant documents. The basic idea is to reformulate the query such that it gets closer to the term-weight vector space of the relevant documents.

Let us define some additional terminology regarding the processing of a given query q as follows,

D_r: set of relevant documents, as identified by the user, among the retrieved documents;

D_n: set of non-relevant documents among the retrieved documents;

C_r: set of relevant documents among all documents in the collection;

$|D_r|$, $|D_n|$, $|C_r|$: number of documents in the sets D_r, D_n, $C_r$respectively;

α, β, γ : tuning constants.

Consider first the unrealistic situation in which the complete set C_r of relevant documents to a given query q is known in advance. In such a situation, it can be demonstrated that the best query vector for distinguishing the relevant documents from the non-relevant documents is given by ([1]),

$$\boxed{\vec{q}_{opt} = \frac{1}{|C_r|} \sum_{\forall \vec{d}_j \in C_r} \vec{d}_j - \frac{1}{N - |C_r|} \sum_{\forall \vec{d}_j \notin C_r} \vec{d}_j \qquad [1]}$$

problem with this formulation is that the relevant documents which compose the set C_r are not known a priori. In fact, we are looking for them. The natural way to avoid this problem is to formulate an initial query and to incrementally change the initial query vector. This incremental change is accomplished by restricting the computation to the documents known to be relevant (according to the user judgment) at that point. There are three classic and similar ways to calculate the modified query q_m as follows,

$$Standard\_Rochio: \quad \vec{q}_m = \alpha\,\vec{q} + \frac{\beta}{|D_r|}\sum_{\forall \vec{d}_j \in D_r}\vec{d}_j - \frac{\gamma}{|D_n|}\sum_{\forall \vec{d}_j \in D_n}\vec{d}_j$$
$$Ide\_Regular: \quad \vec{q}_m = \alpha\,\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r}\vec{d}_j - \gamma\sum_{\forall \vec{d}_j \in D_n}\vec{d}_j$$
$$Ide\_Dec\_Hi: \quad \vec{q}_m = \alpha\,\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r}\vec{d}_j - \gamma\; max_{non\text{-}relevant}(\vec{d}_j)$$

Where $max_{non\text{-}relevant}(\vec{d_j})$ is a reference to the highest ranked non-relevant document. Notice that now D_r and D_n stand for the sets of relevant and non-relevant documents (among the retrieved ones) according to the user judgment, respectively. In the original formulations, ʻochio fixed α = 1 and Ide fixed α = β = γ = 1. The expressions above are modern variants. The current understanding is that the three techniques yield similar results.
The Rochio formulation is basically a direct adaptation of equation 1 in which the terms of the original query are added in. The motivation is that in practice the original query q may contain important information. Usually, the information contained in the relevant documents is more important than the information provided by the non-relevant documents. This suggests making the constant γ smaller than the constant β. An alternative approach is to set γ to 0 which yields a positive feedback strategy.
The main advantages of the above relevance feedback techniques are simplicity and good results. The simplicity is due to the fact that the modified term weights are computed directly from the set of retrieved documents. The good results are observed experimentally and are due to the fact that the modified query vector does reflect a portion of the intended query semantics. The main disadvantage is that no optimality criterion is adopted.

3.2.1.2 Query Expansion and Term Reweighting for the Probabilistic Model

3.2.2 The probabilistic model dynamically ranks documents similar to a query q according to the probabilistic ranking principle. In probabilistic model similarity of a document dj to a. query q can be expressed as ([2])

$$sim(d_j,q)\ \alpha \sum_{i=1}^{t} w_{i,q}\, w_{i,j}\left(\log\frac{P(k_i|R)}{1-P(k_i|R)} + \log\frac{1-P(k_i|\overline{R})}{P(k_i|\overline{R})}\right) \qquad [2]$$

where $P(k_i|R)$ stands for the probability of observing the term lei in the set R of relevant documents and $P(k_i|\overline{R})$ stands for the probability of observing the term ki in the set R of non-relevant documents. Initially, equation 2 cannot be used because the probabilities $P(k_i|R)$ and $P(k_i|\overline{R}))$ are unknown. A number of different methods for estimating these probabilities automatically (i.e., without feedback from the user. With user feedback information, these probabilities are estimated in a slightly different way as follows. For the initial search (when there are no retrieved documents yet), assumptions often made include:
(a $P(k_i|R)$is constant for all terms ki (typically 0.5) and
(b) the term probability distribution $P(k_i|\overline{R})$can be approximated by the distribution in the whole collection. These two assumptions yield:

$$P(k_i|R) = 0.5$$
$$P(k_i|\overline{R}) = \frac{n_i}{N}$$

where, as before, n_i, stands for the number of documents in the collection which contain the term k_i..Substituting into equation 2, we obtain

$$sim_{initial}(d_j,q) = \sum_{i}^{t} w_{i,q}\, w_{i,j}\, \log \frac{N - n_i}{n_i}$$

For the feedback searches, the accumulated statistics related to the relevance or non-relevance of previously retrieved documents are used to evaluate the probabilities $P(k_i|R)$ and $P(k_i|\overline{R})$. As before, let D_r be the set of relevant retrieved documents (according to the user judgement) and $D_{r,i}$ be the sub-set of D_r composed of the documents which contain the term k_i. Then, the probabilities $P(k_i|R)$ and $P(k_i|\overline{R})$ can be approximated by ([4])

[3]

Using these approximations equation 2 can be written a

$$P(k_i|R) = \frac{|D_{r,i}| + 0.5}{|D_r| + 1}; \quad P(k_i|\overline{R}) = \frac{n_i - |D_{r,i}| + 0.5}{N - |D_r| + 1}$$

Notice that here; contrary to the procedure in the vector space model, no query expansion occurs. The same query terms are being reweighted using feedback information provided by the user.

Formula 3 poses problems for certain small values of $|D_r|$ and $|D_{r,i}|$ that frequently arise in practice ($|D_r| = 1$, $|D_{r,i}| =$

$$sim(d_j,q) = \sum_{i=1}^{t} w_{i,q}\, w_{i,j}\, \log \left[\frac{|D_{r,i}|}{|D_r| - |D_{r,i}|} \div \frac{n_i - |D_{r,i}|}{N - |D_r| - (n_i - |D_{r,i}|)} \right]$$

0). For this reason, a 0.5 adjustment factor is often added to the estimation of $P(k_i|R)$and $P(k_i|R)$yielding ([4])

[4]

This 0.5 adjustment factor may provide unsatisfactory estimates in some cases, and alternative adjustments have been proposed such as n_i/N or $(n_i — |D_{r,i}|) /(N - |D_{r,i}|)$. Taking n_i/N as the adjustment factor (instead of 0.5), equa-tion4 becomes

$$P(k_i|R) = \frac{|D_{r,i}| + \frac{n_i}{N}}{|D_r| + 1}; \quad P(k_i|\overline{R}) = \frac{n_i - |D_{r,i}| + \frac{n_i}{N}}{N - |D_r| + 1}$$

The main advantages of this relevance feedback procedure are that the feedback process is directly related to the derivation of new weights for query terms and that the term reweighting is optimal under the assumptions of term independence and binary document indexing ($w_{i.q} \varepsilon \{0,1\}$ and $w_{i,j} \varepsilon \{0, 1\}$).

The disadvantages include:

(1) Document term weights are not considered during the feedback loop.

(2) Weights of terms in the previous query formulations are also disregarded.

(3) No query expansion is used (the same set of index terms in the original query is reweighted over and over again). As a result of these disadvantages, the probabilistic relevance feedback methods do not in general operate as effectively as the conventional vector modification methods.

To extend the probabilistic model with query expansion capabilities, different approaches have been proposed in the literature ranging from term weighting for query expansion to term clustering techniques based on spanning trees. All of these approaches treat probabilistic query expansion separately from probabilistic term rewriting.

3.3 Text Operations

Not all words are equally significant for representing the semantics of a document. In written language, some words carry more meaning than others. Usually, noun words (or groups of noun words) are the ones which are most representative of document content. Therefore, it is usually considered worthwhile to preprocess the text of the documents in the collection to determine the terms to be used as index terms. During this preprocessing phase other useful text operations can be performed such as elimination of stop-words, stemming (reduction of a word to its grammatical root), the building of a thesaurus, and compression.

