1 Introduction to Information Retrieval

UNIT I

Syllabus

Introduction to Information Retrieval, Basic Concepts, Information Versus Data, Trends and research issues in information retrieval.

The retrieval process, Information retrieval in the library, web and digital libraries.

1.1 Introduction and Basic Concepts of IR

Q. Explain basic concept of Information Retrieval.

- This is the information era. We handle vast amount of information. The purpose to maintain such huge amount of information is when we need any information; we should get it as early as possible.
- We require speedy and accurate access whenever we need it.
- One method to get relevant information is read all the documents and then decides which are the relevant and which are the non-relevant documents? This is the manual method.
- Second method is the automatic method in which we store all the information in computer and ask to find the relevant information. Information retrieval handles the representation, storage, organization and access to information items. The representation of information should require less space. The organization of information should be such that, system should require less time to access the items of information which satisfies user needs.
- Following are important terminologies in IR :

1. **Data**
   - Raw representation of collections from various sources or a single source is data.
   - Data may or may not contain information.
   - Data can be either quantitative or qualitative.
   - Qualitative data is in descriptive form and quantitative data is in numerical values and figure form.
   - To utilize data for the decision making process, it is important to convert those into information.
2. **Information**
   - Data which has significance in decision making or initiating some action is information.
   - Information is derived from data.

3. **Knowledge**

- When information is processed, linked, and stored by a human or by a machine then information becom knowledge.
- Patterns acquired from information to help decision making repeatedly is knowledge.
- As more and more information is given, knowledge can increase.
- Knowledge is a collection of information about a person or a for a fact that makes it possible to make decis or solve any issue.
- Knowledge is also defined as the meaningful links which people make in their minds by utilizing informat and its actionable applications in a specific setting.
- Knowledge information and data pyramid is as shown in Fig. 1.1.1.

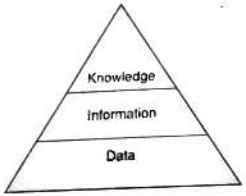


Fig. 1.1.1

- Data can be individual facts, figures, signals, measurements etc.
- Information can be organized, structured, categorized, useful, condensed, calculated etc.
- Knowledge can by idea, learning, notion, concept, synthesized, compared, thought-out, discussed.
- While data is not useful until it is converted to information, the information is used to make strategic important decisions by using the knowledge.
- With available technological tools, it is easy to convert data into information that is helpful in decision mak process.

4. **Retrieval**

- From already present data trying to search or find some particular pattern given by user is retrieval.
- All possible match to pattern given by user are returned by the retrieval system.
- Retrieval can be based on search words, keys, or any other specific data.
- Data can be stored in form of documents or database or other formats.
- Data retrieval identifies and extract the data from the database based on a query provided by the use application.
- Information retrieval obtain information system resources that are relevant to an information need fr collection of those resources. Searches can be based on full text or other content based indexing.
- Knowledge retrieval return information in a structured form, consistent with human cognitive processes

1.2 Information vs. Data Retrieval

Q. Differentiate between data and information retrieval.

- Data retrieval mainly concerned with determining which documents contain specific words in the query. In information retrieval, the user is interested in the relevant information of the query.
- Table 1.2.1 is the comparison of data retrieval and information retrieval.

Table 1.2.1

Sr. No.	Parameters	Data Retrieval (DR)	Information Retrieval (IR)
1.	Matching	Exact match	Partial match, best match
2.	Inference	Deduction	Induction
3.	Model	Deterministic	Probabilistic
4.	Classification	Monothetic	Polythetic
5.	Query language	Artificial	Natural
6.	Query specification	Complete	Incomplete
7.	Items wanted	Matching	Relevant
8.	Error response	Sensitive	Insensitive

1. **Matching**

- In data retrieval, we normally search for exact match for e.g. whether a file contains a particular word or not.
- In information retrieval, we normally find documents which partially match the request and then select best out of them.

Inference

- The inference which get used in data retrieval is deductive e.g. if $a \rightarrow b$ and $b \rightarrow c$ then $a \rightarrow c$.
- In Information Retrieval we follow inductive inference i.e. relations are specified with a degree of certainty or uncertainty.

Model

- As Data Retrieval uses deductive inference, deterministic models can be helpful for finding the relevant documents.
- As the Information Retrieval uses inductive inference, we can use models which give the probabilistic output.

Classification

- In Data Retrieval we are interested in monothetic classification i.e. one with classes defined by objects possessing attributes, both necessary and sufficient, to belong to a class.
- In Information Retrieval monothetic classification is not required.

- In Information Retrieval polythetic classification gets used.
- In such a classification each individual in a class will possess only a proportion of all attributes possessed by members of that class. Hence no attribute is necessary nor sufficient for membership of a class.

5. **Query language**
   - The query language which is used in Data Retrieval is mostly artificial; with restricted syntax and vocabula e.g. we can write a query in SQL in its fixed format with fixed words.
   - In Information Retrieval, query will have no restriction related to syntax or vocabulary.
   - User can provide a query in natural language format. The Information Retrieval system should be able handle such queries.
6. **Query specification**
   - As Data Retrieval finds exact match and the query follows a restricted format, the query must be compl User should provide th`e exact query for his interest.
   - In Information Retrieval, user can use natural language to specify the query and hence it is possible that query may be incomplete e.g. user will not follow the standard grammar of the language. The Informa Retrieval system can handle such queries.
7. **Items wanted**
   - In Data Retrieval, user specifies the exact query and hence the list of items will contain those items wh exactly match the query.
   - In Information Retrieval, the query gets specified in natural language as well as the models which used finding the items are probabilistic, the system will find items which are relevant to the query.
   - User then will decide for best one from the listed output.
8. **Error response**
   - In Data Retrieval, the query must be complete with proper syntax. Hence if there will be any error w specifying the query, meaning will be totally different and we can get wrong items.
   - In Information Retrieval, query gets specified in natural language, hence some sort of relaxation can handled by the system.

1.2.1 Difference between Information and Data Retrieval

Table 1.2.2

Sr. No.	Information Retrieval	Data Retrieval
1.	Information retrieval obtain information system resources that are relevant to an information need from a collection of those resources. Searches can be based on full text or other content based indexing	Data retrieval identifies and extract the data from the database based on a query provided by the user or application

Sr. No.	Information Retrieval	Data Retrieval
2.	For a given subject retrieve the information	Retrieve data for the user query
3.	Small errors can be ignored	A single error can cuse the total failure
4.	Information will not always structred and it can be ambiguous	Data has well defined structure and semantic
5.	The retrieved results are approximate matches	The retrieval result are exact matches
6.	Obtained results can be ordered by relevance	Obtaibed results are unordered by relevance
7.	It is probabilistic model	It is determinstic model

1.3 Trends and Research Issues in Information Retrieval

Information Retrieval is related to text, images, audio, video or object oriented type of information. IR deals with the efficient way of storage of the information and various methods of searching the information based on user's interest. Handling textual information is subdomain of IR.

IR is more to do with search engines where we have large amount of information and based on user's requirement, specific information is extracted from the collection. IR is more hybrid topic which converts, machine learning techniques, Natural Language processing techniques. Now-a-days, more focus of IR is on search engines.

1.3.1 Major Issues in IR

Indexing of Input Data-Main goal of Document and Query Indexing is to find important meanings and creating an internal representation. The factors to be considered are accuracy to represent semantics, exhaustiveness, and facility for a computer to manipulate.

Retrieval of Data as per Query-In the retrieval model how can a document be represented with the selected keywords and how are documents and query representations compared to calculate a score. Information Retrieval (IR) deals with issues like uncertainty and vagueness in information systems.

- **Uncertainty :** The available representation does not typically reflect true semantics of objects such as images, videos etc.
- **Vagueness :** The information that the user requires lacks clarity, is only vaguely expressed in a query, feedback or user action.

Evaluation of an IR system-System Evaluation tells about the importance of determining the impact of information given on user achievement. Here, we see if the efficiency of the particular system related to time and space.

Selection of search vocabulary, search strategy formulations and information overload.

- Partial matches are not retrieved. Only those documents that exactly match the query are retrieved. Hence effective retrieval requires the users to have a good domain knowledge to form good queries for a large set of documents.
- Retrieved documents are not ranked.
- For a given a large set of documents the Boolean model either retrieves too many documents or very few documents.
- The model does not use term weights. If a term occurs only once in a document or several times in a document it is treated in the same way.

1.3.2 Trends in IR

- In terms of novel necessities, with the diffusion of the Internet and the heterogeneous characteristics of user search engines, which can be regarded as the new frontier of IR, a new central issue had arisen, generally known the semantic web. It mainly consists in expanding Information Retrieval Systems (IRSs) with the capability represent and manage the semantics of both user requests and documents so as to be able to account for and document contexts.
- Another research trend, according to Kraft, Bordogna and Pasi, is motivated by the need to manage multim collections with non-print audio elements such as sound, music, and voice, and video elements such as ima pictures, movies, and animation. Retrieval of such elements can include consideration of both metadata content-based retrieval techniques.
- In addition, modern computing technology, including storage media, distributed and parallel proce architectures, and improved algorithms for text processing and for retrieval, has an effect on IRSs. For exar improved string searching algorithms have improved the efficiency of search engines. Improved com networks have made the Internet and the World Wide Web a possibility.
- These novel research trends in IR are faced by turning to technologies such as natural language processing processing, language models, artificial intelligence, and automatic learning. Also fuzzy set theory can play a c role to define novel solutions to these research issues since it provides suitable means to cope with the nee the semantic web (Sanchez, 2006) i.e., to model the semantic of linguistic terms so as to reflect their vaguenes subjectivity and to compute degrees of similarity, generalization, and specialization between their meanings

1.4 Information Retrieval Process

Q. Draw and explain IR system block diagram.

Q. Draw IR system block diagram.

Q. What is a document representative? Explain with a suitable example.

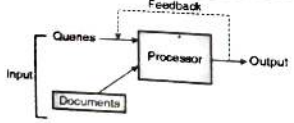


Fig. 1.4.1 : Information Retrieval system block diagram

An information retrieval system deals with representation, storage, organization and access of information. Fig. 1.4.1 shows the block diagram of a typical Information Retrieval system.

The input for this system is the list of documents which contain the information and the query given by the user. The Information Retrieval system will find the list of documents which are relevant to the query.

The main problem here is to obtain representation of each document and query suitable for computer to use. At the first step the documents are converted into its representation. The documents in its natural language format are one of the representations. But if we store these documents in natural language format, space requirement gets increased as well as time to retrieve the items which are relevant to query is also large.

Hence most computer based retrieval systems store only representation of the document. A **document representative** could be a list of extracted words considered to be significant. These words are called as **keywords**. Text of a document is lost once it has been converted into document representation.

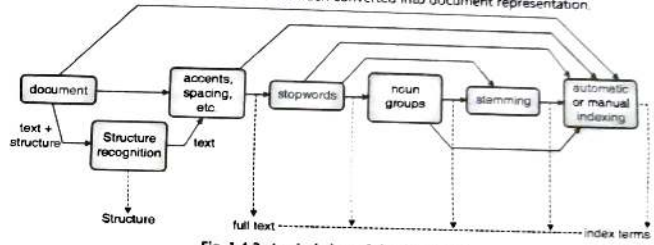


Fig. 1.4.2 : Logical view of the document

Fig. 1.4.2 shows the logical view of the document. A full text can be document representative or list of keywords (index terms) can be a document representative or any intermediate status of the document can be document representative.

As the document gets converted in its internal representation, the queries given by the user are also converted in the same fashion.

The Information Retrieval system will internally find the relevant documents compared with the query and output i.e. the list of relevant documents will be provided to the user.

User can stop here or if user wants to refine the query, he can provide the feedback based on which the query gets modified.

The same process will be done with modified query and finally the output i.e. list of relevant documents are provided to the user.

Information retrieval systems are of two types, one is manual retrieval system and another is automatic retrieval system. Here we are discussing about automatic retrieval system.

In automatic retrieval system computer searches for the relevant documents related to the given query.

Before computerized Information Retrieval system can actually operate the documents to retrieve information, the documents must be stored inside the computer one way to store the documents is in its natural format i.e. text format.

- But the disadvantage of this method is requires more space in memory to store the documents. And the second which searching for the query relevant documents, system require more time.
- Solution for this problem is to find the **document representative** for each document. It can be a title, an abstra or a list of words from that document. Mostly the lists of words from the documents are used as the docume representative.
- The words are those words from documents which contain the semantic of the document. These words are call as keywords.

1.4.1 Objectives and Functions of IRS

- The major objective of an IRS is to retrieve the required information whenever needed.
- It is either the actual information or through the documents containing the information surrogates that fully partially match the user's query.
- Thus, the search output may contain bibliographic details of the documents that matches the query, or the act text, image, video, etc. that contain the required information.
- The database in case of an information retrieval system may contain abstracts or full texts of documents, l newspaper articles, handbooks, dictionaries, encyclopedias, legal documents, statistics, etc., as well as aud images, and video information.
- The major functions of an IRS are:
  1. To identify the sources of information relevant to the areas of interest of the target users' community;
  2. To analyze the contents of the sources (documents);
  3. To represent the contents of the analyzed sources for matching with the users' queries;
  4. To match the search statement with the stored database;
  5. To retrieve the information that is relevant; and
  6. To make necessary adjustments in the system based on feedback from the users

1.5 Information Retrieval in the Library, Web and Digital Libraries

- Information retrieval systems are of two types, one is manual retrieval system and another is automatic retrie system. Here we are discussing about automatic retrieval system.
- In automatic retrieval system computer searches for the relevant documents related to the given query.
- Before computerized Information Retrieval system can actually operate the documents to retrieve information, t documents must be stored inside the computer one way to store the documents is in its natural format i.e. t format.
- But the disadvantage of this method is requires more space in memory to store the documents. And the second which searching for the query relevant documents, system require more time.
- Solution for this problem is to find the **document representative** for each document. It can be a title, an abstra or a list of words from that document. Mostly the lists of words from the documents are used as the docume representative.

- The words are those words from documents which contain the semantic of the document. These words are called as keywords.

1.5.1 Information Retrieval in Library

Libraries contain information in various physical forms. While for many users, the book is still a major vehicle for communication of information; for others, the periodical or the technical report have taken its place; and for yet others, films or gramophone records are significant.

It is clear that the same work can appear in various physical forms. The intellectual content will be the same in each case, but obviously it is not practical to try to arrange the different physical forms together.

We cannot, therefore, rely on the physical arrangement of the items in a library to gather different versions of the same work. We have to rely on a substitute – a set of records (surrogates) of the content of the library. These are in the form of library catalogues and bibliographies.

The information retrieval system serves as a bridge between the world of creator or generation of information and the users of that information.

1.5.2 Web IR

Web IR can be defined as the application of theories and methodologies from IR to the World Wide Web.

It is concerned with addressing the technological challenges facing Information. The utility and ubiquity of web search is making Web Information Retrieval (IR) an increasingly popular research topic.

The incredible success of Google is a striking example of how important it is for academia and industry to foster innovation in the field of Web IR.

The unabated growth of the Web and the increasing expectation placed by the user on the search engine to anticipate and infer his/her information needs and provide relevant results has fostered the development of the field of Web Information Retrieval (Web IR).

The four main components of Web Search are Crawling, Indexing, Querying and Ranking. During the web search the first two are performed offline and last two are done online. Crawling and Indexing is query independent and Querying and Ranking is query dependent.

The model of Web IR has two components. First is a set of premises and second is an algorithm to be used for ranking documents retrieved in response to a user query.

.5.3 IR in Digital Libraries

A digital library, also called an online library, an internet library, a digital repository, or a digital collection is an online database of digital objects that can include text, still images, audio, video, digital documents, or other digital media formats or a library accessible through the internet.