We already know that representing documents by sets of index terms leads to a rather imprecise representation of the semantics of the documents in the collection. For instance, a term like 'the' has no meaning whatsoever by itself and might lead to the retrieval of various documents which are unrelated to the present user query. We say that using the set of all words in a collection to index its documents generates too much noise for the retrieval task. One way to reduce this noise is to reduce the set of words which can be used to refer to (i.e., to index) documents. Thus, the preprocessing of the documents in the collection might be viewed simply as a process of controlling the size of the vocabulary (i.e., the number of distinct words used as an index terms). It is expected that the use of a controlled vocabulary leads to an improvement in retrieval performance.

While controlling the size of the vocabulary is a common technique with commercial systems, it does introduce an additional step in the indexing process which is frequently not easily perceived by the users. As a result, a common user might be surprised with some of the documents retrieved and with the absence of other documents which he expected to see. For instance, he might remember that a certain document contains the string 'the house of the lord' and notice that such a document is not present among the top 20 documents retrieved in response to his query request (because the controlled vocabulary contains neither 'the' nor 'of '). Thus, it should be clear that, despite a potential improvement in retrieval performance, text transformations done at preprocessing time might make it more difficult for the user to interpret the retrieval task. In recognition of this problem, some search engines in the Web are giving up text operations entirely and simply indexing all the words in the text. The idea is that, despite a more noisy index, the retrieval task is simpler (it can be interpreted as a full text search) and more intuitive to a common user.

3.3.1 Document Preprocessing

Document preprocessing is a procedure which can be divided mainly into five text operations (or transformations) (as shown in figure 2):

1. Lexical analysis of the text with the objective of treating digits, hyphens, punctuation marks, and the case of letters.
2. Elimination of stopwords with the objective of filtering out words with very low discrimination values for retrieval purposes.
3. Stemming of the remaining words with the objective of removing affixes (i.e., prefixes and suffixes) and allowing the retrieval of documents con-training syntactic variations of query terms (e.g., connect, connecting, connected, etc).
4. Selection of index terms to determine which words/stems (or groups of words) will be used as an indexing element. Usually, the decision on whether a particular word will be used as index term is related to the syntactic nature of the word. In fact, noun words frequently carry more semantics than adjectives, adverbs, and verbs.
5. Construction of term categorization structures such as a thesaurus, or ex-traction of structure directly represented in the text, for allowing the expansion of the original query with related terms (a usually useful procedure).

3.3.1.1 Lexical Analysis of the Text

Lexical analysis is the process of converting a stream of characters (the text of the documents) into a stream of words (the candidate words to be adopted as index terms). Thus, one of the major objectives of the lexical analysis phase is the identification of the words in the text. At first glance, all that seems to be involved is the recognition of spaces as word separators (in which case, multiple spaces are reduced to one space). However, there is more to it than this. For instance, the following four particular cases have to be considered with care: digits, hyphens, punctuation marks, and the case of the letters (lower and upper case).

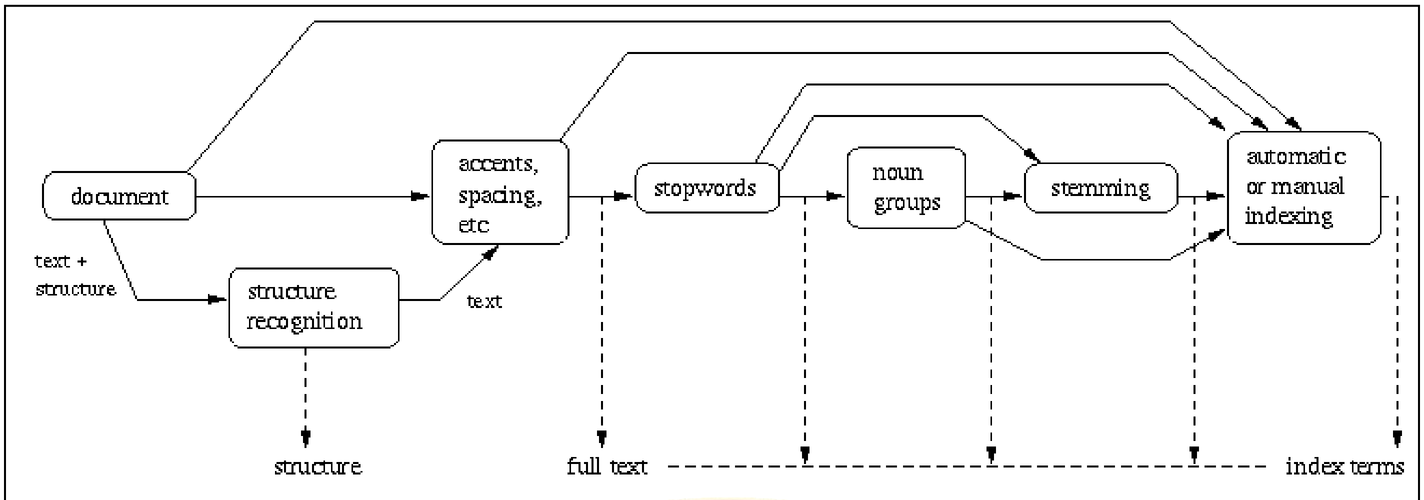


Figure 2: Logical view of a document: from full text to a set of index terms.

3.3.1.2 Elimination of Stopwords

Words which are too frequent among the documents in the collection are not good discriminators. In fact, a word which occurs in 80% of the documents in the collection is useless for purposes of retrieval. Such words are frequently referred to as stopwords and are normally filtered out as potential index terms. Articles, prepositions, and conjunctions are natural candidates for a list of stopwords.

Elimination of stopwords has an additional important benefit. It reduces the size of the indexing structure considerably. In fact, it is typical to obtain a compression in the size of the indexing structure of 40% or more solely with the elimination of stopwords.

3.3.1.3 Stemming

Frequently, the user specifies a word in a query but only a variant of this word is present in a relevant document. Plurals, gerund forms, and past tense suffixes are examples of syntactical variations which prevent a perfect match between a query word and a respective document word. This problem can be partially overcome with the substitution of the words by their respective stems.

A stem is the portion of a word which is left after the removal of its affixes (i.e., prefixes and suffixes). A typical example of a stem is the word connects which is the stem for the variants connected, connecting, connection, and connections. Stems are thought to be useful for improving retrieval performance because they reduce variants of a same root word to a common concept. Fur-thermore, stemming has the secondary effect of reducing the size of the indexing structure because the number of distinct index terms is reduced.

3.3.1.4 Index Terms Selection

If a full text representation of the text is adopted then all words in the text are used as index terms. The alternative is to adopt a more abstract view in which not all words are used as index terms. This implies that the set of terms used as indices must be selected. In the area of bibliographic sciences, such a selection of index terms is usually done by a specialist. An alternative approach is to select terms for index terms automatically.

Distinct automatic approaches for selecting index terms can be used. A good approach is the identification of noun groups

A sentence in natural language text is usually composed of nouns, pro-nouns, articles, verbs, adjectives, adverbs, and connectives. While the words in each grammatical class are used with a particular purpose, it can be argued that most of the semantics is carried by the noun words. Thus, an intuitively promis-ing strategy for selecting index terms automatically is to use the nouns in the text. This can be done through the systematic elimination of verbs, adjectives, adverbs, connectives, articles, and pronouns.

3.3.1.5 Thesauri

The word **thesaurus** has Greek and Latin origins and is used as a reference to a treasury of words. In its simplest form, this treasury consists of

1. A precompiled list of important words in a given domain of knowledge and
2. For each word in this list, a set of related words. Related words are, in its most common variation, derived from a synonymity relationship.

In general, however, a thesaurus also involves some normalization of the vocabulary and includes a structure much more complex than a simple list of words and their synonyms. For instance, the popular thesaurus published by Peter Roget also includes phrases which mean that concepts more complex than single words are considered. 'oget's thesaurus is of a general nature (i.e., not specific to a certain domain of knowledge) and organizes words and phrases in categories and subcategories.

3.3.2 Document Clustering

Document clustering is the operation of grouping together similar (or related) documents in classes. In this regard, document clustering is not really an operation on the text but an operation on the collection of documents.

The operation of clustering documents is usually of two types: global and local. In a global clustering strategy, the documents are grouped accordingly to their occurrence in the whole collection. In a local clustering strategy, the grouping of documents is affected by the context defined by the current query and its **local** set of retrieved documents.

Figure 3: Term Document Matrix

Clustering methods are usually used in IR to transform the original query in an attempt to better represent the user information need. From this perspective, clustering is an operation which is more related to the transformation of the user query than to the transformation of the text of the documents.

3.4 Indexing

Index is a data structure built from the text to speed up the searches.

In the context of an information retrieval system that uses an index, the efficiency of the system can be measured by:

Indexing time: Time needed to build the index

- **Indexing space:** Space used during the generation of the index
- **Index storage:** Space required to store the index
- **Query latency:** Time interval between the arrival of the query and the generation of the answer
- **Query throughput:** Average number of queries processed per second

When a text is updated, any index built on it must be updated as well. Current indexing technology is not well prepared to support very frequent changes to the text collection

- **Semi-static collections:** collections which are updated at reasonable regular intervals (say, daily)

Most real text collections, including the Web, are indeed semi-static, for example, although the Web changes very fast, the crawls of a search engine are relatively slow,for maintaining freshness, incremental indexing is used

3.4.1 Inverted Index

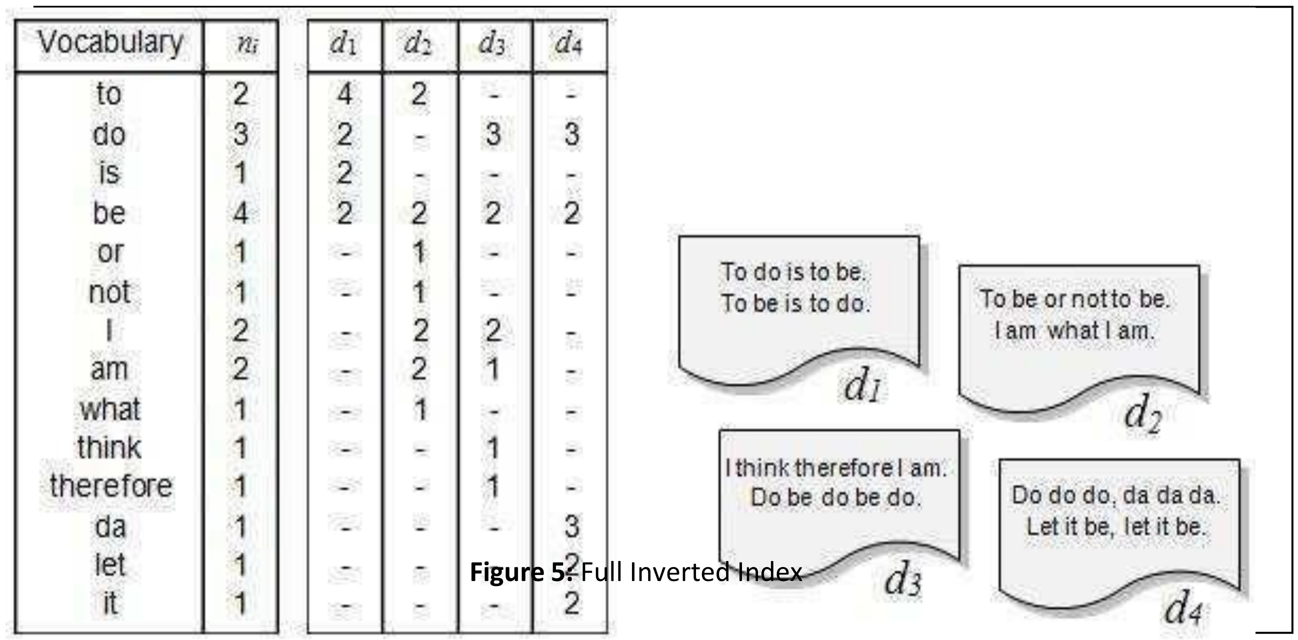
Inverted index is a word-oriented mechanism for indexing a text collection to speed up the searching task.

The inverted index structure is composed of two elements:

1. the vocabulary
2. the occurrences

The vocabulary is the set of all different words in the text, for each word in the vocabulary the index stores the documents which contain that word (inverted index).

Term-document matrix is the simplest way to represent the documents that contain each word of the vocabulary (as shown in figure 3).



Vocabulary	n_i	d_1	d_2	d_3	d_4
to	2	4	2	-	-
do	3	2	-	3	3
is	1	2	-	-	-
be	4	2	2	2	2
or	1	-	1	-	-
not	1	-	1	-	-
I	2	-	2	2	-
am	2	-	2	1	-
what	1	-	1	-	-
think	1	-	-	1	-
therefore	1	-	-	1	-
da	1	-	-	-	3
let	1	-	-	-	2
it	1	-	-	-	2

To do is to be. To be is to do. d_1

To be or not to be. I am what I am. d_2

I think therefore I am. Do be do be do. d_3

Do do do, da da da. Let it be, let it be. d_4

Figure 5: Full Inverted Index

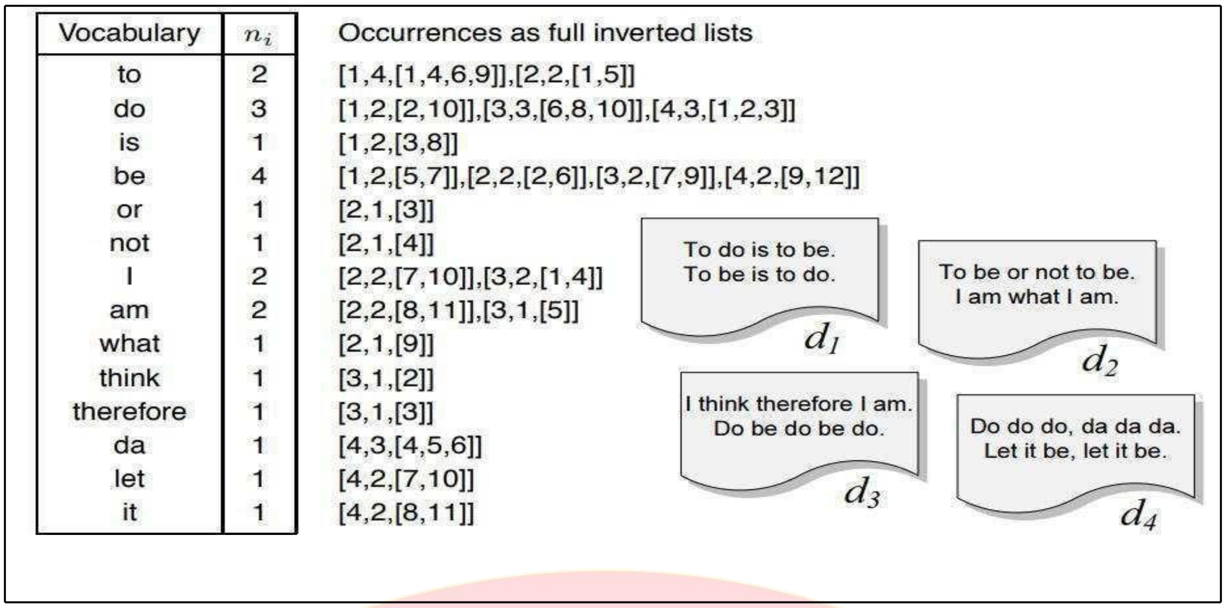
- The main problem of this simple solution is that it requires too much space (as shown in figure 3)
- As this is a sparse matrix, the solution is to associate a list of documents with each word
- The set of all those lists is called the occurrences

Figure 4: Basic Inverted Index

3.4.2 Full Inverted Indexes

The basic index is not suitable for answering phrase or proximity queries. Hence, we need to add the positions of each word in each document to the index (full inverted index)

In the case of multiple documents, we need to store one occurrence list per term-document pair (as shown in figure 5).