Objects can consist of digitized content like print or photographs, as well as originally produced digital content like word processor files or social media posts.

In addition to storing content, digital libraries provide means for organizing, searching, and retrieving the content contained in the collection.

- Digital libraries can vary immensely in size and scope, and can be maintained by individuals or organizations.
- The digital content may be stored locally, or accessed remotely via computer networks. These information retrie systems are able to exchange information with each other through interoperability and sustainability.
- Even with the expansive coverage of some IR systems, such as Web search engines, they are often part of a larg collection of services or activities. An alternative perspective, especially when communities and/or proprieta collections are involved, is the digital library.
- Digital libraries and commercial publishing ventures need mechanisms to ensure that documents have persist identifiers so that when the document itself physically moves, it is still obtainable.
- The original architecture for the Web envisioned by the Internet Engineering Task Force was to have every uniq resource locator (URL), the address entered into a Web browser or used in a Web hyperlink, linked to a uniq resource name (URN) that would be persistent.
- The combination of a URN and URL, a uniform resource identifier (URI), would provide persistent access to dig objects. The resource for resolving URNS and URIs was never implemented on a large scale.

Review Questions

Q. 1 Differentiate between Data Retrieval and Information Retrieval.

Q. 2 Explain Luhn's Idea.

Q. 3 With the help of block diagram explain typical Information Retrieval System.

Q. 4 Write steps required to conflate the following words :Here, Hereafter, Hereby, Herein, Hereupon.

Q. 5 Explain conflation algorithm in detail.

Q. 6 Explain with example signature file structure.

Q. 7 Compare with example suffix array and suffix tree.

2 UNIT II

Modeling in Information Retrieval

Syllabus

Taxonomy of Information Retrieval models, Classic Information Retrieval, Alternate set: Theoretical model, Alternative Algebraic models, Alternative Probabilistic models Structured text Retrieval models, Models for browsing

2.1 Introduction

- In this chapter we discuss the main techniques needed to implement the query operations.
- A ranking algorithm operates according to basic premises regarding the notion of document relevance. Distinct set of premises yield distinct information retrieval models.
- The IR model adopted determines the predictions of what is relevant and what is not relevant.
- The chapter also covers the most important information retrieval models proposed over the years.
- This chapter also concerned with the way in which file structures are used in document retrieval.
- This chapter also discusses the search strategies for the document.

2.2 Taxonomy of Information Retrieval Models

Q. Explain various IR models in details with their advantages and disadvantages.

- Fig. 2.2.1 illustrates taxonomy of these information retrieval models.
- So the fundamental premises which the basis for a ranking algorithm determine the IR models.

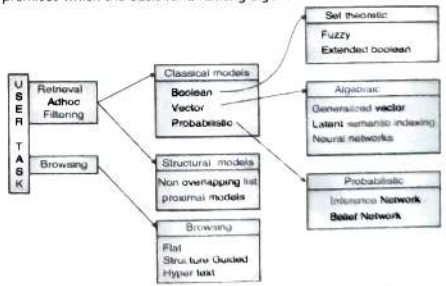


Fig. 2.2.1 : A taxonomy of information retrieval models

- All above models are discussed in this chapter one by one

2.2.1 Definition of Information Retrieval Models

- An information retrieval model is a quadruple [D, Q, F, R (q_i, d_j)]

Where,

D : Set composed of logical views or representations for the documents in the collection.

Q : Set composed of logical views or representations for the user information needs and such representations are called queries.

f : Framework for modeling document representations, queries and their relationships.

R (q_i, d_j) : Ranking function which associates a real number with a query $q_i \in Q$ and a document representation $d_j \in D$, such ranking defines an ordering among the documents with regard to the query q_i.

- For building model we must think of representations for the documents and the user information need.
- So given these representations we then conceive the framework in which they can be modelled.

2.2.2 Basic Concepts of Information Retrieval Models

- The classical models in information retrieval system consider that each document is described by a set representative keywords called index terms.
- An index term is simply a word whose semantics helps in remembering the documents main themes.
- So index terms are used to index and summarize the document content.
- Given a set of index terms for a document, we notice that not all terms are equally useful for describing t document contents.
- It should be clear that distinct index terms have varying relevance when used to describe document contents. A this effect can be captured through the assignment of numerical weights to each index.
- This weight quantifies the importance of the index term for describing the document semantic content. And t index term weights are usually assumed to be mutually independent.
- We will use the following definitions support for discussing the three classic information retrieval models.
  - t be the number of index terms in the system.
  - K_i be a generic index term.
  - $K = (K_1, K_2, \ldots K_t)$ is the set of all index terms.
- A weight $\omega_{i,j} > 0$ is associated with each index term K_i of a document d_j.
- For an index term which does not appear in the document text $\omega_{i,j} = 0$.
- With the document d_j is associated an index term vector $\vec{d_j}$ represented by

$$\vec{d_j} = (\omega_{1,j}, \omega_{2,j}, \ldots, \omega_{t,j}).$$

- g_i be a function that returns the weight associated with the index term K_i in any t-dimensional vector $g_i(\vec{d_j}) = \omega_{i,j}$

2.2.3 Boolean Model

Q. Explain Boolean model in detail.
Q. Describe Boolean model.

- Boolean model is a simple retrieval model based on set theory and Boolean algebra.
- Boolean model provides a framework which is easy to grasp by a common user of an IR system.
- As queries are specified as Boolean expressions which have precise semantics.
- Given its inherent simplicity and neat formation, the Boolean model received great by attention in past years and was adopted by many of the early commercial bibliographic system.
- Boolean model suffer from following drawbacks.

1. Retrieval strategy is based on a binary decision criterion i.e. a document is predicted to be either relevant or non-relevant without any notion of a grading scale which prevents good retrieval performance.

   So in reality Boolean model is much more a data retrieval model.

2. Every time it is not simple to translate an information need into a Boolean expression as user finds it difficult and awkward to express their query requests in terms of Boolean expressions.

Even though it has above mentioned drawbacks, it is still the dominant model with commercial document database systems and provides a good starting point for those new to the field.

Boolean model considers that index terms are present or absent in a document. As a result the index term weights are assumed to be all binary i.e. $\omega_{i,j} \in \{0, 1\}$.

A query q is composed of index terms linked by three connectives not, and, or.

Thus, a query is essentially a conventional Boolean expression which can be represented as a disjunction of conjunctive vectors i.e. in disjunctive normal form.

The query [q = $K_a \wedge (K_b \vee \neg K_c)$] can be written in disjunctive normal form as [$\vec{q}_{dnf}$ = (1, 1, 1) ∨ (1, 1, 0) ∨ (1, 0, 0)] where each of the components is a binary weighted vector associated with the tuple (K_a, K_b, K_c). These binary weighted vectors are called the conjunctive components of $\vec{q}_{dnf}$.

Fig. 2.2.2 illustrates the three conjunctive components for the query q.

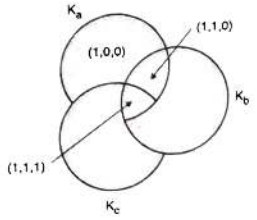


Fig. 2.2.2 : Three conjunctive component for the query [q = $K_a \wedge (K_b \vee \neg K_c)$]

- The similarity of a document d_j to the query q is defined as.

$$sim(d_j, q) = \begin{cases} 1 \text{ if } \exists \vec{q}_{cc} \mid C\ \vec{q}_{cc} \in \vec{q}_{dnf} \wedge \left(\forall K_i, g_i\left(\vec{d}\right) = g_i\left(\vec{q}_{cc}\right)\right) \\ 0 \text{ other wise} \end{cases}$$

- For the Boolean model :
  - The index term weight variable are all binary i.e. $\omega_{i,j} \in \{0, 1\}$.
  - A query q is a conventional Boolean expression.
  - $\vec{q}_{dnf}$ be the disjunctive normal form for the query q.
  - $\vec{q}_{cc}$ any of the conjunctive components or $\vec{q}_{dnf}$.
- If sim $(d_j, q) = 1$ then Boolean model predicts that the document d_j is relevant to the query q otherwise it is n relevant. There is not partial match to the query conditions.
- If d_j is the document for which $\vec{d}_j = (0, 1, 0)$, so document includes the index HTM K_e but is considered no relevant to the query $[q = K_a \wedge (K_b \vee \neg K_c)]$

Advantages of Boolean model

1. This is clean formalism behind the Boolean model.
2. It is very simple.

Disadvantage of Boolean model

- Exact matching may lead to retrieval of too few or too many documents.

2.2.4 Vector Model

Q. Explain vector model in detail.
Q. Describe Vector model.

- The use of binary weights is to limiting in Boolean model so vector model recognized that and proposed framework in which partial matching is possible.
- This partial matching is accomplished by assigning non-binary weights to index terms in queries and in documents.
- These weights are used to compute the degree of similarity between each document stored in the system and the user query.
- Retrieved documents are sorted in decreasing order by using degree of similarity and then vector model takes into consideration the documents which match the query terms only partially.
- So the ranked document answer set is more precise than the document answer set retrieved by the Boolean model.
- The query vector $\vec{q}$ is defined as $\vec{q} = (\omega_{1,q}, \omega_{2,q}, \ldots, \omega_{t,q})$
- The vector for a document d_j is represented by $d_j = (\omega_{1,j}, \omega_{2,j}, \ldots, \omega_{t,j})$.

  Where, for the vector model

  $\omega_{i,j}$ → Weight associated with a pair (K_i, d_j) is positive and non-binary.

  $\omega_{i,q}$ → Weight associated with the pair [ki, q] where $\omega_{i,q} \geq 0$.

- Index terms in the query are weighted.
- A document d_j and a user query q are represented as t-dimensional vectors as shown in Fig. 2.2.3.

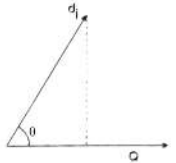


Fig. 2.2.3 : The cosine of θ is adopted as sin (d_j, q)

- The vector model proposed to evaluate the degree of similarity of the document d_j with regard to the query q as the correlation between the vector $\vec{d}_j$ and $\vec{q}$.
- This correlation can be quantified, for instance by the cosine of the angle between these two vectors as.

$$sim(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|}$$

$$= \frac{\sum_{i=1}^{t} \omega_{i,j} \times \omega_{i,q}}{\sqrt{\sum_{i=1}^{t} \omega_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} \omega_{i,q}^2}}$$

Where,

$|\vec{d}_j|$ and $|\vec{q}|$ → The norms of the document and query vectors.

- The factor $|\vec{q}|$ is same for all the document it will not affect on the ranking.
- The factor $|\vec{d}_j|$ provides a normalization in the space of the document.
- As $\omega_{i,j} \geq 0$ and $\omega_{i,q} \geq 0$, sum (q, d_j) varies from 0 to + 1, so instead of attempting to predict whether a document is relevant or not, the vector model ranks the documents according to their degree of similarity to the query.
- A document can be retrieved even though it matches the query only partially.
- One can establish a threshold on sum (d_j, q) and retrieve the document with a degree of similarity above that threshold.
- We need to specify how index term weights are obtained to compute the ranking.

Advantages of vector model

1. The term-weighting scheme used in vector model improves the retrieval performance.
2. Partial matching strategy allows retrieval of documents that approximate the query conditions.
3. Cosine ranking formula sorts the documents according to their degree of similarity to the query.

Disadvantages of vector model

1. Index terms are assumed to be mutually independent.
2. Due to the locality of many term dependencies, their indiscriminate application to all the documents in a collection might hurt the overall performance.

2.2.4(A) Vector Space

$X = (x_1, x_2 \ldots x_n)$ is a vector in n-dimension vector space.

1. **Length :** Length of x is given by,

$$|X|^2 = x_1^2 + x_2^2 + x_3^3 \cdots x_n^2$$

If x_1 and x_2 are vectors,

2. **Inner Product :** Inner product or dot product is given by,

$$x_1 \cdot x_2 = x_{11}\, x_{21} + x_{12}\, x_{22} + x_{13}\, x_{23} \cdots x_{1n}\, x_{2n}$$

3. **Cosine of angle** between the vector x_1 and x_2,

$$\cos(\theta) = \frac{x_1 \cdot x_2}{|x_1| \cdot |x_2|}$$

4. **Similarity :**

Finding similarity between documents ⇒

Documents	Text	Terms
d_1	ant ant bee	ant bee
d_2	dog bee dog hog dog ant dog	ant bee dog hog
d_3	cat gnu dog eel fox	Cat dog eel fox gnu

1. **Term vector space**

	d1	d2	d3
ant	1	1	
bee	1	1	
cat			1
dog		1	1
eel			1
fox			1
gnu			1
hog		1	
length	$\sqrt{2}$	$\sqrt{4}$	$\sqrt{5}$

$t_{ij} = 1$; if term i is present in doc. j

$= 0$; otherwise

Comparing documents (no weighing)

$$\text{Sim}(d_1, d_2) = \frac{d_1 \cdot d_2}{|d_1|\ |d_2|} = \frac{2}{\sqrt{2} \cdot \sqrt{4}} = \frac{2}{2\sqrt{2}} = \frac{1}{\sqrt{2}} = 0.71$$

	d_1	d_2	d_3
d_1	1	0.71	0
d_2	0.71	1	0.22
d_3	0	0.22	1

2. **Similarity of query and documents**

Term vector space :

	q	d_1	d_2	d_3
ant	1	1	1	
bee		1	1	
cat				1
dog	1		1	1
eel				1
fox				1
gnu			1	
hog			1	
length	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{4}$	$\sqrt{5}$

Similarly of documents to query

	d_1	d_2	d_3
q	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{10}}$
	0.5	0.71	0.32

Ex. 2.2.1 : Find the similarity of the following query with documents - D1, D2, D3using vector model.

Query	Keywords	
Q	mouse, dog	
Document	Text	Terms
D1	mouse mouse bee	mouse bee

D2	dog bee dog hog hog dog mouse dog	mouse bee dog hog
D3	cat gnu dog eel fox	Cat dog eel fox gnu

Soln. :

Term vector space :

	q	D_1	D_2	D_3
bee		1	1	
cat				1
dog	1		1	1
eel				1
fox				
gnu				1
hog			1	
mouse	1	1	1	
length	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{4}$	$\sqrt{4}$

$$\text{sim}(d_1, q) = \frac{1}{\sqrt{2} \times \sqrt{2}} = \frac{1}{2} = 0$$

$$\text{sim}(d_2, q) = \frac{2}{\sqrt{4} \times \sqrt{2}} = \frac{\cancel{2}}{\cancel{2}\sqrt{2}} = \frac{1}{\sqrt{2}}$$

$$\text{sim}(d_3, q) = \frac{1}{\sqrt{4} \times \sqrt{2}} = \frac{1}{2\sqrt{2}}$$

∴ Similarly of docs to query :

	d_1	d_2	d_3
q	$\frac{1}{2} = 0.5$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2\sqrt{2}}$

Ex. 2.2.2 : Find the similarity of following query with D1, D2, D3 using vector model.

Query	keywords	
q	ant, dog	
document	Text	Terms
D1	ant ant bee	ant bee

D2	dogbeedoghogdogantdog	ant bee dog hog
D3	cat gnu dog eel fox	catdogeelfoxgnu

Soln. :

Term vector space :

	q	d_1	d_2	d_3
ant	1	1	1	
bee		1	1	
dog	1		1	1
eel				1
fox				1
hog			1	
gnu				1
cat				1
length	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{4}$	$\sqrt{5}$

$$\text{sim}(d_1, q) = \frac{1}{\sqrt{2} \cdot \sqrt{2}} = \frac{1}{2} = 0.5$$

$$\text{sim}(d_2, q) = \frac{2}{\sqrt{4} \cdot \sqrt{2}} = \frac{\cancel{2}}{\cancel{2}\sqrt{2}} = \frac{1}{\sqrt{2}}$$

$$\text{sim}(d_3, q) = \frac{1}{\sqrt{5} \cdot \sqrt{2}} = \frac{1}{\sqrt{5} \cdot \sqrt{2}}$$

Similarity of documents to query :

	d_1	d_2	d_3
q	$\frac{1}{2} = 0.5$	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{5} \cdot \sqrt{2}}$

2.4(B) Difference between Boolean and Vector Models

Q. Compare Boolean and vector model.

Table 2.2.1

Sr. No.	Boolean Model	Vector Model
1.	Here only presence or absence of a term is modeled.	Here weightage of a term is modeled.
2.	Efficient in calculation.	Requires more calculations than Boolean model.

Sr. No.	Boolean Model	Vector Model
3.	There is no way to model global dependency.	Global dependency of a document is represented by inverse document frequency.
4.	Dependencies between documents are partially represented.	Dependencies between documents are better represented.
5.	It is less popular than vector model.	It is useful in many applications.

2.2.5 Probabilistic Model

Q. Explain with formulae basic probability model.

- The probabilistic model tries to fit the IR problem within a probabilistic framework.
- The fundamental idea used in this model is as follows :
  1. With given user query, there are set of documents which contain exactly the relevant documents only.
  2. This set of documents is reformed as ideal answer set.
  3. If the description of this ideal answer set is given then we can easily retrieve its documents.
- Thus, we can think of the querying process as a process of specifying the properties of an ideal answer set.
- We do not know exactly what these properties are, we know only there are index terms whose semantic shou used to characterize these properties.
- As properties are not known at query time so effort has to be made at initially guessing these properties.
- This initial guess allows up to generate a preliminary probabilistic description of the ideal answer set which is to retrieve a first set of documents.
- An interaction with the user is then initiated with the purpose of improving the probabilistic description o ideal answer set.
- Such user interaction is performed as follow :
  1. The user takes a look at the retrieved documents and decides which ones are relevant and which are not relev
  2. The system then uses this information to define the description of the ideal answer set.
  3. By replacing this process many times such a description will evolve and become closer to the real descri of the ideal answer set.
- The probability model is based on following probabilistic principle.
- Given a user query q and a document d_j in the collection.
  1. The probabilistic model tries to estimate the probability that the user will find the document d_j relevant.
  2. It assumes that this probability of relevance depends on the query and the document representations.
  3. The model assumes that these are a subset of all documents which the user prefers at the answer set fo query q.
- Given a query q, the probabilistic model assigns to each document d_j, as a measure of its similarity to the q the ratio $p(d_j \text{ relevant to } q) \mid p(d_j \text{ non relevant to } q)$ which computes the odds of the document d_j being relev the query q.

- The similarity sim (d_j, q) of the document d_j to the query q is defined as,

$$sim(d_j, q) = \frac{p\left(R \mid \vec{d_j}\right)}{p\left(\bar{R} \mid \vec{d_j}\right)}$$

Where,

$\omega_{i,j} \in \{0, 1\}, \omega_{i,q} \in \{0, 1\}$	:	Index term weight variables are all binary
R	:	Set of documents known to be relevant
$\bar{R}$	:	Complement of R, set of documents non relevant
$P(R \mid \vec{d_j})$	:	Probability that document d_j is relevant to the query q
$P(\bar{R} \mid \vec{d_j})$	:	Probability that document d_j is non-relevant to the query q

Advantages of Probabilistic Model

Documents are ranked in decreasing order of their probability of being relevant.

Disadvantages of Probabilistic Model

1. We have to guess the initial separation of documents into relevant and non-relevant sets.
2. The method does not take into account the frequency with which an index term occurs inside a document.
3. The adoption of the independence assumption for index terms.

2.3 Alternet Set Models

2.3.1 Alternet Set Theoretic Models

A. Fuzzy Set Retrieval

- Fuzzy representation model is very useful for better interpretation of information.
- In normal Boolean logic there can only be two levels, true or false.
- But Fuzzy logic can express same thing in multiple levels.
- Example, in Boolean logic temperature can be hot or cold.
- But in fuzzy logic, temperature can be expressed as very cold, cold, warm, hot and very hot.
- So, same information is presented with much better details for better interpretation.
- In information retrieval this helps by representing the query in better manner.
- Here first information is converted in fuzzy form using fuzzification operation.
- Each level is some degree of membership to different classes.

For example, temperature between warm and cold, belongs to warm class as well as cold class.

- Such temperature is partially warm and partially cold.
- Such fuzzified information can be used for various pattern matching and retrieval of detailed information.
- When information is retrieved in form of fuzzy levels, then it needs to be converted back to original form.
- Such process is known as de-fuzzification.
- In Neural Networks, information can be fuzzified and used.
- Neural networks can be used to defuzzify the information.
- Adaptive Neuron Fuzzy Inference Systems (ANFIS) are popular for many applications of neural networks.

Fuzzy membership functions

- Membership functions are used for fuzzification any information
- There are several types of fuzzy membership functions.
- Here most popular ones are discussed.

1. **Trapezoidal**
   - Here, membership function forms a trapezoidal shape for each class.
   - So, highest value of membership is achieved between lower and upper limit.
   - Beyond these limits slowly membership reduces to zero.
   - Because highest membership is between limits, this is most useful function.
2. **Triangular**
   - Here membership function forms a triangular shape for each class.
   - Only a single value will receive full membership.
   - On both sides of highest peak, membership values slowly reduces to zero.
   - This function is useful after trapezoidal function.
3. **Gaussian**
   - Here, membership function forms a Guassian curve or Bell Curve.
   - Here highest membership is given exactly at the center.
   - On both sides of center, membership value reduces slowly to zero.
   - This is standard membership function.

B. Extended Boolean Retrieval

- Standard Boolean model has some drawbacks.
- Exact matching may lead to too few or too many document matches.
- So, extended model goes for a partial match.
- Here weights of terms are normalized per document.
- First product of each Term's Frequency (TF) and Inverted Document Frequency (IDF) is calculated
- Then, all term's Inverted Document Frequency (IDF) is normalized by maximum IDF value

$$W_{term} = (tf * idf) / max(idf)$$

- For all practical purposes, Boolean model is efficient than extended Boolean model.

2.3.2 Alternet Algebraic Model

Generalized Vector Model (Generalized Vector Space Model) GVSM

- It is generalization of VSM (Vector Space Model)
- In VSM, every word or term is assumed to be independent of each other
- This assumption in real life scenarios may not be valid
- So, GVSM was created with assumption that words / terms may not be independent of each other
- Here each word or term can be assumed to have relation with other words or terms

Word Similarity / Term Similarity in GVSM

- It is calculated based on existing dictionary / wordnet / sentinet
- Similarity is based on steps required to reach the word and number of terms related to given term in dictionary / wordnet / sentinet

Latent Semantic Indexing (LSI)

2. Explain Latent Semantic Indexing with a suitable example.

- It is also known as Latent Semantic Analysis (LSA).
- It is used to find latent or hidden pattern in a document.
- Here analysis is based on Singular Value Decomposition (SVD).
- SVD gives importance of each document.
- Here in LSI, linearly independent documents are found out, also known as topics.
- Similar documents are placed closer to each other in new latent space.
- Various documents can be compared based on cosine similarity.
- Cosine similarity is discussed in previous section of vector space model.
- Here term document matrix is used as base.
- Term document matrix can be Vector Space or Boolean or any other form.

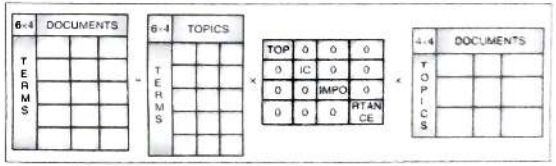


Fig. 2.3.1

- It is one of the most popular models for finding patterns in document.
- One of the drawbacks of this model is that it is time taking.
- Calculating SVD for very large number of document set is computationally heavy.
- LSI improves recall but may not help precision.
- After decomposition results cannot be interpreted easily.
- Also, selection of independent documents has influence on overall performance of LSI.

2.3.3 Neural Networks

- Neural Networks can be used for ranking search results
- Here neural network learns raw features from various texts.
- Neural Network is composed of artificial neurons with multiple layers.
- Neural network can be simple or Deep Neural Network.
- Artificial neuron is simulated version of biological neurons in human brain.
- Every artificial neuron has inputs, associated weights, and activation function.
- Multiple inputs are taken in, and then every input is multiplied with respective weight.
- Then all these products are summed up to form the complete input.
- This complete input is given to activation function.
- Activation functions can be linear or non linear.
- Most popular activation functions are Sigmoid, and Rectified Linear Unit (ReLU).
- All neurons from one layer have same activation function.
- Such multiple layers are stacked up together to form a complex neural network.
- Final layer of neural network is output layer.
- First layer of neural network is input layer.
- After output is given, error is calculated with respect to the idea output.
- This error is back propagated, known as Neural Network with back propagation.

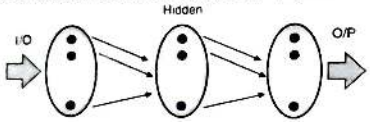


Fig. 2.3.2

- In deep neural network, layers of various neurons can be together like convolution, Recurrent Neurons, and Short Term Memory Neurons
- Also, there can be advanced architectures like reservoir computing or BERT, etc.
- Neural networks can be applied as stacked networks like auto encoders or viable encoders

2.3.4 Difference between Neural Network-based Retrieval and Fuzzy Set Retrieval Method

Q. Compare Neural network-based retrieval and Fuzzy set retneval methods.

Table 2.3.1

Sr. No.	Neural Network based Retrieval	Fuzzy Set Retrieval
1	Interactive activation model of word perception	Fuzzy document representation based on Boolean mod

Sr. No.	Neural Network based Retrieval	Fuzzy Set Retrieval
2.	Query is represented in form of tags	Query is considered as weighted words or terms
3.	Documents are assumed to be connected to each other in a dependency graph format.	Documents are considered as independent of each other.
4.	Document can belong to multiple classes but membership to each class is 0 or 1	Each document may have memberships to many classes and that membership can be of any value between 0 to 1.
5.	May not model vagueness in queries	Best models vagueness of natural language queries.

.3.5 Alternet Probabilistic Models

.3.5(A) Inference Networks

An inference network is a directed acyclic graph. In this graph nodes represent propositional variables or constants and edges represent dependence relationship between the propositions.

tructure of the Inference Network

The inference network used for information retrieval consists of two network components:

1. Document network
2. Query network

Fig. 2.3.3 shows the example of inference network.

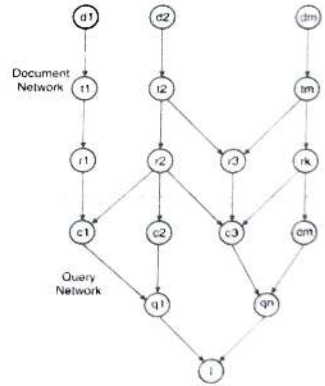


Fig. 2.3.3 : Basic document inference network

1. **Document network** : The document network consists of
   i. **Document nodes** : A document node represents a specific document in the collection. (eg. Monogra journal article, office document)
   ii. **Text representation nodes** : text node corresponds to a specific text representation of a document (eg. t images, figures, audio or video information)
   iii. **Concept representation nodes** : It represents the concept represented by the words present in the text Fig. 2 shows the example of inference network in which the document network consists of (d1, d2...dm) docume (t1, t2...tm) text representation nodes, and (r1, r2...rk) concept representation nodes. The edges between no represents that a document contain specific text which contains words having particular concepts.
2. **Query network** : Query network consists of the nodes for user queries. The types of nodes are:
   1. **Information need** : Each query network consists of an information need node which is the root of network.
   2. **Query nodes** : A Query node represents the information needs in query form. A network may contain on many query nodes because based on user's information needs the queries can be represented in different fo Also, during the search the queries can be refined and new query nodes can be generated in the network.
   3. **Concept nodes** : A single query has multiple concept nodes. A concept node can be linked to more than query. For example: the concept node "Information" will be linked to query "Information" and also with query "Information retrieval".

- The dependencies and relations between Information need, query and concept nodes are represente edges in the network.

Probability calculation of nodes :

- Inference network is the probability based network. Once the graph is defined the probability of each nod computed based on the probability that a user will search for a specific query and the specific document may retrieved as the answer to the query. Each document node has a prior probability associated with it that desc the probability of observing that document; this prior probability will generally be set to 1/(collection size). this probability value is reversely proportional to the document collection size. For all remaining vertices in network, the probability is computed based on its parents. For example, if node 'a' has a 3 parents with probabilities (p1, p2, p3), then the conditional probability of node is computed as P(a|p1.p2.p3).
- For the query network nodes, the probabilities are assigned based on the form of query. For example, if network is designed to handle the Boolean queries with the operators 'and', 'or', 'not' then the rules are define compute the probabilities of the nodes based on the probability of "information need" node and the operat present in the query. In the literature work, there is various logic introduced to assign these probabilities.
- In the network of Fig. 2.3.4, the values of the root nodes are fixed during instantiation, but the conditi probabilities at the remaining nodes must be estimated. Estimates are required at representation and q concept nodes and at the information need. The estimates used in the query network are either fixed by operator type (Boolean or sum) or require estimation of the relative contribution of the parent nodes (weig sum).

- When a weighted sum is used at a query node, the contribution of the parent concepts is based on the frequency of the parent concept in the query. A sum estimate is used most often for the information need, although a weighted sum can be used if we have externally supplied information about the importance or quality of individual queries.

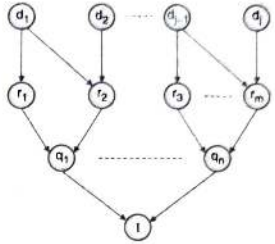


Fig. 2.3.4 : Simplified inference network

- The estimates at the representation concept nodes make use of a weighted sum, but setting the parent weights is somewhat more involved. Concept depends upon two factors: the frequency with which the concept occurs in an instantiated document and the 'surprise' associated with observing that concept assigned to a randomly selected document. These two factors, when combined, provide a good estimate of the term weight.

If the probability that concept *t* is assigned to a randomly selected document is

Pt = number of documents containing *t* /number of documents in the collection

- The surprise associated with observing *I* in a randomly selected document is

st = –log pt.

This surprise factor is equivalent to the familiar inverse document frequency *(idf)* measure used in the vector space model.