Vocabulary	n_i	Occurrences as full inverted lists
to	2	[1,4,[1,4,6,9]],[2,2,[1,5]]
do	3	[1,2,[2,10]],[3,3,[6,8,10]],[4,3,[1,2,3]]
is	1	[1,2,[3,8]]
be	4	[1,2,[5,7]],[2,2,[2,6]],[3,2,[7,9]],[4,2,[9,12]]
or	1	[2,1,[3]]
not	1	[2,1,[4]]
I	2	[2,2,[7,10]],[3,2,[1,4]]
am	2	[2,2,[8,11]],[3,1,[5]]
what	1	[2,1,[9]]
think	1	[3,1,[2]]
therefore	1	[3,1,[3]]
da	1	[4,3,[4,5,6]]
let	1	[4,2,[7,10]]
it	1	[4,2,[8,11]]

To do is to be.
To be is to do.
d_1

To be or not to be.
I am what I am.
d_2

I think therefore I am.
Do be do be do.
d_3

Do do do, da da da.
Let it be, let it be.
d_4

- The space required for the vocabulary is rather small Heaps' law: the vocabulary grows as $O(n^{\beta})$, where
  - n is the collection size.
  - β is a collection-dependent constant between 0.4 and 0.6.
- For instance, in the TREC-3 collection, the vocabulary of 1 gigabyte of text occupies only 5 megabytes.
- This may be further reduced by stemming and other normalization techniques.
- The occurrences demand much more space the extra space will be $O(n)$ and is around.
- 40% of the text size if stopwords are omitted 80% when stopwords are indexed.
- Document-addressing indexes are smaller, because only one occurrence per file must be recorded, for a given word.
- Depending on the document (file) size, document-addressing indexes typically require 20% to 40% of the text size.
- To reduce space requirements, a technique called block addressing is used
- The documents are divided into blocks, and the occurrences point to the blocks where the word appears

Searching

- Searching a single word is made by comparing its bit mask W with the bit masks B_i of all the text blocks
- Whenever ($W \ \& B_i = W$), where & is the bit-wise AND, the text block may contain the word
- Hence, an online traversal must be performed to verify if the word is actually there
- This traversal cannot be avoided as in inverted indexes (except if the risk of a false match is accepted)

3.5 Pattern Matching

A pattern is a set of syntactic features that must occur in a text segment. Those segments satisfying the pattern specifications are said to 'match' the pattern. We are interested in documents containing segments which match a given search pattern. Each system allows the specification of some types of patterns, which range from very simple (for example, words) to rather complex (such as regular expressions). In general, as more powerful is the set of patterns allowed, more involved are the queries that the user can formulate and more complex is the implementation of the search. The most used types of patterns are:

- **Words** A string (sequence of characters) which must be a word in the text. This is the most basic pattern.
- **Prefixes** A string which must form the beginning of a text word. For instance, given the prefix I comput' all the documents containing words such as 'computer,' `computation,' `computing,' etc. are retrieved.
- **Suffixes** A string which must form the termination of a text word. For instance, given the suffix `tars' all the documents containing words such as 'computers.' testers, painters,' etc. are retrieved.

- **Substrings** A string which can appear within a text word. For instance, given the substring `tal' all the documents containing words such as 'coastal,' talk,' 'metallic,' etc. are retrieved. This query can be restricted to find the substrings inside words, or it can go further and search the substring anywhere in the text (in this case the query is not restricted to be a sequence of letters but can contain word separators). For instance, a search for 'any flow' will match in the phrase ...many flowers '
- **Ranges** A pair of strings which matches any word lying between them in lexicographical order. Alphabets are normally sorted, and this induces an order into the strings which is called lexicographical order (this is indeed the order in which words in a dictionary are listed). For instance, the range between words 'held' and 'hold' will retrieve strings such as 'hoax' and `hissing.'
- **Allowing** errors, a word together with an error threshold. This search pattern retrieves all text words which are 'similar' to the given word. The concept of similarity can be defined in many ways. The general concept is that the pattern or the text may have errors (coming from typing, spelling, or from optical character recognition software, among others), and the query should try to retrieve the given word and what are likely to be its erroneous variants. Although there are many models for similarity among words, the most generally accepted in text retrieval is the Levenshtein distance, or simply edit distance. The edit distance between two strings is the minimum number of character insertions, deletions, and replacements needed to make them equal. Therefore, the query specifies the maximum number of allowed errors for a word to match the pattern (i.e., the maximum allowed edit distance). This model can also be extended to search substrings (not only words), retrieving any text segment which is at the allowed edit distance from the search pattern. Under this extended model, if a typing error splits 'flower' into `flower' it could still be found with one error, while in the restricted case of words it could not (since neither 'flo' nor `wer' are at edit distance 1 from 'flower'). Variations on this distance model are of use in computational biology for searching on DNA or protein sequences as well as in signal processing.

- **Regular expressions** some text retrieval systems allow searching for regular expressions. A regular expression is a rather general pattern builttext structure. The text collections tend to have some structure built into them, and allowing the user to query those texts based on their structure (and not only their content) is becoming attractive. The standardization of languages to represent structured texts such as HTML has pushed forward in this direction.

Unit IV

Introduction:

Web Search: Crawling and Indexes, Search Engine architectures, Link Analysis and ranking algorithms such as HITS andPageRank, Meta searches, Performance Evaluation of search engines using various measures, Introduction to search engine optimization.

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

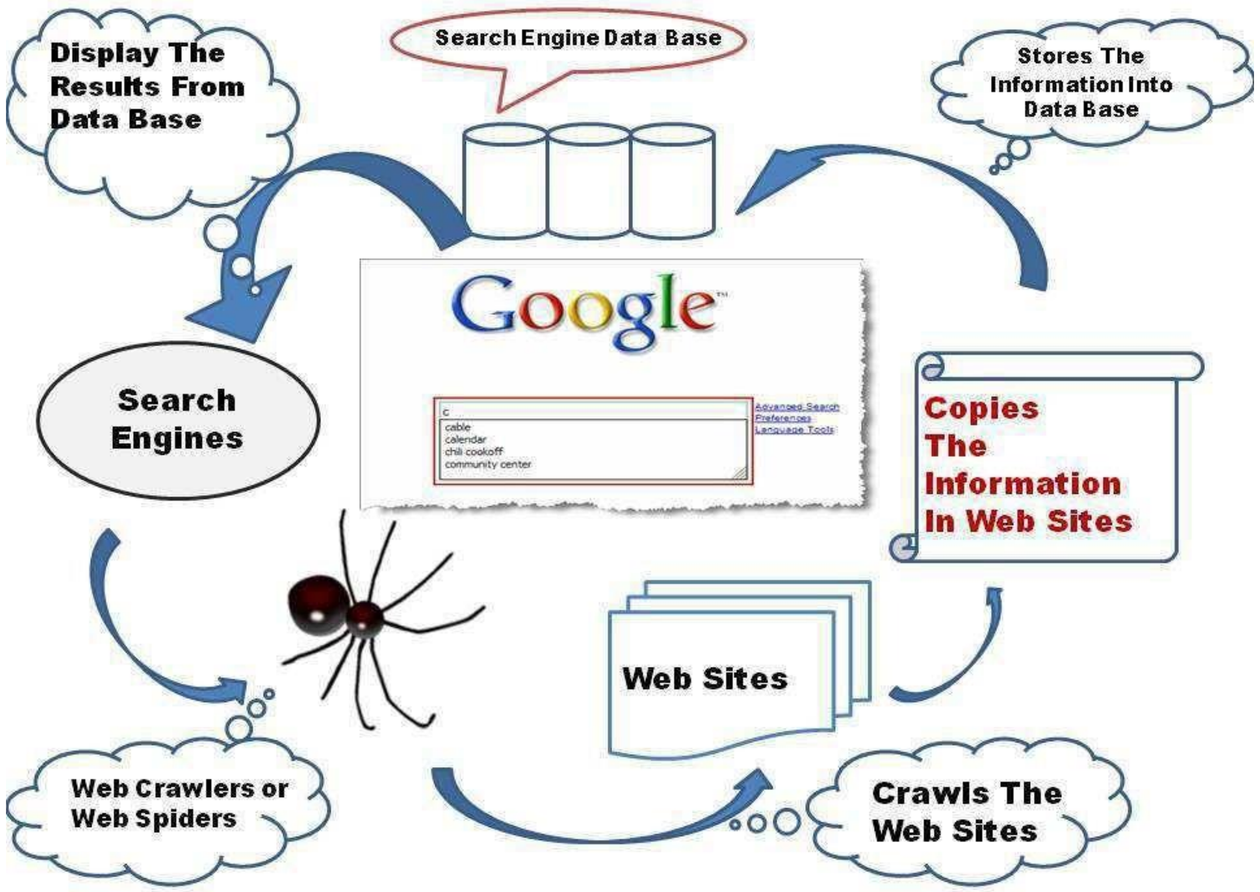

Display The Results From Data Base
Search Engine Data Base
Stores The Information Into Data Base
Google
Search Engines
Copies The Information In Web Sites
Web Sites
Web Crawlers or Web Spiders
Crawls The Web Sites

Fig.1 Web Search Engine Information Retrieval Process

Crawling and indexing the web

In layman's terms, indexing is the process of adding webpages into Google search. Depending upon which meta tag you used (index or NO-index), Google willcrawl and index your pages. A no-index tag means that that page will not be added into the web search'sindex. By default, every WordPress post and page isindexed.