Fig. 2.3.5 shows the example of inference network for the query "(wireless and network) or not computer"

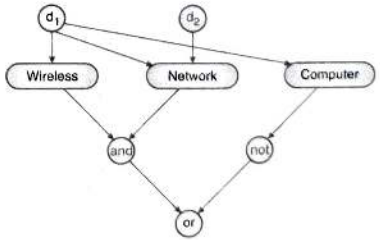


Fig. 2.3.5 : Inference network for (wireless ∧ network) ∨ ¬ computer

2.3.5(B) Belief Networks (Bayesian Belief Networks)

- It captures known conditional dependency between a random variables (inputs) and dependent variable (output)
- They are tool to visualize the probabilistic model for given domain or data
- Challenge in modelling probabilistic model is finding probabilities of all possible combinations
- This challenge is addressed by using some simple assumptions
- So, here combinations are found by using the dataset rather than all possible combinations
- It uses Directed Acyclic Graph (DAG)
- Here join probability of the event represented in graph is calculated
- Joint probability of at a point (node) dependents on parent node in the graph

Applications :

1. Prediction problems like disease diagnosis
2. Forecasting problems like rain forecasting
3. Social Network Analysis

2.4 Structured Text Retrieval

- Text data is sequence of words with very minimal structure.
- For retrieval process there needs to be defined structure.
- Also there needs to be query language or structure to perform search and retrieval in given text data.
- So, XML (eXtensibleMarkup Language) like format helps to define flexible structure
- Structured text retrieval is also known as XML (eXtensibleMarkup Language) based retrieval
- There are databases which are based on XML like XDMA database.
- Following are three important points related to retrieval of structured text (XML)
- Structure
  - XML file structure is used
  - XML has self-describing structure
  - XML has used defined tags and values of the tags or sub tags
  - XML has flexible structure
  - It enables data to be stored in strict structure or even sem-structure format
- Query Language
  - In XML queries are based on keywords and documents
  - Query needs to have structure hints to match with the structure
  - Traditionally XQuery is used as query language in such data

- Ranking or Matching Documents
  - Similarity in documents can be based on structure or keywords
  - Similarity between query and documents decide the rank of the document to be retrieved
  - There can be exact match of contents or regular expression based matches

2.5 Models for Browsing

- Browsing text data is important task because there is lot of text data now days
- There are three major ways to browse the text data
  - Flat files
  - Structure Guided
  - Hyper Text Data
- Flat Files
  - In flat files text is present in simplest format
  - Text is unorganized
  - Browsing is generally sequential
  - There can be direct search options based on word to word matching
  - Ex: txt files, notepad files
- Structure Guided Files
  - Here files have structure
  - There can be indexes or tags in the file
  - Index or tag can guide the user to browse specific contents in file
  - Every section or part of file is mentioned in the index. (Ex. PDF with index)
  - In tag based file, part of file is present under that tag.(Ex. XML)
  - Files also can have inverted index for better structure guide based on words
- Hyper Text Files
  - Hyper Text Markup Language (HTML) based files come under this
  - Here, links can help user browse the data
  - Here structure can be very complex and can be cyclic
  - One part of text may lead to many other documents
  - Also many documents can have link to one part of the text
  - Ex. Every website on internet has Hyper Text or HTML structure

Review Questions

Q. 1 Explain Boolean model in detail.

Q. 2 Explain vector model with example.

Q. 3 Explain various measures of association.

Q. 4 What is Inference Network?

Q. 5 What is Neural Network?

3

UNIT III

Query and Operations in Information Retrieval

Syllabus

Query structures, Keyboard based querying, Pattern matching, structured queries, User relevance feedback, Automatic local analysis, Automatic global analysis

3.1 Query Languages

- In this section we will discuss different types of queries normally used for text retrieval systems. Different types of queries are be entered by the user. Few languages needs to rank the documents as the output of the query.
- Keyword based queries are such queries in which a single word or a collection of words are given as the query.
- In most of the query languages, the query is mentioned in terms of content and the structure if the text.
- In natural language queries, users specify the query in terms of sentences. In this case, stopwords need to be removed from the query.
- Let us discuss few important queries in next sections.

3.2 Keyword Based Querying

User may give single word query or possible combination of multiple word queries. Keyword based queries are simple and ranking is easier in such queries.

3.2.1 Single Word Queries

- These are the simplest queries. The documents are considered as the collection of words.
- In terms of Boolean model, the single word present in the query is searched in the documents. The documents which contain the word are listed as the relevant documents.
- In terms of vector model and probabilistic model, the similarity value is computed between the query and the document.
- Some models treat the keyword in the query as the "collection of letters". The words in the documents are treated as the collection of letters which are separated by *separating* letters. In this case, the match between the query and the document happens at the letters level.
- In vector model, the term frequencies are considered while ranking the documents. Term frequency is the count that the word (term) is present in the document. As the count of terms more, so is the relevance.

3.2.2 Context Queries

- In terms of context queries, user enters the keyword. During the search, the system searches the documents w contains the keyword and also the synonym of the keyword. Here, the context of the keyword has given n importance.
- There are 2 types of context queries :
  - **Phrase :** Phrase is the sequence of single-word queries. For example, if the query is,

    "Information retrieval" or

    "Information in the retrieval ..."

    "Information which is in the retrieval..."

    For above all sentences, the system consider the query Phrase as "Information retrieval". Even though words "Information" and "retrieval" get separated by in between stop words, the query phrase for all three sentences is "Information retrieval".
  - **Proximity** : A more relaxed situation of phrase query is the proximity in which the sequence of keywor given along with maximum possible count of words which can appear in between these two keywords example, the two keywords must be present in any 6 continuous words.
- Phrase query and proximity queries can be ranked and most relevant documents are returned to the user.

3.2.3 Boolean Queries

- Boolean queries are the combination of keywords and the operators.
- For example :

  Information AND retrieval
- In this query, IR system searches for the documents which contain both the keywords.
  - Query syntax tree is generated for the Boolean query.
  - For example, Consider the query,
  - Translation AND (syntax OR semantic)

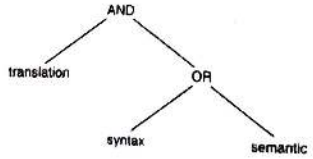


Fig. 3.2.1 : Query syntax tree for the query "Translation AND (syntax OR semantic)"

- Fig. 3.2.1 shows the query syntax tree for the example.
- The possible operators in Boolean query are :
  - **OR** : The query (K_1 OR K_2) selects all the documents which contain either K_1 or K_2. Duplications are elimin

  - **AND** : The query (K_1 AND K_2) selects all the documents which contain both K_1 and K_2. Duplications are eliminated.
  - **BUT** : The query (K_1 BUT K_2) selects all the documents which contain K_1 but not K_2.
- In terms of boolean query, mostly the exact match found. The IR system will not rank the documents. The documents which satisfy the conditions of keyword and boolean operators are listed as the result.

3.2.4 Natural Language

A fuzzy boolean model is used in such queries where the boundaries of the boolean operators are blurred and the query becomes an enumeration of words or context queries. The ranking is applied where the documents which contains more part of the query are listed as more relevant.

3.3 Pattern Matching

A pattern is a set of syntactic features that must occur in a text segment. The text parts which satisfies such syntactic patterns are considered as the match. In pattern matching queries, IR system finds the documents which has such segment of patterns in the text. Following are the possible patterns:

Word : It is a string i.e. the sequence of characters that must be present in the text.

Prefixes : It is as string which must be present at the start of the word. For example, if the prefix is "ters", then all documents containing the words "computers", "tensers", "painters", etc are relevant to the query.

Substrings : A string which can appear in the word. For example, if the substring is "tal". Then IR system searches for the documents which contain the words having the substring "tal". The words can be "talk", "postal", "metallic", etc.

Ranges : A pair of strings are given as the range. All words which are in this range of strings (as per lexicographic order) are searched in the documents. The documents which contains any of the words in this range are considered as the relevant documents.

Allowing errors : A word with the allowing error threshold. During the search of the words, here we declare two words match even though they are far from each other by difference of some characters. For examples consider the query word as "flower" and the word in the document as "flower". In this case, these two words differ by a single error. Thus in such queries, the maximum possible error threshold is defined in terms of characters. The words which fall in this error threshold are considered as match. The document which contains query word or the words which are near to the query word having error less than threshold, are considered as the relevant documents.

Regular expressions : A regular expression is the collection of strings along with the following operators:

- **Union** : if e_1 and e_2 are regular expressions, then $(e_1|e_2)$ is the regular expression in which strings are matched if they follow either e_1 or e_2 pattern
- **Concatenation :** if e_1 and e_2 are regular expressions, then $(e_1.e_2)$ is the regular expression in which strings satisfy the condition of e_1 followed by e_2.
- **Repetition :** if e_1 is the regular expressions, then (e^*) is the regular expression in which strings are matched if they follow zero or more occurrences of the pattern e.

- **Extended patterns** : Extended patterns are simplified form of regular expression. The user can give the qu extended pattern where the IR system converts the input internally in terms of regular expression. There are l variants of extended patterns, few examples are like :
  - **Classes of characters** i.e. one or more positions in the pattern satisfies the condition, or case-sensiti insensitive patterns. Complement, enumeration, etc.
  - **Conditional expression** : These expressions defines the condition of part of the patterns.
  - **Wild characters** in which the pattern is defined for the words. For examples words which starts with "tra ends with "s". The possible words are "transactions", "trails"
  - **Combinations that allow** some part of the patterns matches the condition and remaining part is accep with errors

3.4 Structured Queries

- Till yet we have discussed the queries which defines the conditions on text only. But when we save the docum in the collection, documents follows the structures. In such cases, users can define the query in which he mention the conditions of the text structure as well. The query language which gives facility to define the q over text as well as structure, becomes effective language.
- Fig. 3.4.1 shows main structures of the text :

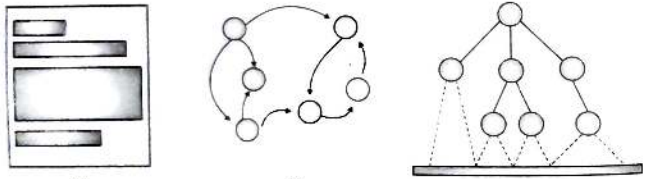


Fig. 3.4.1 : Three main structures: (a) Form-like fixed structure (b) hypertext structure (c) hierarchical structu

3.4.1 Fixed Structure

- This is the traditional way to save the text document in which the structure contains fixed fields. Each field has text. For example, the fields can be category of the document, date, author, location, etc. When the document saved in the database, these fields need to be filled for each document.
- But in practical situation, this structure may face some problems such as all fields of the document are not few documents cannot be categorized in any class which is predefined.
- While defining the query for the documents, nested or overlapped queries are not allowed. The queries he the basic queries in which user specifies that whether a particular value is present in any specific field.
- As the structure is fixed, for each field we can define its type. For example date fields is of date type, doc number is of type number, the actual text is of type text, etc. As the pattern is fixed, the document can be using relational model such as SQL.

3.4.2 Hypertext

Hypertext is the directed graph in which the nodes represents the text and the links represents the connection between the nodes. Hypertext structure become more popular because of web. Here the nodes are the web pages are the links helps to make connection between these web pages. User can follow the link and move from one page to another while searching his interest. Web is the very big network in which lots of web pages are linked with each other with the help of links.

The browsing and searching task can be combined with the help of WebGlimpse. It allows the navigation as well as provides the facility to search by contents in the neighborhood of the current node. Number of tools are available now which simplifies the users burden to search the contents using these hyperlinks.

3.4.3 Hierarchical Structure

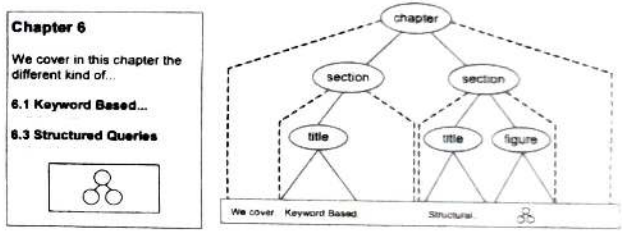


Fig. 3.4.2 : Hierachical structure

The hierarchical structure lies in between fixed structure and hypertext. Fig. 3.4.3 shows the example of the book structure in which the chapters and its sections are arranged in hierarchical manner.

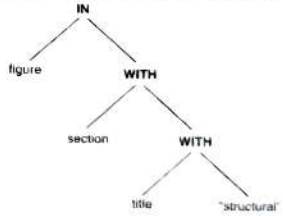


Fig. 3.4.3 : Example of query for hierarchical structure

Fig. 3.4.2 shows the example of query for the hierarchical structure. Its meaning is find the chapter in which the title is "structural", which is present in specific "section" along with the specific "figure".

3.5 User Relevance Feedback, Automatic Local Analysis and Automatic Global Anal

Introduction

When the naïve users try to formulate the query for IR system, users are not familiar and face difficulty framing the queries. Most of the times user needs to reform the queries to get the correct result. Thus the first formulation are considered as the initial phase of the query formulation. The documents retrieved as the respon such queries are analyzed based on which the queries get refined by the users.

Query modification comprises of:

- **Relevance feedback : It is the user feedback given to IR system about the** relevant documents to a query
- **Query expansion : when information** related to the query is used to expand it.

3.5.1 User Relevance Feedback

- Relevance feedback is the most popular query reformulation technique. In this technique, user provides qu the IR system. IR system lists the documents which are relevant. User does observation of these document marks them as relevant or not relevant. Mostly, first 10 to 20 ranked documents are judged and this feedb given to IR system to refine the results. The main idea in the feedback is selecting important terms or expre related to the documents and refining the results.
- User relevance method technique has more advantage over query expansion as,
  - User do not have burden to refine the query. He has to just give the judgment about the results.
  - The searching process breaks down in smaller steps.
  - The process is controlled and refinement in the query proceeds to increase accuracy in the result.

3.5.2 Query Expansion and Term Reweighting for the Vector Model

The application of relevance feedback considers that the documents which are relevant to the query has simi among themselves. Also, there is less similarity between the pairs of relevant and non-relevant documents.

3.5.2(A) The Rocchio Method

- Consider following parameters :
  - C_r : Set of relevant documents among the collection.
  - $|C_r|$: The number of documents present in set C_r
  - D_r : IR system retrieves the documents as per the query. Dr is the set of those documents from re documents which user identified by user as really relevant.
  - $|D_r|$: The number of documents present in set D_r
  - D_n : set of non-relevant documents among the retrieved documents.
  - $|D_n|$: The number of documents present in set D_n
  - α, β, γ : tuning constants

Assume that the set C_r is already known. Then, the best query vector for distinguishing the relevant from the non-relevant docs is given by,

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

Where,

$\vec{d}_j$: a weighted vector associated with document d_j

$\vec{q}_{opt}$: Optimal weighted vector for query q

Above equation is true when the set C_r is already known. But, initially we do not know this set. Hence, we can formulate the initial query and go on refining it and update the initial vector.

Consider,

$\vec{q}_m$: It is the modified query

There are three classic and similar ways to calculate the modified query $\vec{q}_m$ as follows.

$$\text{Standard\_Rocchio} : \vec{q}_m = \alpha \vec{q} + \frac{\beta}{N_r} \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{N_n} \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$\text{Ide\_Regular} : \vec{q}_m = \alpha \vec{q} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$\text{Ide\_Dec\_Hi} : \vec{q}_m = \alpha \vec{q} + \beta \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \max\_rank(D_n)$$

Where, $max\_rank(D_n)$ is the highest ranked Non-relevant doc

α, β, γ are the tunable parameters. The values of these parameters are suggested by researchers.

$\alpha = 1$: proposed by Rocchio

$\alpha = \beta = \gamma = 1$: proposed by Ide

$\gamma = 0$: which yields a positive feedback strategy

The experimentations with these parameters shows that we get similar results, by using any of these techniques.

dvantages of Rocchio Method

- **Simplicity** : Modified term weights are computed directly from the set of retrieved documents
- **Good results** : The modified query vector does reflect a portion of the intended query semantics similar results.

.5.2(B) Term Reweighting for the Probabilistic Model

The probabilistic model ranks the results based on the probability values of the documents with respect to the query.

- Consider the parameters
  - $P(k_i \mid R)$: Probability of observing the term k_i in the set R of relevant documents
  - $P(k_i \mid \bar{R})$: Probability of observing the term k_i in the set R of non-relevant docs
- The similarity value between the document and the query is represented as,

$$sim(d_j, q) \alpha \sum_{k_i \in q \wedge k_i \in d_j} \left(\log \frac{P(k_i \mid R)}{1 - P(k_i \mid R)} + \log \frac{1 - P(k_i \mid \bar{R})}{P(k_i \mid \bar{R})} \right)$$

- **In the initial phase**, the above equation cannot be used as we do not know the values of $P(k_i \mid R)$ and $P(k_i \mid \bar{R})$
- For probabilistic model, there are different methods to define initial values of initial values of $P(k_i \mid R)$ and $P(k_i \mid \bar{R})$
  For the user feedback information, the initial values are set as,
  $P(k_i \mid R)$ is constant for all terms k_i (typically 0.5)
- The term probability distribution $P(k_i \mid \bar{R})$ can be approximated by the distribution in the whole collection,
- Hence, the initial values are set as,

$$P(k_i \mid R) = 0.5 \quad P(k_i \mid \bar{R}) = \frac{n_i}{N}$$

Where,

- N : number of documents in the collection
- n_i : Number of documents in the collection that contain the term k_i
- $n_{r,i}$: Number of documents in set D, that contain the term k_i

- Thus, the similarity equation becomes,

$$sim_{initial}(d_j, q) = \sum_{k_i \in q \wedge k_i \in d_j} \log \frac{N - n_i}{n_i}$$

- Feedback of the user is used to update the values of $P(k_i \mid R)$ and $P(k_i \mid \bar{R})$ as,

$$P(k_i \mid R) = \frac{n_{r,i}}{N_r} \qquad P(k_i \mid \bar{R}) = \frac{n_i - n_{r,i}}{N - N_r}$$

- And the similarity function is updated as,

$$sim(d_j, q) = \sum_{k_i \in q \wedge k_i \in d_j} \left(\log \frac{n_{r,i}}{N_r - n_{r,i}} + \log \frac{N - N_r - (n_i - n_{r,i})}{n_i - n_{r,i}} \right)$$

- In the case of above similarity equations, the query expansion does not occur. The same query te reweighted using feedback information provided by the user. But the equation may introduce problem fo values of N_r and $n_{r,i}$. Hence, 0.5 is normally added to $P(k_i \mid R)$ and $P(k_i \mid \bar{R})$

$$P(k_i \mid R) = \frac{n_{r,i} + 0.5}{N_r + 1} \qquad P(k_i \mid \bar{R}) = \frac{n_i - n_{r,i} + 0.5}{N - N_r + 1}$$

- This adjustment factor of 0.5 may provide unsatisfactory results in some cases. Hence, other value suggest adjustment is n_i/N.

Advantage of probabilistic model

The main advantage of this feedback method is the derivation of new weights for the query terms.

Disadvantage of probabilistic model

Document term weights are not taken into account during the feedback loop.

Weights of terms in the previous query formulations are disregarded

No query expansion is used (the same set of index terms in the original query is reweighted over and over again)

This probabilistic method is not in general operate as effectively as the vector modification methods.

3.6 Automatic Local Analysis

Clustering for query expansion is the basic technique for IR system. In terms of user feedback, the relevant documents are clustered together and the clusters are updated using the user's feedback. The cluster of relevant documents get expanded when there is refinement in the query based on user feedback.

Another approach to obtain the description of the cluster containing relevant documents automatically. This process consists of identifying terms which are related to query terms. Such terms can be synonyms, stemming variants or terms which are close to the query terms. Two basic strategies are used in this case, viz, local and global.

In global strategy, all documents in the collection are considered and a global thesaurus-like structure is created. This structure defines the term relationships. This structure is showed to user and user selects the terms from the structure for query expansion.

In local strategy, the documents which are relevant to the query q are considered. The terms present in these documents are considered for query expansion. It is similar to the user feedback process but happen automatically, without the help of user. Two local strategies are discussed below

3.6.1 Query Expansion through Local Clustering

Consider the parameters :

- V(s) : Non-empty subset of words which are grammatically variants of each other. A canonical form of V(s) is called stem.
- For example, if V(s) = {polish, polishing, polished} then s = polish

For a given query q, let

- D_l : **local document set,** i.e., set of documents retrieved by q
- N_l : number of documents in D_l
- V_l : local vocabulary, i.e., set of all distinct words in D_l
- $f_{i,j}$: frequency of occurrence of a term k_i in a document $d_j \in D_l$
- $M_l = [m_{ij}]$: term-document matrix with V_l rows and N_l columns
- $m_{ij} = f_{i,j}$: an element of matrix M_l
- M_l^T : transpose of M_l

The matrix

$$C_l = M_l M_l^T$$

is a local term-term correlation matrix

- $C_{u,v}$ - C_l: the term which defines the correlation between k_u and k_v terms. The joint-coorelation between the k
- are depend on the joint co-occurrences of the terms in the documents which are part of collection. If the term
- co-related in large number of documents, ultimately, their correlation value is bigger. Thus, if one term is pr
- in query. Then its correlated term can also be used as the part of query expansion.
- There are 3 types of clusters : association clusters, metric clusters and scalar clusters.

1. Association Clusters

- An association cluster is computed from a local correlation matrix C_l. For that, we re-define the correl
- factors $c_{u,v}$ between any pair of terms k_u and k_v as follows :

$$C_{u,v} = \sum_{d_j \in D_l} f_{u,j} \times f_{v,j}$$

- In this case the correlation matrix is referred to as a local association matrix. The motivation is that terms
- co-occur frequently inside documents have a synonym association.
- The correlation factors $C_{u,v}$ and the association matrix C_l are said to be unnormalized.
- An alternative is to normalize the correlation factors :

$$C^l_{u,v} = \frac{C_{u,v}}{C_{u,u} + C_{v,v} - C_{u,v}}$$

- In this case the association matrix C_l is said to be normalized.
- Given a local association matrix C_l, we can use it to build local association clusters as follows :
  - $C_u(n)$: Function that returns the n largest factors $c_{u,v} \in C_l$, where v varies over the set of local terms $v \neq u$
  - Then, $C_u(n)$ defines a local association cluster, a neighborhood, around the term k_u
- Given a query q, we are normally interested in finding clusters only for the |q| query terms
- This means that such clusters can be computed efficiently at query time.

2. Metric Clusters

- Association clusters considers the occurrence of the terms in the documents. It does not consider the po
- at which the terms are present in the sentence.
- Metric terms considers the location/ position where the terms occurs in the sentences. Two terms occur
- same sentences tend to be correlated. A metric cluster re-defines the correlation factors $c_{u,v}$ as a funct
- their distances in documents.
- Let $k_u(n, j)$ be a function that returns the n^{th} occurrence of term k_u in document d_j.
- Further, let $r(k_u(n, j), k_v(m, j))$ be a function that computes the distance between
  - The n^{th} occurrence of term k_u in document d_j and j
  - The m^{th} occurrence of term k_v in document d_j

- We define,

$$C_{u,v} = \sum_{d_j \in D_l} \sum_{n} \sum_{m} \frac{1}{r(k_u(n, j), k_v(m, j))}$$

- In this case the correlation matrix is referred to as a local **metric matrix.**
- If k_u and k_v are in distinct documents we take their distance to be infinity.
- Variants of computing the value of $C_{u,v}$ is :

$$\frac{1}{r^2}(k_u(n, j), k_v(m, j))$$

- $C_{u,v}$ quantifies absolute inverse distances and said to be un-normalized and thus the local metric matrix C_l is un-normalized.
- The normalized correlation factor,

$$C^l_{u,v} = \frac{C_{u,v}}{\text{total number of } [k_u, k_v] \text{ pairs considered}}$$

- In this case, the C_l value is considered as normalized.

3. Scalar Clusters

- The co-relation between two terms can be defined based on their neighborhood as well. The logic behind this is the terms which are near to each other have some synonymity relationship. We can take help of the logic of siliarity between two scalar measures.
- Let
  - $\vec{S}_u = (C_{u,x_1}, S_{u,x_2}, \ldots, S_{u,x_n})$: vector of neighbourhood correlation values for the term k_u
  - $\vec{S}_v = (C_{v,y_1}, C_{v,y_2}, \ldots, C_{v,y_m})$: vector of neighbourhood correlation values for term k_v
- Define,

$$C_{u,v} = \frac{\vec{S}_u \cdot \vec{S}_v}{|\vec{S}_u| \times |\vec{S}_v|}$$

- In this case the correlation matrix C_l is referred to as a **local scalar matrix.**
- The local scalar matrix C_l is said to be induced by the neighbourhood.
- Let $C_u(n)$ be a function that returns the n largest $C_{u,v}$ values in a local scalar matrix $C_{l\, v \neq u}$
- Then, $C_u(n)$ defines a scalar cluster around term k_u

Neighbor terms

Consider the problem of expanding a given user query q with neighbour terms.

One possibility is to expand the query as follows :

- For each term $k_u \in q$, select m neighbour terms from the cluster $C_u(n)$ and add them to the query
- This can be expressed as follows :

$$q_m = q \cup (k_v | k_v \in C_u(n), k_u \in q)$$

- Hopefully, the additional neighbour term k_v will retrieve new relevant documents.
- The set $C_u(n)$ might be composed of terms obtained using correlation factors normalized and unnormalized.
- Query expansion is important because it tends to improve recall.
- However, the larger number of documents to rank also tends to lower precision.
- Thus, query expansion needs to be exercised with great care and fine tuned for the collection at hand.

3.6.2 Query Expansion through Local Context Analysis

- Local context analysis is an approach that combines global and local analysis.
- It is based on the use of noun groups, i.e., a singlenoun, two nouns, or three adjacent nouns in the text.
- Noun groups selected from the top ranked documentsare treated as document concepts
- However, instead of documents, passages are used for determining term co-occurrences. Passages are windows of fixed size.

Three step procedure for local context analysis

1. Retrieve the top n ranked passages using the original query
2. For each concept c in the passages compute the similarity sim(q, c) between the whole query q and the concept
3. The top m ranked concepts, according to sim(q, c), are added to the original query q

- A weight computed as $1 - 0.9 \times i/m$ is assigned to each concept c,

  Where,

  i : position of c in the concept ranking

  m : number of concepts to add to q
- The terms in the original query q might be stressed by assigning a weight equal to 2 to each of them.

Finding similarity between query q and the concept c

- The similarity sim(q, c) between each concept c and the original query q is computed as follows :

$$sim(q, c) = \prod_{k_i \in q} \left(\delta + \frac{\log (f(c,k_i) \times idf_c)}{\log n}\right)^{idf_i}$$

  Where, n is the number of top ranked passages considered.
- The function $f(c, k_i)$ quantifies the correlation between the concept c and the query term k_i and is given by.

$$f(c, k_i) = \sum_{j=1}^{n} pf_{i,j} \times pf_{c,j}$$

  Where,
  - $Pf_{i,j}$ is the frequency of term k_i in the j^{th} passage; and
  - $Pf_{c,j}$ is the frequency of the concept c in the j^{th} passage.
- Notice that this is the correlation measure defined for association clusters, but adapted for passages.

- The inverse document frequency factors are computed as,

$$idf_i = \max\left(1, \frac{\log_{10} / np_i}{5}\right)$$

$$idf_c = \max\left(1, \frac{\log_{10} N/np_c}{5}\right)$$

  Where,
  - N is the number of passages in the collection
  - np_i is the number of passages containing the term k_i; and
  - np_c is the number of passages containing the concept c
- The idf_i factor in the exponent is introduced to emphasize infrequent query terms.
- The procedure above for computing sim(q, c) is a non-trivial variant of tf-idf ranking.
- It has been adjusted for operation with TREC data and did not work so well with a different collection.
- Thus, it is important to have in mind that tuning might be required for operation with a different collection.

Automatic Global Analysis

In the local analysis method, the documents which are retrieved with respect to query are used to expand the query. In another approach, all the documents in the collection are used to expand the query and the approach is named as global analysis.

There are two variants of the global analysis :

1. Query expansion based on a similarity thesaurus
2. Query expansion based on a statistical thesaurus

Let us discuss each variant in detail in next sections.

3.7 Query Expansion Based on a Similarity Thesaurus

This model is based on global similarity thesaurus which is constructed automatically by the IR system.

The similarity thesaurus uses term to term relationships rather than on a matrix of co-occurrence. Terms are carefully selected and the reweighting of these terms is really interesting. Terms for expansion are selected based on their similarity to the whole query.

While defining the term thesaurus, the terms are considered as concepts in concept space. The terms are indexed by the documents in which they appear. Thus the terms plays important role and the documents are handled as the indexing elements.

Let,

- t : number of terms in the collection
- N : number of documents in the collection
- $f_{i,j}$: frequency of term k_i in document d_j
- t_j : number of distinct index terms in document d_j

- Then,

$$\text{It } f_j = \log \frac{t}{t_j}$$

Is the inverse term frequency for document d_j (analogous to inverse document frequency)

- Within this framework, with each term k_i is associated a vector $\vec{k}_i$ given by,

$$\vec{k}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,N})$$

- These weights are computed as follows

$$w_{i,j} = \frac{\left(0.5 + 0.5 \frac{f_{i,j}}{\max_i(f_{i,j})}\right) itf_j}{\sqrt{\sum_{i=1}^{N} \left(0.5 + 0.5 \frac{f_{i,j}}{\max_i(f_{i,j})}\right)^2 itf_j^2}}$$

where $\max_i(f_{i,j})$ computes the maximum of all $f_{i,j}$ factors for the i^{th} term

- The relationship between two terms k_u and k_v is computed as a correlation factor $c_{u,v}$ given by,

$$C_{u,v} = \vec{k}_u \cdot \vec{k}_v = \sum_{\forall d_j} w_{u,j} \times w_{v,j}$$

- The global similarity thesaurus is given by the scalar term-term matrix composed of correlation factors $c_{u,v}$
- This global similarity thesaurus has to be computed only once and can be updated incrementally.
- Once the global similarity is given, the query expansion is done by following 3 steps :

1. Represent the query in the same vector space used for representing the index terms :

   The query is represented by vector $\vec{q}$ and is computed as,

$$\vec{q} = \sum_{k_i \in q} w_{i,q} \vec{k}_i$$

2. Compute a similarity sim(q, k_v) between each term k_v correlated to the query terms and the whole query q.

   The similarity sim(q, k_v) is computed as,

$$sim(q, k_v) = \vec{q} \cdot \vec{k}_v = \sum_{k_i \in q} w_{i,q} \times C_{i,v}$$

- A term k_v might be closer to the whole query centroid qc than to the individual query terms.

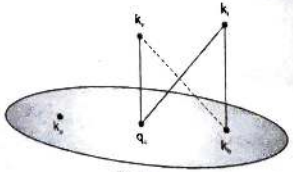


Fig. 3.7.1

- Thus, terms selected here might be distinct from those selected by previous global analysis methods.

3. Expand the query with the top r ranked terms according to sim(q, k_v):

   The top r ranked terms are added to the query q to form the expanded query q_m. To each expansion term kv in query qm is assigned a weight w_{v,q_m} given by,

$$W_{vq_m} = \frac{sim(q, k_v)}{\sum_{k_i \in q} w_{i,q}}$$

- The expanded query q_m is then used to retrieve new documents.
- Consider a document d_j which is represented in the term vector space by

$$\vec{d}_j = \sum_{k_i \in d_j} w_{i,j} \vec{k}_i$$

- Assume that the query q is expanded to include all the t index terms (properly weighted) in the collection
- Then, the similarity sim(q, d_j) between d_j and q can be computed in the term vector space by

$$sim(q, d_j) \propto \sum_{k_v \in d_j} \sum_{k_u \in q} W_{v,j} \times W_{u,q} \times C_{u,v}$$

- The above expression is analogous to the similarity formula in the generalized vector space model. Thus, the generalized vector space model can be interpreted as a query expansion technique
- The two main differences are :
  1. The weights are computed differently
  2. Only the top r ranked terms are used

3.8 Query Expansion based on a Statistical Thesaurus

- The global thesaurus is composed of classes that group correlated terms in the context of the whole collection.
- Such correlated terms can then be used to expand the original user query. In this method, documents are clustered in terms of classes.
- The low frequency terms in these documents are used to define thesaurus classes.
- The algorithm which produces small and tighter clusters is the complete link algorithm.