Indexing a web page

Web indexing (or Internet indexing) refers to various methods for indexing the contents of a website or of the Internet as a whole. With the increase in the number of periodicals that have articles online, web indexing is also becoming important for periodical websites.

Crawling in SEO

Crawling is the process performed by search engine crawler, when searching for relevant websites on the index. For instance,Google is constantly sending out "spiders" or "bots" which is a search engine's automatic navigator to discover which websites contain the most relevant information related to certain keywords.

Search Engine architectures

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.

Architecture

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

Content collection and refinement.

Search core

User and application interfaces

Technology used by search engine to crawl websites:

A Web crawler, sometimes called a spider, is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing (webspidering). Web search engines and some other sites use Web crawling orspidering software to update their web content or indices of others sites' web content.

Search Engine Architecture:

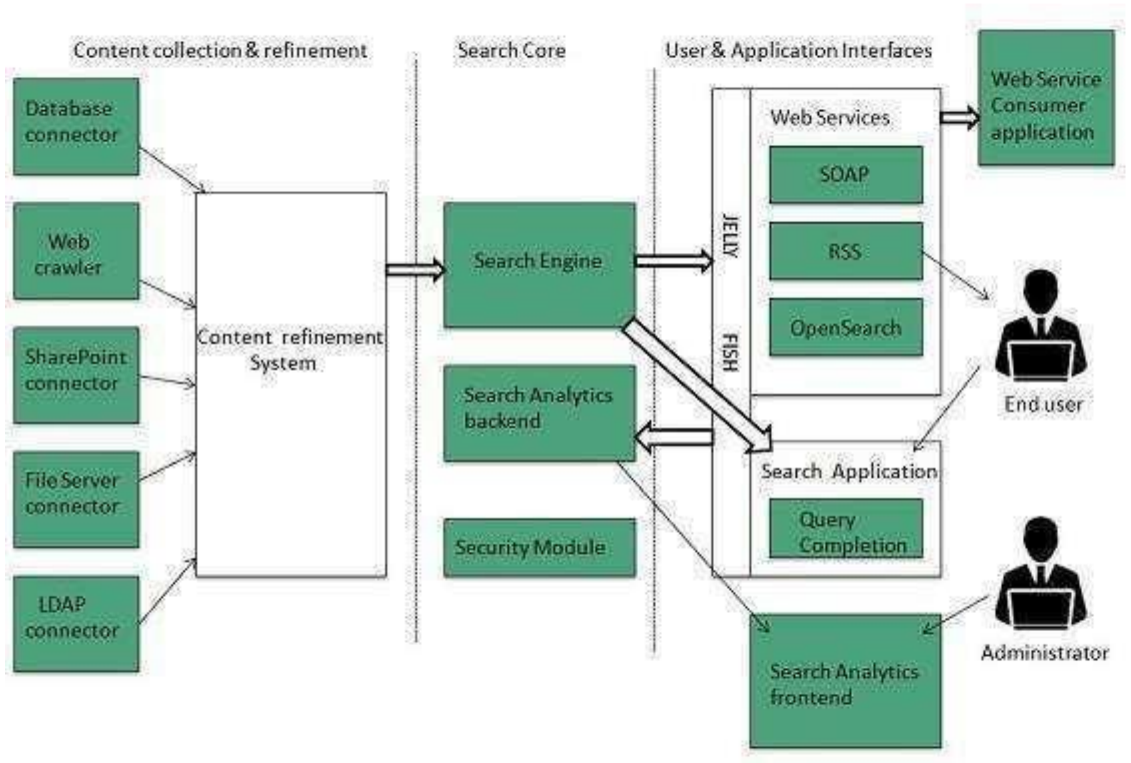


Fig.2 Search Engine archiecture

Software architecture consists of software components, the interfaces provided by those components and the relationships between them.

Architecture of a search engine determined by two requirements:

i.)Effectiveness(quality of results)

ii.)Efficiency(speed: response time and throughput)

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 suport fast searching.

Query Process

Query process comprises of the following three tasks:

User interaction

Ranking

Evaluation

Use of hyperlinks for ranking web search results

Link Analysis:

Such link analysis is one of many factors considered by web search engines in computing a composite score for a web page on any given query.

The web as a graph:

Link analysis builds on two intuitions:

i.)The anchor text pointing to page B is a good description of page B.

ii.) The hyperlink from A to B represents an endorsement of page B, by the creator of page A. This is not always the case; for instance, many links amongst pages within a single website stem from the user of a common template. For instance, most corporate websites have a pointer from every page to a page containing a copyright notice - this is clearly not an endorsement. Accordingly, implementations of link analysis algorithms will typical discount such ``internal'' links.

Ranking:

Ranking of query results is one of the fundamental problems in information retrieval(IR), the scientific/engineering discipline behind search engines. Given a query q and a collection D of documents that match the query, the problem is to rank, that is, sort, the documents in D according to some criterion so that the "best" results appear early in the result list displayed to the user. Classically, ranking criteria are phrased in terms of relevance of documents with respect to an information need expressed in the query.

Ranking is often reduced to the computation of numeric scores on query/document pairs; a baseline score function for this purpose is the cosine similarity between tf–idf vectors representing the query and the document in a vector space model.

Ranking functions are evaluated by a variety of means; one of the simplest is determining the precision of the first k top-ranked results for some fixed k; for example, the proportion of the top 10 results that are relevant, on average over many queries.

HITS algorithm:

Hyperlink-Induced Topic Search (HITS; also known as hubs and authorities) is a link analysisalgorithm that rates Web pages, developed by Jon Kleinberg. The idea behind Hubs and Authorities stemmed from a particular insight into the creation of web pages when the Internet was originally forming; that is, certain web pages, known as hubs, served as large directories that were not actually authoritative in the information that they held, but were used as compilations of a broad catalog of information that led users direct to other authoritative pages. In other words, a good hub represented a page that pointed to many other pages, and a good authority represented a page that was linked by many different hubs.

Algorithm:

In the HITS algorithm, the first step is to retrieve the most relevant pages to the search query. This set is called the root set and can be obtained by taking the top pages returned by a text-based search algorithm. A base set is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it. The web pages in the base set and all hyperlinks among those pages form a focused subgraph. The HITS computation is performed only on this focused subgraph. According to Kleinberg the reason for constructing a base set is to ensure that most (or many) of the strongest authorities are included.

The algorithm performs a series of iterations, each consisting of two basic steps:

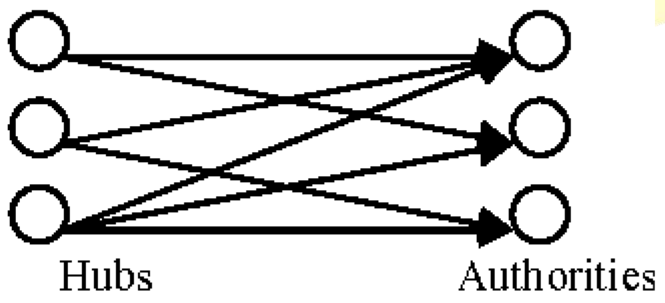


Figure 1. Hubs and authorities

$$a_i = \sum_{j \in B(i)} h_j \quad \text{and} \quad h_i = \sum_{j \in F(i)} a_j$$

Authority update: Update each node's authority score to be equal to the sum of the hub scores of each node that points to it. That is, a node is given a high authority score by being linked from pages that are recognized as Hubs for information.

Hub update: Update each node's hub score to be equal to the sum of the authority scores of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

The Hub score and Authority score for a node is calculated with the following algorithm:

- Start with each node having a hub score and authority score of 1.
- Run the authority update rule
- Run the hub update rule
- Normalize the values by dividing each Hub score by square root of the sum of the squares of all Hub scores, and dividing each Authority score by square root of the sum of the squares of all Authority scores.
- Repeat from the second step as necessary.