3.8.1 Complete Link Algorithm

1. Initially, place each document in a distinct cluster
2. Compute the similarity between all pairs of clusters
3. Determine the pair of clusters $[C_u, C_v]$ with the highest inter-cluster similarity
4. Merge the clusters C_u and C_v
5. Verify a stop criterion (if this criterion is not met then go back to step 2)
6. Return a hierarchy of clusters

- The algorithm keeps the clusters small and tight. The similarity between the two document pair is computed the cosine similarity between the document vectors, which is same like vector model.
- Clusters contains number of documents.
- The similarity between two clusters is defined as the minimum of the similarities between two documents the same cluster.
- It helps in keeping the clusters small and tight.

Example :

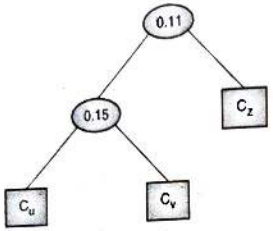


Fig. 3.8.1 : Clusters and similarity among clusters

- Fig. 3.8.1 shows the hierarchical structure of the clusters. The similarity between clusters C_u and C_v is 0.15 similarity between clusters C_{u+v} and C_z is 0.11.
- Similarity decreases when we move from bottom towards top in this hierarchy because high level clusters co more documents and thus represents looser grouping. Thus, the tighter clusters lies at the bottom of the hie
- The terms that compose each class of the global thesaurus are selected as follows :
- Obtain from the user three parameters :
  - TC : threshold class
  - NDC : number of documents in a class
  - MIDF : minimum inverse document frequency
- Parameter TC determines the document clusters that will be used to generate thesaurus classes
  - Two clusters C_u and C_v are selected, when TC is surpassed by $sim(C_u, C_v)$
- Use NDC as a limit on the number of documents of the clusters
  - For instance, if both C_{u+v} and C_{u+v+s} are selected then the parameter NDC might be used to d between the two
- MIDF defines the minimum value of IDF for any term which is selected to participate in a thesaurus class
- Given that the thesaurus classes have been built, they can be used for query expansion
- For this, an average term weight wt_C for each thesaurus class C computed as follows

$$wt_C = \frac{\sum_{i=1}^{|C|} w_{i,C}}{|C|}$$

Where

- $|C|$ is the number of terms in the thesaurus class C, and
- $W_{i,C}$ is a weight associated with the term-class pair $[k_i, C]$

This average term weight can then be used to compute a thesaurus class weight w_C as,

$$w_C = \frac{wt_C}{|C|} \times 0.5$$

The above weight formulations have been verified through experimentation and have yielded good results.

Review Questions

1. Give examples of context queries.
2. What is the structure of Boolean queries?
3. Which are the possible operators for Boolean queries?
4. Elaborate abbot the different patterns to specify the query for Information retrieval system.
5. What is structure query? Explain different types of structured queries.
6. Explain the Rocchio method for query expansion.
7. How the Term Reweighting takes place for the Probabilistic Model.
8. Differentiate between automatic local analysis and global analysis.
9. Write short note on: Association clusters
10. Elaborate about metric clusters used for query expansion in IR.

□□□

4 Indexing and Scoring in Information Systems

UNIT IV

Syllabus

Introduction, Inverted Files, Other Indices for Text, Boolean queries and Introduction to Sequential searching, Scoring, term weighting and the vector space model, Parametric and zone indexes, Weighted zone scoring, Learning weights, The optimal weight, Term frequency and weighting, Inverse document frequency, Tf-idf weighting, The vector space model for scoring, Queries as vectors, Computing vector scores, Efficient scoring and ranking, Inexact top k document retrieval

Introduction

One of the important characteristic of Information retrieval system is the efficiency. Efficient retrieval syste process the user queries within minimum computation time and needs minimum resources. The computation efficien and space efficiency is the key of success of any information retrieval system. IR system handles large amount of dat For example, in Web search engines the data size is in millions of bytes. The IR system must save this data in efficien fashion such that once the user gives the query, within fraction of seconds, the relevant information needs to be give to the user. The storage representation of information plays an important role during the search process of IR system.

Indexing is the solution to efficiently store the data. Index is a data structure built from the text to speed up th searches. In the context of an IR 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 the system handles the information which is in textual form, the index is built over the text. When there are changes or updates in the text, respective index entries must also be updated. In some cases, the collections are updated at a specific interval. For example, daily or monthly. Such collections are called as semi-static collections. The best example of semi-static collection are web pages. Although the Web changes very fast, the crawls of a search engine are relatively slow.

4.1 Inverted Files

4.1.1 Introduction

- In order to speed up the searching task, a word-oriented mechanism for indexing a text collection is required and that is called as inverted file or inverted index.

- The inverted file structure consists of two elements
  1. Vocabulary
  2. Occurrences
- The vocabulary is the set of all different words in the text.
- For each word a list of all the text positions where the word appears is stored. And the set of all those lists is called the occurrences.
- Example of inverted file is as shown in Fig. 4.1.1.
- In the Fig. 4.1.1, a sample index and an inverted index built on it is shown. The words are converted to lower case and some are not indexed. It also shows the occurrences point to character positions in the text.
- The positions can refer to words or characters.
  - Word positions (i.e. position i refers to the i^{th} word) is phrase and proximity queries.
  - Character positions (i.e. positions i is the i^{th} character) facilitate direct access to the matching text positions.
- The space required for the vocabulary is small and it grows as $O(n^{\beta})$ where β is constant in between 0 and 1, dependent on the text.

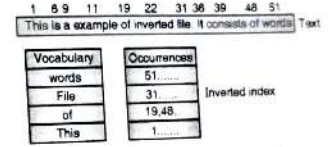


Fig. 4.1.1 : Sample text and inverted index

- The space required for the occurrences is more because each word appearing in the text is referenced once in that structure so extra space is O (n).
- To reduce space requirements, a technique called block addressing is used.
- In block addressing,
  1. The text is divided into blocks.
  2. The occurrences point to the blocks where the word appears.
  3. The classical indices which point to the exact occurrences are called full inverted indices.
- By using block addressing
  1. Pointers are smaller because there are fewer blocks than positions.
  2. All the occurrences of a word inside a single block are collapsed to one reference.
- This is shown in Fig. 4.1.2, the sample text is split into four blocks and an inverted index using block addressing built on it. The occurrences denote block numbers.

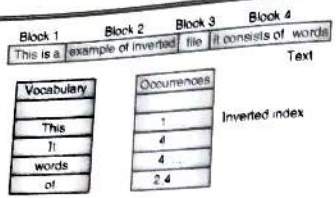


Fig. 4.1.2 : By using block addressing

- This block can be of fixed size or they can be defined using the natural division of the text collection into fi documents or web pages.
- The division into blocks of fixed size improves efficiency at retrieval time. If more variance in the block size then th more amount of text sequentially traversed on average. This is because larger blocks match queries mo frequently and are more expensive to traverse.
- The division using natural cuts may eliminate the need for online traversal.
- For example if one block per retrieval unit is used and the exact match positions are not required then there is n need to traverse the text for single word queries because it is sufficient to know which retrieval units to report. Bu if many retrieval units are packed into single block then the block has to be traversed to determine which units t retrieve.
- In order to use block addressing the text must be readily available at search time.

4.1.2 Searching

The search algorithm on an inverted index follows following three general steps.

1. **Vocabulary search**

   In this search, the words and patterns present in the query are isolated and searched in the vocabulary.

2. **Retrieval of occurrences**

   In this the lists of the occurrences of all the words found are retrieved.

3. **Manipulation of occurrences**

   - In this the occurrences are processed to solve phrases, proximity or Boolean operations.
   - If block addressing is used then we can search directly the text to find information mission from the occurrences.

- Searching on an inverted index always starts in the vocabulary. Because it is good to have it in separate file so that separate file can fits in main memory even for large collections.
- To speed up the search, Single-word queries can be searched using any suitable data structure like hashing, Tries or B-trees.
- But storing the words in lexicographical order is cheaper in space and very competitive in performance as the word can be binary searched at O (log n) cost.
- Prefix and range queries can also be solved with binary search, tries or B-tree but not with hashing.

- If the query is formed by single words, then the process ends by delivering the list of occurrences.
- It is more difficult to solve context queries with inverted indices.
- Each element must be searched separately and a list in increasing positional order must be generated for each one. Then the lists of all elements are traversed in synchronization to find places where all the words appear in sequence for a phrase or appear close enough for proximity. If one list is much shorter than the others then it is better to binary search its elements into the longer lists rather than performing a linear merge.
- If block addressing is used then it is necessary to traverse the blocks for these queries because the position information is needed.

  Then it is better to intersect the lists to obtain the blocks which contain all the searched words and then sequentially search the context query in those blocks.

4.1.3 Construction

- An inverted index on a text of a characters can be built in O(n) time.
- All the vocabulary is kept in a trie data structure and for each word a list of its occurrences are stored.
- Each word of the text is read and searched in the trie. If it is not found then it is added to the trie with an empty list of occurrences and the new position is added to the end of its list of occurrences.
- This is shown in Fig. 4.1.3.

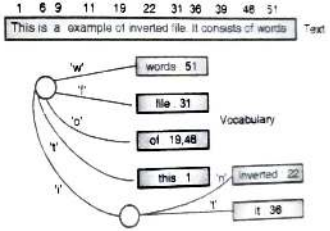


Fig. 4.1.3 : Building an inverted index

- Once the text is exhausted, the trie is written to disk together with the list of occurrences.
- The index is splitted into two files.
  1. In first file the lists of occurrences are stored contiguously and the file is called as 'posting file'.
  2. In the second file the vocabulary is stored in lexicographical order and for each word a pointer to its list in the first file is included.
- This allows the vocabulary to be kept in memory at search time in many cases and also the number of occurrences of a word can be immediately known from the vocabulary with less or no space overhead.
- Using this scheme, the overall process take O(n) time in worst case. In the trie O(1) operations are performed per text character and the positions can be inserted at the end of the lists of occurrences in O(1) time.
- But the above discussed scheme is not suitable for large texts where the index does not fit in the main memory and the paging mechanism used reduce the performance of the algorithm.

- So alternative to above scheme is discussed below.
- When no memory is available then the partial index I_i obtained is written to disk and erased from main memo
  before continuing with the test of the text.
- So, finally we will have a number of partial indices I_i exist on disk. These indices are then merged in hierarch
  manner.
- Indices I_1 and I_2 are merged to obtain I_{1-2}, I_3 and I_4 are merged to obtain I_{3-4} and so on. So in this way
  resulting partial indices are approximately twice the size.
- When all the indices at one level are merged then merging proceeds for the next level. Joining the index I_{1-2} wi
  the index I_{3-4} to form I_{1-4}. This is continued until there is only one index comprising the whole text.
- In the Fig. 4.1.4 rectangles are used to represent partial indices, rounded rectangles are used to represent merg
  operations. The numbers inside the margin operations show a possible merging order.

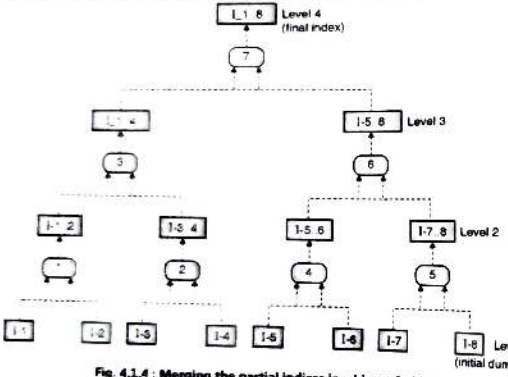


Fig. 4.1.4 : Merging the partial indices in a binary fashion

- Merging two indices consists of merging the sorted vocabularies and whenever the same word appears in bot
  indices then both lists of occurrences are merged.
- This approach is very fast and its complexity is O $(n_1 + n_2)$ where, n_1 and n_2 are the sizes of the indices.
- The total time required to generate the number of partial indices O (n / M) is O(n). Each level of merging perform
  a linear process over the whole index with the cost of O(n). To merge the O(n/M) partial indice
  $\log_2$ (n/M) merging levels are necessary. So total cost of the algorithm is O (n log (n/M)).
- More than two indices can be merged at once, this will not change the complexity but it improves the efficien
  because less merging levels exist.
- Also the memory buffers for each partial index to merge will be smaller and more disk seeks will be performed.
- In practice it is a good idea to merge 20 partial indices at once.
- We can perform in-place merging to reduce build-time space requirements. When two or more indices ar
  merged. Write the result in the same disk blocks of the original indices instead of on a new file.

- It is also possible to perform the hierarchical merging as soon as the files are generated. This reduces the space requirements because the vocabularies are merged and redundant words are eliminated.
- If block addressing is used, this algorithm changes very little. So with block addressing maintenance is cheap.
- If new text of size n' is added to the database, then building inverted index for the new text and merging both the indices takes O (n + n' log (n' | M)). Deleting text can be done by an O(n) pass over the index eliminating the occurrences that point inside eliminated text areas.

4.2 Other Indices for Text

Q. Explain suffix array with the help of diagram. **(4 Marks)**

Other indices for the text are suffix trees and suffix arrays, signature files, hash addressing and clustered files. Let us discuss about each index in detail.

Suffix Trees and Suffix Arrays

4.2.1 Introduction

- Inverted indices see the text as sequence of words and this restricts the kinds of queries than can be answered. Similarly other queries like phrases are expensive to solve. And the concept of words does not exist in some applications like genetic databases.
- Suffix arrays are space efficient implementation of suffix trees. Suffix arrays allow and answer more complex queries very efficiently.
- This structure can be used to index only words as the inverted index as well as to index any text character.
- Main drawback of this structure is construction process is very costly, the text must be readily available at query time and the results are not delivered in text position order.
- This data structure is suitable for a wider spectrum of applications such as generic databases. But for word-based applications, inverted files perform better.
- This index considers the text as one long string. Each position in the text is considered as a text suffix.
- Text suffix is a string that goes from that text position to the end of the text.
- Each suffix is uniquely identified by its position.
- There is no need to index all text positions so it is possible to index only word beginnings to have functionality similar to inverted indices.
- Index points are selected from the text, which points to the beginning of the text positions which will be retrievable and those elements which are not index points are not retrievable. Fig. 4.2.1 illustrates this.

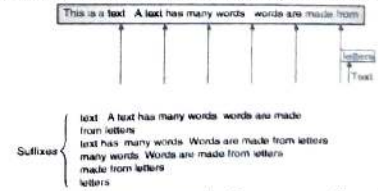
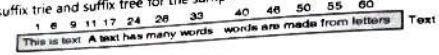


Fig. 4.2.1 : Sample text with index points and suffixes corresponding to index points

4.2.2 Structure of Suffix Tree

Q. Explain working of suffix tree. (3 Ma

- Suffix tree is a trie data structure which is built over all the suffixes of the text.
- The pointers to the suffixes are stored at the leaf nodes. For better space utilization, trie structure is comp into a patricia tree.
- It involves compressing unary paths i.e. paths where each node has just one child. An indication of th character position to consider is stored at the nodes which root a compressed path.
- If unary path are absent then the tree has O(n) nodes instead of the worst case $O(n^2)$ of the trie.
- Fig. 4.2.2 shows the suffix trie and suffix tree for the sample text.



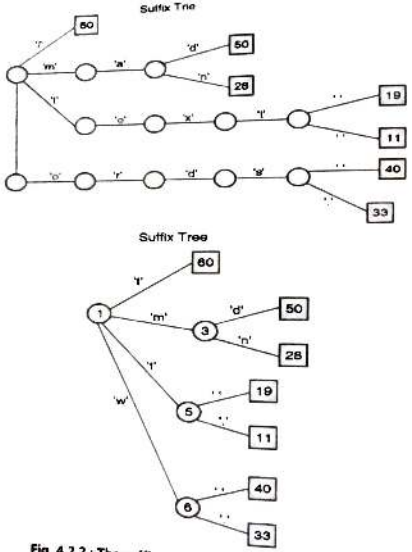


Fig. 4.2.2 : The suffix tree and suffix tree for sample text

- The problem with this structure is its space. Suffix arrays provide the same functionality as suffix trees wit space requirements.
- If the leaves of the suffix tree are traversed in left-to-right order then all the suffixes of the text are retri lexicographical order.

- A suffix array is like a simple array and it contains all the pointers to the text suffixes listed in lexicographical order.
- This is shown in Fig. 4.2.3. As suffix array stores one pointer per indexed suffix so the space requirements are almost the same that of inverted indices.

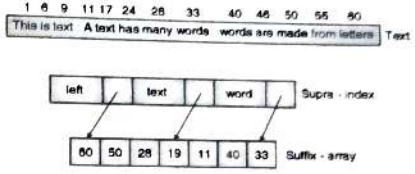


Fig. 4.2.3 : Suffix array for sample text

- Suffix arrays are designed to allow binary searches by comparing the contents of each pointer. If the suffix array is large then this binary search performs poorly because of the number of random disk accesses. To avoid this problem supra-indices over the suffix array has proposed.
- In supra-indices a sampling of one out of b suffix array entries where for each sample *l* and suffix characters are stored in the supra-index.
- This supra-index is then used as a first step of the search to reduce external accesses.
- Fig. 4.2.4 shows supra-index over suffix array. In this figure one out of three entries are sampled. Keeping their first four characters.
- In supra-index there is no need to take samples at fixed intervals or of the same length. For word-indexing suffix array.

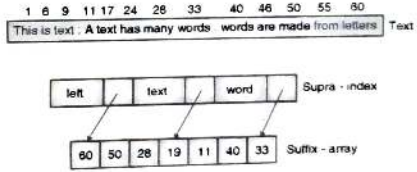


Fig. 4.2.4 : Supra index over suffix array

- A new sample could be taken each time the first word of the suffix changes and word is stored instead of *l* characters.
- The difference between suffix and inverted index is that, the occurrences of each word in an inverted index are sorted by text position. While in suffix array they are sorted lexico-graphically by the text following the word. This is shown in Fig. 4.2.5.
- The space requirements of the suffix array with a vocabulary supra index are exactly the same as for inverted indices.

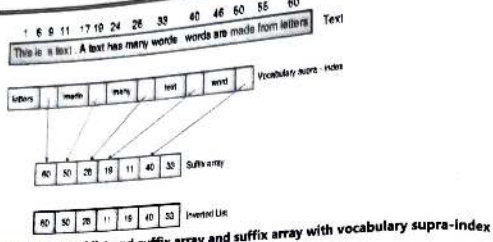


Fig. 4.2.5 : Relation between inverted list and suffix array and suffix array with vocabulary supra-index

4.2.3 Searching in Suffix Tree

- By using simple trie search, many basic patterns such as words, prefixes and phrases can be searched in O(m) ti if suffix tree on the text is afforded.
- Suffix array perform the same search operation in O(1.9 m) time by doing a binary search instead of a tree searc
- This searching is performed as follows.
  1. Search starts with two 'limiting patterns' P_1 and P_2.
  2. We have to search for any suffix s such that $P_1 \leq S \leq P_2$.
  3. Both limiting patterns are binary searched in the suffix array.
  4. All the elements lying between both positions point to exactly those suffixes that start like the original patte
  5. For example in Fig. 4.2.5 to find the word 'text' who have to search for 'text' and 'texu' obtain the portion the array that contains the pointers 19 and 11.
- Simple phase searching is good for this kind of indices. A simple phrase of words can be searched as if it wa simple pattern because the suffix tree or array sorts with respect to the complete suffixes and not only their f word.
- A proximity search has to be solved element-wise. The matches for each element must be collected and sorted a then they have to be intersected as for inverted files.
- The binary search performed on suffix arrays is done on disk, where the accesses to text positions force a se operation which spans the disk tracks containing the text. As random seek is O(n) in size, binary search cost O(n log n) time.
- To avoid performing O (log n) random accesses to the text on disk, the search starts in the supra-index which fits main memory.
- After this search is completed, the suffix array block which is between the two selected samples is bought in memory and the binary search is completed.

4.2.4 Construction in Main Memory

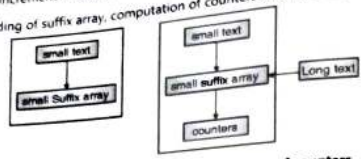
- For a text of n characters, a suffix tree can be built in O(n) time.

- The algorithm does not perform good if the suffix tree cannot fit in main memory because of space requirements.
- In this section we will discuss the direct suffix array construction because the suffix array is nothing but the set of pointers lexicographically sorted, the pointers are collected in ascending text order and then sorted by the text they point to.
- We need to access the corresponding text positions to compare two suffix array entries. And these accesses are random so, both the suffix array and the text must be in main memory. This method of construction costs O (n log n) string comparisons.
- All the suffixes are bucket-sorted in O (n) time according to the first letter. Then each bucket is bucket-sorted again according to their first two letters.
- At iteration i, the suffixes begin already sorted by their 2^{i-1} first letters and end up sorted by their first 2^i letters.
- At each iteration the total cost of all the bucket sorts is O(n), the total time is O(n log n) and the average is O(log log n) because O(log n) comparisons are required on average to distinguish two suffixes.
- In order to sort the strings in the i^{th} iteration, all suffixes are sorted by their first 2^{i-1} letters to sort the text positions T_a... and T_b... in the suffix array. It is sufficient to determine the relative order between text positions $T_{a+2^{i-1}}$ and $T_{b+2^{i-1}}$ in the current stage of the search.

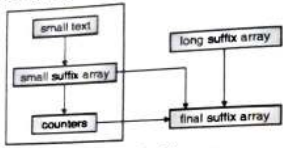
4.2.5 Construction of Suffix Arrays for Large Texts

- Large text database will not fit in main memory so for such databases it is possible to apply an external memory sorting algorithm.
- But in such kind of sorting algorithm each comparison involves accessing the text at random positions on the disk and this random access decreases the performance of the sorting process.
- So the algorithm is especially designed for large texts and it consists of following steps.
  1. Divide the text into different blocks that can be sorted in main memory.
  2. For each block build suffix array in main memory.
  3. Build the suffix array for first block.
  4. Build the suffix array for second block.
  5. Merge the suffix arrays of both the blocks.
  6. Build the suffix array for the third block.
  7. Merge this suffix array with the array formed by merging suffix array of first and second block.
  8. Then perform this building suffix array and merging with previous suffix array for each and every block.
- It is difficult to merge large suffix array with small suffix array because for merging we have to compare text positions which are spread in a large text.
- We can do this by determining the number of elements of the large array to be placed between each pair of elements in the small array and then using this information to merge the arrays without accessing the text and counters are used to stoke this information.
- The counters are computed without using large suffix array. The text corresponding to large suffix array is sequentially dead into main memory.

- Then each suffix of that text is searched in the small suffix array. Once we know the inter-element position the suffix lie we have to increment the appropriate counter. This is illustrated in Fig. 4.2.6.
- Fig. 4.2.6 shows the building of suffix array, computation of counters and merging of suffix arrays.



(a) Building of local suffix array (b) Computation of counters



(c) Merging of suffix array

Fig. 4.2.6 : Construction of suffix array for large sets

- If there is O (m) main memory to index then there will be O(n/m) text blocks. Each block is merged against an of size O(n) where all the O(n) suffixes of the large text are binary searched in the small suffix array. This gi total complexity of $O(n^2 \log (M) / M)$.
- So the construction process is more costly as compare with inverted files.

4.2.6 Difference between Suffix Array and Suffix Tree

Q.	Explain the difference between suffix array and suffix tree.	(8 Mar
Q.	Compare with example suffix array and suffix tree.	(8 Mar
Q.	Give the difference between suffix array and suffix tree.	(5 Mar

Table 4.2.1

Sr. No.	Suffix Array	Suffix Tree
1	Memory efficient-Requires less memory	Requires more memory.
2.	Can answer more complex queries.	Cannot answer complex queries.
3.	Construction is costly.	Construction is cheap.
4.	Results are not delivered in text position order.	Results are delivered in text position order.
5.	Binary search is possible.	Binary search is not possible.

Sr. No.	Suffix Array	Suffix Tree
6.	Uses supra index for fast retrieval.	Uses Patricia tree to save memory
7.	Uses Array, linear data structure for construction.	Uses tree, non linear data structure for Construction
8.	Uses Arrays as supporting data structures.	Uses trie as supporting data structures
9.	Can be used for large texts.	Not practical for large texts.
10.	Example 1 11 19 28 33 40 46 50 60 This is a text. A text has many words. Words are made from letters. Refer Fig. 4.2.7(a).	Example- 1 11 19 28 33 40 46 50 60 This is a text. A text has many words. Words are made from letters. Refer Fig. 4.2.7(b).

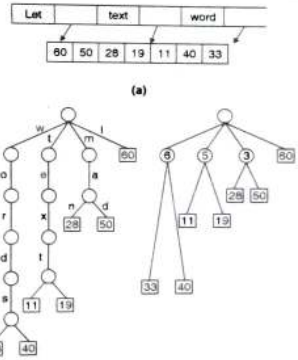


Fig. 4.2.7

4.3 Signature Files

Q.	Explain Signature file with example.	(8 Marks)
Q.	What is signature file? How can it be constructed?	(4 Marks)

- Signature files are index structured based on hashing and are word oriented.

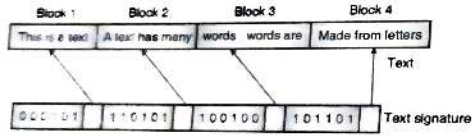
- They posses a low overhead at the cost of forcing a sequential search over the index.
- Search complexity is linear but constant.
- It is not suitable for very large texts.

4.3.1 Structure of Signature Files

Q. Explain signature structure in detail. **(5 Marks)**

Q. Explain with example signature file structure. **(8 Marks)**

- Signature file used hash function which maps words to bit masks of B bits.
- It divides text in blocks of b words each.
- Then it assigns a bit mask of size B to each text block of size b.
- The mask is obtained by performing bitwise ORing the signatures of all the words in the text block.
- So signature file is the sequence of bits masks of all blocks with a pointer to each block.
- If the word is present in a text block, then all the bits set in its signature are also set in the bit set in its signature are also set in the bit mask of the text block.
- Whenever a bit is set in the mask of the query word and not in the mask of the text block then the word is not present in the text block.
- Fig. 4.3.1 shows the example. In this, the sample text is cut into blocks.



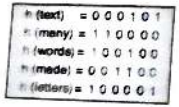


Fig. 4.3.1 : Signature file for sample text

- Even if the word is not there then also it is possible to set all the corresponding bits. And this is called as false drop.
- So, while designing signature file, the probability of a false drop should be low enough.
- The hash function is forced to deliver bit masks which have at least l bits set. A good model assumes that l bits are randomly set in the mask.
- Let $\propto = l / B$. As each of the b words sets l bits at random. The probability that a given bit of the mask is set in the word signature is

$$1 - (1 - 1/B)^{bl} = 1 - e^{-b\propto}$$

- So, the probability that the l random bits set in the query are also set in the mask of the text block is $(1 - e^{-b\propto})^{\propto B}$

  Which is minimized for $\propto = \ln(2) / b$

- The false drop probability under the optimal selection is

  $l = B \ln(2) / b$ is

  $(1 / 2^{\ln(2)})^{B/b} = 1/2^l$

- The space overhead of the index is approximately $(180) \times (B/b)$ because B is measured in bits and b in words. Then the false drop probability is a function of the overhead to pay.

4.3.2 Searching in Signature Files

- In signature file searching a single word is done by hashing it to a bit mask W and then comparing the bit masks Bi of all the text blocks.
- Whenever there is W and B = W (and is the bitwise AND ing) all the bits set is W are also set in Bi and therefore the text block may contain the word.
- So far all candidate text blocks, an online traversal must be performed to verify if the word is actually there.
- This scheme is efficient to search phrases and reasonable proximity queries because all the words must be present in a block in order for that block to hold the phrase or the proximity query.
- Hence, the bitwise OR of all the query masks is searched so that all their bits must be present. This reduces the probability of false drops.
- Some case must be taken at block boundaries to avoid missing a phrase which crosses a block limit. To allow searching phrases of j words or proximities of upto j words, consecutive blocks must overlap in j words.
- If the blocks correspond to retrieval units, simple Boolean conjunctions involving words or phrases can also be improved by forcing all the relevant words to be in the block.

4.3.3 Construction of Signature File

- The construction of signature file is very easy. In this the text is cut in blocks and for each block an entry of the signature file is generated. This entry is the bitwise OR of the signatures of all the words in the block.
- Adding text in signature file is easy, it is only necessary to keep adding records.
- Text deletion in signature file is carried out by deleting the appropriate bit masks.
- It is possible to make a different file for each bit of the mask i.e. one file holding all the first bits, another file holding all the second bits and so on.
- This reduces the disk times to search for a query. Since only the files corresponding to bits which are set in the query have to be traversed.

4.4 Scatter Storage or Hash Addressing

Q. Explain the concept of hash addressing. **(5 Marks)**

- Scatter Storage use hash addressing as technique to store the files.

- Hash Addressing (HA) is technique to assign a location to the file using hash function on the KEY of the file.
- KEY of a file can be any unique property (like record no or name of file).
- KEY can be combination of multiple properties of a file.
- Here it is assumed that at each location there is only a single file.
- In HA most important function is hashing function (f).
- Hashing function (f) should distribute the address of the files uniformly over the available storage.
- Hashing function (f) is bottleneck for the performance of the Scatter Storage.
- Scatter Storage fails in case of a poor hashing function (f).

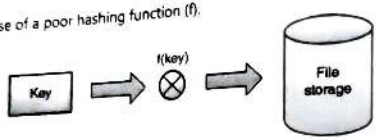


Fig. 4.4.1

4.5 Clustered Files

- Clustered files or Clustered File System (CFS) is
  - Simultaneously mounted on multiple nodes or servers
  - Gives location independent addressing
  - Supports redundancy

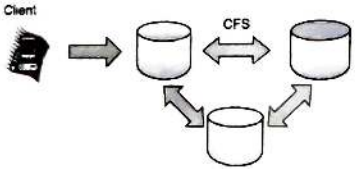


Fig. 4.5.1

- In CFS all nodes or servers work together to give best performance to the client.
- CFS internally distributes the elements or files to different nodes.
- Redundancy is supported to give reliability to the system.
- Parallel file systems are a type of CFS.
- Example : Active Scale file system of Panansa.

4.6 Boolean Queries and Introduction to Sequential Searching

- User provides his search requirement through query. Query is a single word or the collection of words provided by the user. The vocabulary search is carried out using any suitable data structure.

- When the query has more than one word, we have to consider two forms
  1. **Conjunctive queries :** Conjunctive queries imply to search for all the words in the query.
  2. **Disjunctive queries :** Disjunctive queries imply to search for either of the words in the query.
- Boolean queries are the example of multiple word queries where the words are connected with the logical operators such as AND, OR. In Boolean query, the query syntax tree can be constructed.
- For example :

  Consider the query:
  - Translation AND (syntax or semantic)
  - In this case, user is interested in the result where translation word is must and either syntax or semantic word is present in the result. The query can be represented in the syntax tree structure as shown in Fig. 4.6.1.