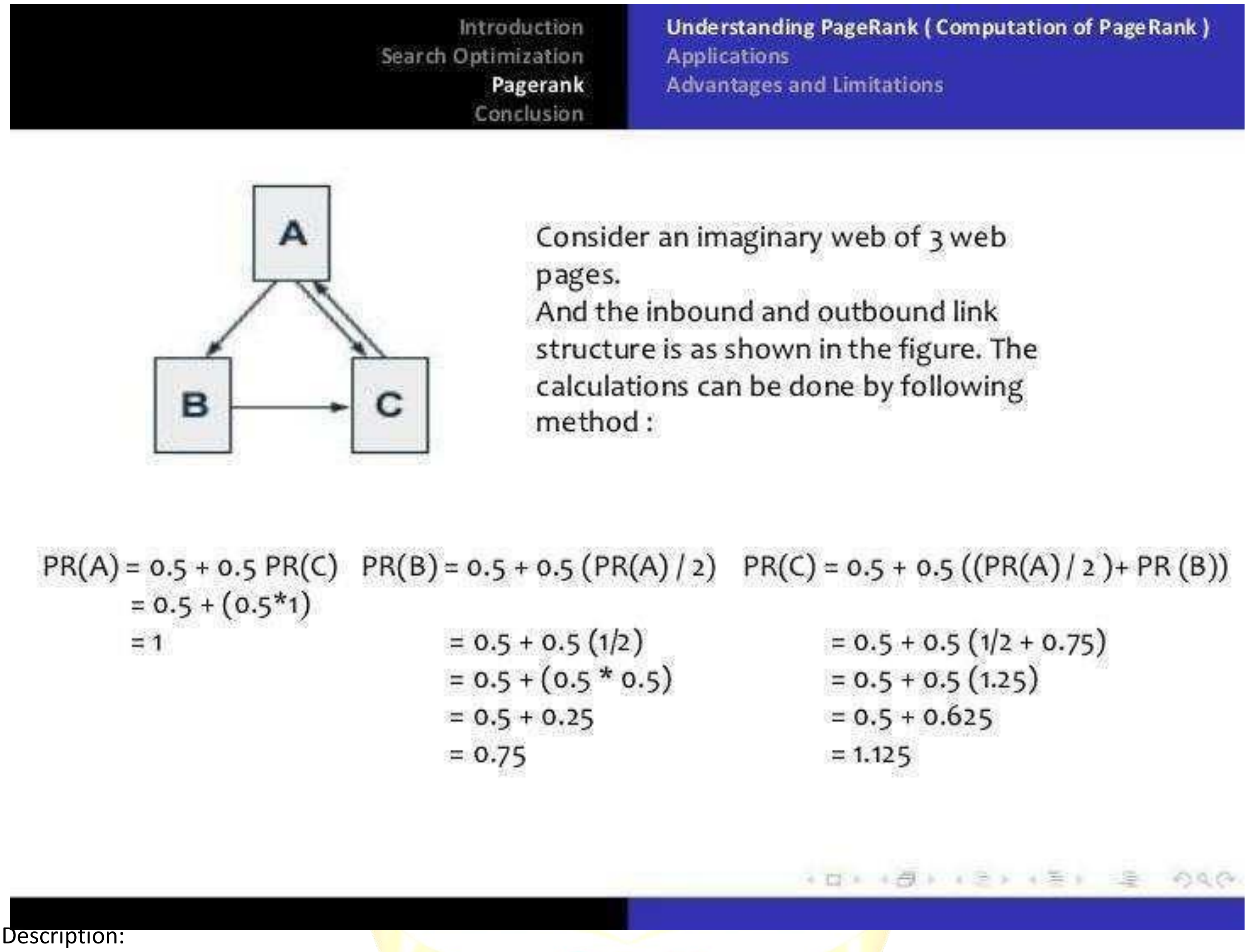
PageRank algorithm:

The Hub score and Authority score for a node is calculated with the following algorithm:

- Start with each node having a hub score and authority score of 1.
- Run the authority update rule
- Run the hub update rule
- Normalize the values by dividing each Hub score by square root of the sum of the squares of all Hub scores, and dividing each Authority score by square root of the sum of the squares of all Authority scores.
- Repeat from the second step as necessary.

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known.



$$\begin{tikzcd} & A \arrow[dl] \arrow[dr, shift left] & \\ B \arrow[rr] & & C \arrow[ul, shift left] \end{tikzcd}$$

Consider an imaginary web of 3 web pages.
And the inbound and outbound link structure is as shown in the figure. The calculations can be done by following method :

$$\begin{aligned} PR(A) &= 0.5 + 0.5\,PR(C) \\ &= 0.5 + (0.5*1) \\ &= 1 \end{aligned}$$

$$\begin{aligned} PR(B) &= 0.5 + 0.5\,(PR(A) / 2) \\ &= 0.5 + 0.5\,(1/2) \\ &= 0.5 + (0.5 * 0.5) \\ &= 0.5 + 0.25 \\ &= 0.75 \end{aligned}$$

$$\begin{aligned} PR(C) &= 0.5 + 0.5\,((PR(A) / 2) + PR(B)) \\ &= 0.5 + 0.5\,(1/2 + 0.75) \\ &= 0.5 + 0.5\,(1.25) \\ &= 0.5 + 0.625 \\ &= 1.125 \end{aligned}$$

Description:

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element *E* is referred to as the *PageRank of E* .Other factors like *Author Rank* can contribute to the importance of an entity.

Algorithm:

The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called "iterations", through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

A probability is expressed as a numeric value between 0 and 1. A 0.5 probability is commonly expressed as a "50% chance" of something happening. Hence, a PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to the document with the 0.5 PageRank.

Meta-Searches:

A **metasearch engine** (or aggregator) is a search tool that uses another search engine's data to produce its own results from the Internet.

Metasearch engines take input from a user and simultaneously send out queries to third party search engines for results. Sufficient data is gathered, formatted by their ranks and presented to the users.

Metasearch engines have their own sets of unique problems. All of the websites stored on search engines are different, which draws irrelevant content. Problems such as spammingreduce result accuracy. The process of fusion aims to tackle this issue and improve the engineering of a metasearch engine.

Advantages:

By sending multiple queries to several other search engines this extends the search coverage of the topic and allows more information to be found. They use the indexes built by other search engines, aggregating and often post-processing results in unique ways. A metasearch engine has an advantage over a single search engine because more results can be retrieved with the same amount of exertion.It also reduces the work of users from having to individually type in searches from different engines to look for resources.Metasearching is also a useful approach if the purpose of the user's search is to get an overview of the topic or to get quick answers. Instead of having to go through multiple search engines like Yahoo! or Google and comparing results,

metasearch engines are able to quickly compile and combine results. They can do it either by listing results from each engine queried with no additional post-processing (Dogpile) or by analyzing the results and ranking them by their own rules (IxQuick, Metacrawler, and Vivismo)

Disadvantages:

Metasearch engines are not capable of decodingquery forms or able to fully translate query syntax. The number of links generated by metasearch engines are limited, and therefore do not provide the user with the complete results of a query. The majority of metasearch engines do not provide over ten linked files from a single search engine, and generally do not interact with larger search engines for results

Metasearching also gives the illusion that there is more coverage of the topic queried, particularly if the user is searching for popular or commonplace information. It's common to end with multiple identical results from the queried engines. It is also harder for users to search with advanced search syntax to be sent with the query, so results may not be as precise as when a user is using an advanced search interface at a specific engine. This results in many metasearch engines using simple searching.

Performance evaluation of search engine:

Many metrics exist to perform the task of search engine evaluation that are either looking for the experts judgments or believe in searchers decisions about the relevancy of the web documents. However, search logs can provide us information about how real users search

There are six criteria for search engine evaluation. These criteria are web coverage, dwell time, recall, precision, presentation and user efforts. We can explore the user efforts in the form of session duration, Ranked Precision and Clicks hits

Search Engines have become an integral part of daily internet usage. The search engine is the first stop for web users when they are looking for a product. Information retrieval may be viewed as aproblem of classifying items into one of two classes corresponding to interesting and uninteresting items respectively. A natural performance metric in this context is classification accuracy, defined as the fraction of the system's interesting/uninteresting predictions that agree with the user's assessments. On the other hand, the field of information retrieval has two classical performance evaluation metrics: precision, the fraction of the items retrieved by the system that are interesting to the user, and recall, the fraction of the items of interest to the user that are retrieved by the system.

Criteria for Evaluating Search Engine's Performance:

Many publications compare or evaluate Web search engines (e.g. Notess, 2000). Perhaps the best known of these are Search Engine Watch (http://www.searchenginewatch.com) and Search Engine Showdown (http://www.searchengineshowdown.com).

Recall in Search Engine Performance Evaluation:

Recall has always been a difficult measure to calculate because it requires the knowledge of the total number of relevant items in the collection.

Precision in Search Engine Performance Evaluation:
Precision is always reported in formal information retrieval experiments. However, there are variations in the way it is calculated depending on how relevance judgments were made.

Precisionmeasures the ability of Search Engine to produce only relevant results. Precision is the ratio between the number of relevant documents retrieved by the system and the total number of documents retrieved. An ideal system would produce a precision score of1, i.e. every document retrieved by the system is judged relevant. Precision is relatively easy to calculate, which mainly accounts for its popularity. But a problem with precision in the search engine context is the number of results usually given back in response to typical queries.
In many cases, search engines return thousands of results. In an evaluation scenario, it is not feasible to judge so many results. Therefore, cut-off rates (e.g. 20 for the first 20 hits) are used in retrieval tests.
The coverage of a search enginecan be determined as the total number of pages returned by the search engine.

Search engine Optimization:

Search engine optimization (**SEO**) is the process of affecting the online visibility of a website or a web page in a web search engine's unpaid results—often referred to as "natural", "organic", or "earned" results.

As an Internet marketing strategy, SEO considers how search engines work, the computer programmed algorithms which dictate search engine behavior, what people search for, the actual search terms or keywords typed into search engines, and which search engines are preferred by their targeted audience. Optimizing a website may involve editing its content, adding content, doing HTML, and associated coding to both increase its relevance to specific keywords and to remove barriers to the indexing activities of search engines.

Search Engine Optimization (SEO) is the activity of optimizing web pages or whole sites in order to make them search engine friendly, thus getting higher positions in search results.

SEO stands for Search Engine Optimization. SEO is all about optimizing a website for search engines. SEO is a technique for:

designing and developing a website to rank well in search engine results.

improving the volume and quality of traffic to a website from search engines.

marketing by understanding how search algorithms work, and what human visitors might search.

Search engines such as Google and Yahoo! often update their relevancy algorithm dozens of times per month. When you see changes in your rankings, it is due to an algorithmic shift or something else beyond your control. Although the basic principle of operation of all search engines is the same, the minor differences between their relevancy algorithms lead to major changes in the relevancy of results

What is On-Page and Off-Page SEO?

Conceptually, the re are two ways of optimization:

•On-Page SEO –

It includes providing good content, good keywords selection ,putting keywords on correct places, giving appropriate title to every page ,etc.

•Off-Page SEO-

It includes link building, increasing link popularity by submitting open directories, search engines, link exchange,etc

Unit-5

Introduction to online IR Systems:

The underlying basics of online systems have arguably changed the least over time. Lancaster's overall systems approach, viewing information retrieval as a complex system that can be broken into many separate system components for better understanding, is the approach long favored by researchers who realize that each subsystem must be understood to understand or improve the whole.

While the underlying technology of inverted file structure has improved dramatically to provide efficient retrieval of massive full text databases, the importance was established in early online systems. Although often criticized and now faced with many alternatives, Boolean logic remains the standard for information retrieval systems.

The most common criticism of Boolean logic systems throughout the 1980s and 1990s was that end users had trouble understanding Boolean logic and thus query formulation is too difficult.

One thing that had to be changed to make online systems friendlier or easier to use was to improve the interface. Today's Web search engines have extremely simple interfaces that hide the inner workings of a complex system, and commercial information retrieval systems have improved over the decades, although interfaces for online information retrieval systems are still not considered user-friendly.

Online IR Systems:

The use of the computer for bibliographic information retrieval was first demonstrated in the 1950s, and initiated by the National Library of Medicine in 1964 using batch processing. Also in the 1960s, federally funded projects were carried out to develop prototype online systems which were then implemented in government research laboratories.

The last production service, Lochheed's DIALOG system, was implemented for NASA and subsequently made available to other government locations before becoming a commercial activity in the early 1970's.

Today DIALOG operates worldwide with databases offered via the Internet to Libraries and other organizations as well as individuals.

With a few exceptions, database vendors do not produce information but rather make it available to searchers via a common search interface. Database vendors license databases from the Producers, process the databases to introduce as much standardization as is feasible(e. g. standard field names), mount the database through the creation of inverted indexes, create database descriptions and aids to searchers in a standard format, and conduct training sessions for clients. These organizations offer a value-added service by providing a common gateway to multiple databases. A database vendor may offer cross-database searches; for example, DIALOG allows the searcher to search simultaneously a predetermined or searcher-selected grouping of databases to create a merged set of references, then process the set to remove duplicates.

IR in Online Retrieval Systems:

Since the inception of these online retrieval services, their retrieval functionality has been primarily on the Boolean model for retrieval, in contrast to research in the IR field which has focused on improving retrieval performance through non-Boolean models, such as the vendor space model. A number of factors guided the choice of the Boolean model as the basis for these services. Research in indexing and retrieval at the time, particularly the cranfield studies, a series of experiments comparing natural and controlled vocabulary indexing, suggested that a natural language" retrieval provided a level of retrieval performance comparable to manual indexing. Boolean logic was already being used in some libraries for manual retrieval systems, such as edge-notched cards and optical coincidence cards, and seemed to offer a natural mechanism for implementing retrieved based on combinations of words in documents.

Despite developments in IR research which suggested that alternative models might provide improved retrieval performance, Boolean retrieval has remained the commonest access method offered by database vendors, although in recent years some systems have added a form of natural language input with ranked output processing as an alternative access method.

Online Public Access Catalogs (OPACs):

An OPAC (Online Public Access Catalog) is an online bibliography of a library collection that is available to the public. OPACs developed as stand-alone online catalogs, often from VT100 terminals to a mainframe library catalog. With the arrival of the Internet, most libraries have made their OPAC accessible from a server to users all over the world.

User searches of an OPAC make use of the Z39.50 protocol. This protocol can also be used to link disparate OPCS into a single "union" OPAC.

Although a handful of experimental systems existed as early as the 1960s, the first large-scale online catalogs were developed at Ohio State University in 1975 and the Dallas Public Library in 1978.

The newest generation of library catalog systems are distinguished from earlier OPACs by their use of more sophisticated search technologies, including relevancy ranking and faceted search, as well as features aimed at greater user interaction and participation with the system, including tagging and reviews. These new features rely heavily on existing metadata which is often poor or inconsistent, particularly for older records.

These newer systems are almost always independent of the library's integrated library system (ILS), instead providing drivers that allow for the synchronization of data between the two systems. While older online catalog systems were almost exclusively built by ILS vendors, libraries are increasingly turning to next generation catalog systems built by enterprise search companies and open source projects, often led by libraries themselves.[5][6] The costs associated with these new systems, however, have slowed their adoption, particularly at smaller institutions.

An example of a next generation OPAC system is included in the Libramatic software package.

Three generations of OPAC:

First generation: known-item finding tools through search by author, title, control number.Phrase searching OPACs', as they are generally called, were in a way the machine readable forms of conventional catalogues providing such access points as class mark, author, title , subject as phrase and simple left to right phrase matching. Such systems had certain obvious drawbacks, for the probability of exact matching between search phrases with indexing terms was rather small: Much of the computer capabilities were wasted as the system worked like a card catalogue. It was not user-friendly as user/system interaction was quite limited.

Second generation: increased search technology with access by subject headings, keywords, boolean queries problems included failed searches, navigational confusion enhancements represented large investments for a library.

Most of the existing OPACs are still at this stage. Influenced by the commercial bibliographic database, second generation OPACs have adopted many of their features likes 'online help messages', `alphabetical index displays' for searching search terms and using `Boolean logic' for their combination and effective retrieval.

Despite the improvements, the second generation OPACs have made the first generation, Hildreth regards them as 'deficient tools' for effective subject searching, for the following reasons:

❖ They offer little or no help in translation of entry query terms into the vocabulary used in the catalogue;

❖ They provide no help to the user in making alternate search statements and techniques, when the initial approach fail;

❖ They do not in all cases lead to a successful free text search(e.g. of the title words); to the corresponding subject headings or class numbers assigned to a broader range of related material;

❖ The retrieval records are generally devoid of such information as table of contents , abstracts and book reviews, that might help user to judge the usefulness of the documents;

Third generation: focus on open systems architectures, improved GUI, support for Z39.50 and Dublin Core, hypertext links, java programming, ranked results sets problems include slow pace of development failure to match trends in the Web.

The above listed deficiencies were investigated and some of the remedies that emerged were incorporated into third generation OPACs to enhance their subject searching capability. These systems are enriched by the inclusion of additional controlled and uncontrolled access points. Queries are accepted as a 'natural language' statement eliminating the need for the user to know quarry formulation and search techniques. Some of the systems use partial match techniques instead of Boolean operators. The retrieved sets are sometimes ranked according to the query relevance. These catalogues ensure vastly improved search system interaction at every level of the search process.

Digital Libraries:

Digital Libraries (DLs) are advanced and complex information (retrieval) systems, which offer many valued services besides searching and browsing, such as document preservation and recommendation reference services selective information dissemination, among others. All services are provided over various types of multimedia data (e.g., audio, video) in a distributed fashion.

Definition of Digital Libraries: DLs are organized and focused collection of digital objects, including text, images, video, and audio, along with methods of access and retrieval, and for selection, creation, organization, maintenance, and sharing of the collection.

Traditional libraries are among the first institutions to use IR systems, to create catalogs of records for the material from the library. The catalogs can be search by users in the library or over the Web (online public access catalogs). These catalogs use database technology; the records are structured according to standards such as MARC (title, a few subject headings, and a classification number).

The basic 'Ss' and fundamental concepts of a 'minimal' DL.

Streams: sequences of arbitrary items (e.g., bits, characters, pixels, images) representing the content of a DL.

Structures: can be viewed as labeled directed graphs, which impose organization on the DL content.

Spaces (e.g., vector, probabilistic): used as support for services and for presentation purposes; can be seen as sets with operations that obey certain mathematical constraints.

Scenarios: stories that describe the behavior of services and consist of sequences of events or actions that modify states of a computation in order to accomplish a functional requirement.

Societies: sets of entities and activities and the relationships among them.

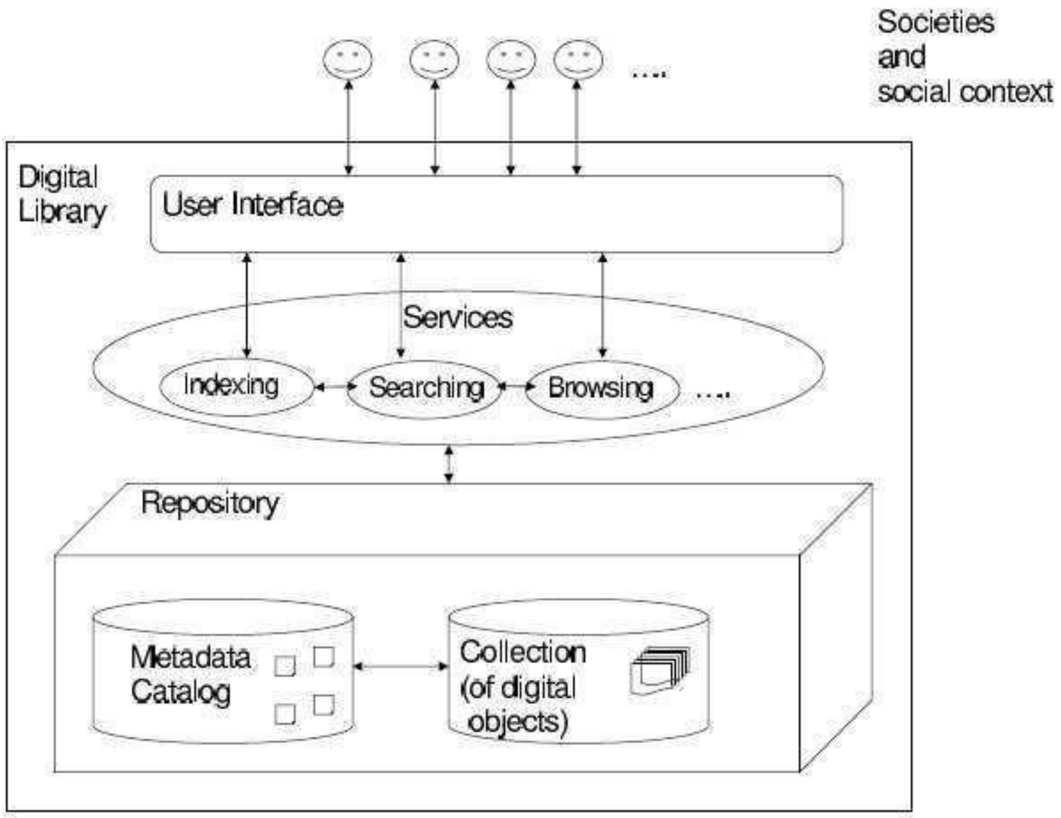


Figure 5.1: General Reference Architecture

- A DL is comprised of a collection of digital objects (e.g., digital documents, images, etc) and a catalog of metadata records that serve either to describe, to organize, or to specify how the objects in the collection can be used and by whom.

- Ideally every object should have a corresponding metadata record in the catalog, and this record should have a specific structure defined by a schema.
- Collections and catalogs are usually stored together in a repository that provides access and management capabilities to collections and catalogs.
- Services to create digital objects or metadata records, to preserve content, to add value to it, and to satisfy information needs are built on top of the repositories and are used by actors in a social context.
- Services can cooperate in terms of reuse or extension of capabilities to create more advanced services from simple ones.
- Usually a digital library provides simple searching and browsing services and indexing services to support the former two as a minimal set of services.
- The user interface serves as a ”glue" to organize and display all the provided services.

Modern libraries are being transformed to digital libraries as a result of the growth in electronic publishing, which makes information available in a digital form. Through the Web, a single interface provides access to local resources, as well as remote access to databases in the sciences, humanities, and business, including full-text journals, newspapers, and directories.
Special collections, in multimedia not only in text format, become available through the same gateway. For more details about the technology of digital libraries see, for example, (Lesk, 1997). Many libraries, particularly academic and large public libraries, have undertaken digital library project to achieve interoperability and ease of use and access. Two such projects are the Los Angeles Public Library's Virtual Electronic Library project (http://www.lapl.org), and University of Pennsylvania's Digital Library (http://www.library.upenn.edu).
A digital library could have no connection to an actual library, for example the ACM Digital Library (http://portal.acm.org/dl.cfm) that contains journal and conference publications in Computer Science. Digital libraries are more than complex IR systems. They are social systems centered around various communities of users. They also have component for building, cataloging, maintaining, and preserving repositories. There are many international or national digital library projects. One such project is the Digital Libraries Initiative (DLI) (phase one 1994-1998, phase two in progress) supported by the National Science Foundation (NSF), the Department of Defense Advanced Research Projects Agency (DARPA) and the National Aeronautics and Space Administration (NASA).

The DLI phase one contained large research projects at six universities: University of Illinois Urbana-Champaign, Carnegie-Mellon University, Stanford University, University of California at Berkeley, University of California at Santa Barbara and University of Michigan. These projects are developing the next generation of tools for information discovery, management, retrieval and analysis.

Web Personalization:

Global information retrieval and anywhere, anytime information access has stimulated a need to design and model the personalized information search in a flexible and agile way that can use the specific personalization techniques, algorithms, and available technology infrastructure to satisfy high-level functional requirements for personalization.

Personalized Information Retrieval and Access: Concepts, Methods and Practices surveys the main concepts, methods, and practices of personalized information retrieval and access in today's data intensive, dynamic, and distributed environment, and provides students, researchers, and practitioners with authoritative coverage of recent technological advances that are shaping the future of globally distributed information retrieval and anywhere, anytime information access.

Except when indexes are updated, a simple search engine delivers the same results for a query, But in reality, search performance may be improved if answers to the following questions can be obtained

- Who is searching?
- What role are they playing?
- What are they interested in?
- Why are they searching?
- Where are they located?

Personalized information retrieval represents only one aspect of a broad field of Personalization research Teevan et al quantified the potential for gain from personalizing search For queries supplied by the experimenters, they found a diversity of imputed intents Even when the imputed intents were the same, subjects disagreed substantially in the ratings they gave to results.

Pitkow et al studied the value of using a client-side personalization system termed as Outride.

Outride builds up a model of the user-based on their searching, browsing, demographic and application use profile.

Search results are re-ranked with reference to a vector-space representation of the user's profile.

The authors observed dramatic reductions in both:

(a) The time taken to complete a search task

(b) The number of user actions such as mouse clicks or keyboard entries

(c) Search tools can be customized according to characteristics of groups, individuals, or tasks

(d) There is also a particular potential for contextualizing search by employees within an organization