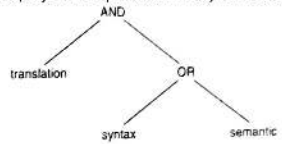


Fig. 4.6.1 : Query syntax tree

4.6.1 Searching in Boolean Queries

Normally, for Boolean queries, the search proceeds in three phases

1. The first phase determines which documents to match
2. The second determines the likelihood of relevance of the documents matched
3. The final phase retrieves the exact positions of the matches to allow highlighting them during browsing, if required
   - Once the leaves of the query syntax tree find the classifying sets of documents, these sets are further operated by the internal nodes of the tree.
   - Under this scheme, it is possible to evaluate the syntax tree in full or lazy form
   - In lazy evaluation, the partial results from operands are delivered only when required, and then the final result is recursively generated

4.6.2 Processing the Internal Nodes of the Query Syntax Tree

- The processing of the internal nodes of the query syntax tree is in 2 forms
  1. **Full evaluation of the syntax tree :** In the full evaluation form, both operands are first completely obtained and then the complete result is generated.

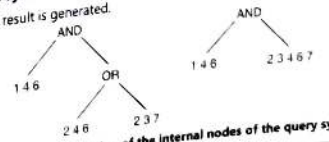


Fig. 4.6.2 : Full form evaluation of the internal nodes of the query syntax tree

Fig. 4.6.2 shows the step by step evaluation of the query specified in Fig. 4.6.1. In this case, for any operator, both the operands are fully evaluated and then the complete result is generated. In this example, the documents 1,4,6 are relevant to the word "translation", the documents 2,4,6 are relevant to the word "syntax" and documents 3,4,7 are relevant to the word "semantic". Once the list of all documents related to the words are ready, the respective operators AND, OR are executed on this list and the final result is generated.

2. **Lazy evaluation of the syntax tree :** In the lazy evaluation form, the partial results from operands are delivered only when required, and then the final result is recursively generated. Fig. 4.6.3 shows the step-by-step evaluation of the query.

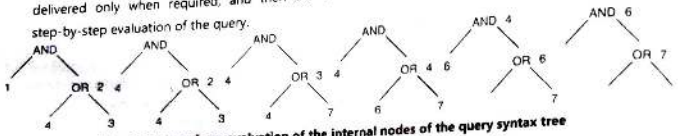


Fig. 4.6.3 : Lazy form evaluation of the internal nodes of the query syntax tree

4.7 Sequential Searching

In general the sequential search problem is :

- Given a text $T = t_1 t_2 \ldots t_n$ and a pattern denoting a set of strings P, find all the occurrences of the strings of P in T
  - **Exact string matching :** the simplest case, where the pattern denotes just a single string $P = p1p2 \ldots pm$
- This problem covers many of the basic queries, such as word, prefix, suffix, and substring search. We assume that the strings are sequences of characters drawn from an alphabet Σ of size σ
- Sequential searching : simple strings

The brute force algorithm :

- In brute force method, the given text is traced to search the given pattern. Try out all the possible pattern positions in the text and checks them one by one. The algorithm slides a window of length m across the text, $t_{i-1}t_{i+2} \ldots t_{i+m}$ for $0 \le i \le n - m$. Each window denotes a potential pattern occurrence that must be verified. Once verified, the algorithm slides the window to the next position. The time requirement for brute force method is O(mn) in worst case, but O(n) in average case. Fig. 4.7.1 shows the example of brute force method. In this case, the given text is T = abracabracadabra and the pattern to match is P = abracadabra. The algorithm starts with the window size equal to the number of characters present in the pattern, and try to match the text window with the pattern. In the next step, the text window is shifted by one position and the process is repeated. The algorithm iterates the process till it checks all the possible windows of the text and reaches to the end of the text.

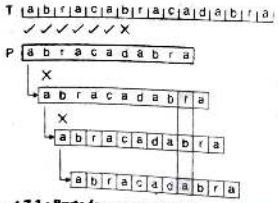


Fig. 4.7.1 : Brute force method for sequential search

Simple Strings : Horspool

- Horspool's algorithm is in the fortunate position of being very simple to understand and program. It is the fastest algorithm in many situations, especially when searching natural language texts. Horspool's algorithm uses the previous idea to shift the window in a smarter way.
- A table *d* indexed by the characters of the alphabet is pre-computed as *d[c]* tells how many positions can the window be shifted if the final character of the window is c. In other words, *d[c]* is the distance from the end of the pattern to the last occurrence of c in P, excluding the occurrence of p_m. Figure shows the same example mentioned above. Here rather than consider all windows, the Horspool shift is taken place.

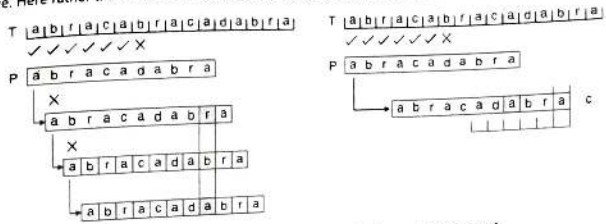


Fig. 4.7.2 : Horspool algorithm example for sequential search

- Following is the pseudo code for Horspool algorithm :

Horspool ($T = t_1 t_2 \ldots t_n$, $P = p_1 p_2 \ldots p_m$)

1. for $c \in \sum$ do $d[c] \leftarrow m$
2. for $j \leftarrow 1 \ldots m - 1$ do $d[p_j] \leftarrow m - j$
3. $i \leftarrow 0$
4. **while** $i \le n - m$ **do**
5. $j \leftarrow 1$
6. **while** $j \le m \wedge t_{i+j} = p_j$ **do** $j \leftarrow j + 1$
7. **if** $j > m$ **then** report an occurrence at text position $i + 1$
8. $i \leftarrow i + d[t_{i+m}]$

Fig. 4.7.3 : Pseudo code of Horspool algorithm

- When searching for long patterns over small alphabets Horspool's algorithm does not perform well. This problem can be alleviated by considering consecutive pairs of characters to shift the window. In other words, we can align the pattern with the last pair of window characters, $t_{i+m-1}t_{i+m}$
- In general we can shift using q characters at the end of the window: which is the best value for q: We cannot shift by more than m, and thus $\sigma^q \le m$ seems to be a natural limit. If we set $q = \log_\sigma m$, the average search time will be $O(n \log_\sigma(m)/m)$. Actually, this average complexity is optimal, and the choice for q we derived is close to correct. It can be analytically shown that, by choosing $q = 2 \log_\sigma m$, the average search time achieves the optimal $O(n \log_\sigma(m)/m)$.

- This technique is used in the agrep software. A hash function is chosen to map q-grams (strings of length q) onto an integer range. Then the distance from each q-gram of P to the end of P is recorded in the hash table. For the q-grams that do not exist in P, distance m – q + 1 is used.
- Following is the pseudo code for agrep algorithm :

Agrep ($T = t_1t_2 \ldots t_n$, $P = p_1p_2 \ldots p_m$, q, h( ), N)

1. **for** $i \in [1, N]$ **do** $d[i] \leftarrow m - q + 1$
2. **for** $j \leftarrow 0 \ldots m - q$ **do** $d[h(p_j + 1p_j + 2 \ldots p_{j+q})] \leftarrow m - q - j$
3. $i \leftarrow 0$
4. **while** $i \leq n - m$ **do**
5. $s \leftarrow d[h(t_{i+m-q+1} t_{i+m-q+2} \ldots t_{i+m})]$
6. **if** $s > 0$ **then** $i \leftarrow i + s$
7. **else**
8. $j \leftarrow 1$
9. **while** $j \leq m \wedge t_{i+j} = p_j$ **do** $j \leftarrow j + 1$
10. **if** $j > m$ **then** report an occurrence at text position $i + 1$
11. $i \leftarrow i + 1$

Fig. 4.7.4 : Pseudo code of Agrep algorithm

Automata and Bit-Parallelism :

Horspool's algorithm, as well as most classical algorithms, does not adapt well to complex patterns. We now show how automata and bit-parallelism permit handling many complex patterns.

Automata :

- Deterministic Finite Automata(DFA) and Non-deterministic Finite Automata (NDFA) are used to match the complex patterns with the text. Fig. 4.7.5 shows the example of NDFA for the pattern abracadabra. The loop defined for initial state of NDFA shows that all other characters can occur in text at the start. Other edges shows the transition from one state to another state in automata. Once this pattern occurs in the text, the control reached to the end state which is shown with double circle. Note that several states can be simultaneously active. For example, after reading 'abra', NFA states 0, 1, and 4 will be active

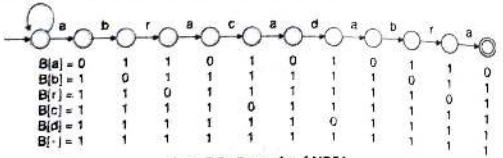


Fig. 4.7.5 : Example of NDFA

Bit-parallelism and Shift-And

- Bit-parallelism takes advantage of the intrinsic parallelism of bit operations.

- Bit masks are read right to left, so that the first bit of $b_m \ldots b_1$ is b_1.
- Bit masks are handled with operations like:
  - | to denote the bit-wise or
  - & to denote the bit-wise and, and
  - ! to denote the bit-wise XOR
  - Unary operation '~' complements all the bits
  - *mask* << *i* means shifting all the bits in mask by i positions to the left, entering zero bits from the right
  - mask >> i is shifting all the bits in mask by *i* positions to the right, entering zero bits from the left
- It is possible to operate bit masks as numbers, for example adding or subtracting them

Shift-And algorithm

- The simplest bit-parallel algorithm permits matching single strings, and it is called Shift-And.
- The algorithm builds a table *B* which, for each character, stores a bit mask $b_m \ldots b_1$. The mask in *B[c]* has the i-th bit set if and only if $p_i = c$.
- The state of the search is kept in a machine word $D = d_m \ldots d_1$, where d_i is set if the state i is active. Therefore, a match is reported whenever $d_m = 1$.
- State number zero is not represented in D because it is always active and then can be left implicit.
- Fig. 4.7.6 shows the pseudo code for Shift-And algorithm.

Shift-And ($T = t_1t_2 \ldots t_n$, $P = p_1p_2 \ldots p_m$)

1. **for** $c \in \sum$ **do** $B[c] \leftarrow 0$
2. **for** $j \leftarrow 1 \ldots m$ **do** $B[p_j] \leftarrow B[p_j] \mid (1 << (j - 1))$
3. $D \leftarrow 0$
4. **for** $i \leftarrow 1 \ldots n$ **do**
5. $D \leftarrow ((D << 1) \mid 1) \,\&\, B[t_i]$
6. **if** $D \,\&\, (1 << (m - 1)) \neq 0$
7. **then** report an occurrence at text position $i - m + 1$

Fig. 4.7.6 : Pseudo code for Shift-And algorithm

4.8 Scoring in Information Retrieval System

4.8.1 Scoring

In the first section of the chapter, we have handled the indices which handles Boolean queries. In terms of Boolean queries, a document either matches or does not match. But in practical applications, such kind of match retrieves very less documents. Also, the search engines needs to rank the documents according to their match with the query. In case of Boolean match, we cannot assign the rank to the collection. Thus we need to go one step ahead, and need to understand how search engines assign the rank values to the collection. In such cases, a score value is assigned to the documents with respect to the queries. Score value defines how the document is relevant to the query. The score value is in the range of 0 to 1. According to this score value, the search engine can sort the documents and provides the resultant collection of documents to the user.

4.8.2 Parametric and Zone Indexes

- Till yet, we have discussed that the document is the sequence of terms. Actually, for each document, additional information is also save which is called Metadata of the document.
- Metadata of the document contains the fields such as,
  - o date of creation
  - o format of the document
  - o author
  - o title of the document

  Fields of the metadata are finite.
- When user enters the query, the fields in the metadata are used for searching purpose. For example if user gives the query, "List of books whose publication date ranges from 1 Jan 2007 to 1 Jun 2022". In this case, the date of creation of the document is used for defining the relevance of the document. Such queries are called as **parametric queries.**
- Fig. 4.8.1 shows the user's view of such a parametric search.

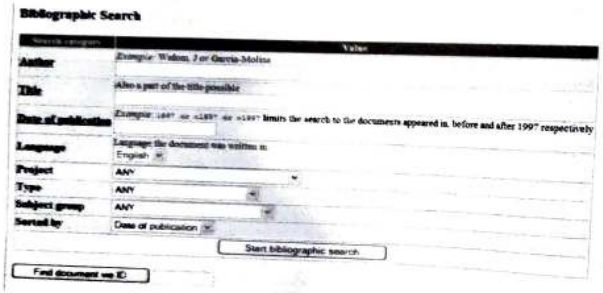


Fig. 4.8.1 : Bibliographic search with number of parameters

Zones :

- Zones are similar like fields. Only the difference is the zone can be arbitrary free text. Fields are relatively small set of values, but the zones can be the unbounded text. As the zone may contain large texts, a separate inverted indices can be built for the zones. Consider the query, "find documents with merchant in the title and william in the author list and the phrase gentle rain in the body". We can build the index which is shown in Fig. 4.8.2. The entries for fields may come from fixed words whereas the zone index is a structure whatever vocabulary stems from the text of that zone.

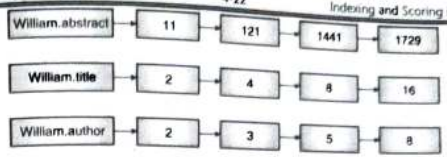


Fig. 4.8.2 : Basic zone index; zones are encoded as extensions of dictionary entries

Fig. 4.8.3 : Zone index in which the zone is encoded in the postings rather than the dictionary

4.8.3 Weighted Zone Scoring

- Weighted zone scoring is also called as "ranked Boolean retrieval". In this case, when user enters the Boolean query, the score value is assigned to the document with respect to query q. The weight value for document d with respect to Boolean query q is normalized and in the range of [0,1].
- Consider the following parameters :
  - o L : The number of zones present in each document
  - o $g1, \ldots, gl \in [0, 1]$ such that $\sum l\, i=1\, gi = 1$.
  - o s_i : the Boolean score denoting a match (or absence thereof) between q and the i^{th} zone. For instance, the Boolean score from a zone could be 1 if all the query term(s) occur in that zone, and zero otherwise; indeed, it could be any Boolean function that maps the presence of query terms in a zone to 0, 1

$$\text{weighted zone score} = \sum_{i=1}^{l} g_i s_i$$

Example :

Consider that the collection of documents has 3 zones for each document namely, author title and body. Let the query is "Agarkar". The Boolean score function for a zone takes on the value 1 if the query term "Agarkar" is present in the zone, and zero otherwise. For weighted zone scoring, let the 3 values of zones related to query "Agarkar" needs to be score g1 = 0.2, g2 = 0.3 and g3 = 0.5 (so that the three weights add up to 1), this corresponds to an application in which a match in the author zone is least important to the overall score, the title zone somewhat more, and the body contributes even more. Thus if the term "Agarkar" were to appear in the title and body zones but not the author zone of a document, the score of this document would be 0.8.

```
ZoneScore (q1, q2)
1. float scores [N] = [0]
2. constant g[l]
3. p1 ← postings (q1)
4. p2 ← postings (q2)
5. // scores[ ] is an array with a score entry for each document, initialized to zero.
```

```
6.  // p1 and p2 are initialized to point to the beginning of their respective postings.
7.  // Assume g[ ] is initialized to the respective zone weights.
8.  while p1 ≠ NIL and p2 ≠ NIL
9.    do if doc ID(p1) = doc ID(p2)
10.   then scores [docID(p1)] ← WEIGHTEDZONE (p1, p2, g)
11.   p1 ← next (p1)
12.   p2 ← next (p2)
13.   else if doc ID (p1) < docID(p2)
14.   then p1 ← next (p1)
15.   else p2 ← next (p2)
16.   return scores
```

Fig. 4.8.4 : Pseudo code of zone score algorithm

- Fig. 4.8.4 shows the Pseudo code of zone score algorithm.

4.8.4 Learning Weights and Optimal Weight of g

- Assigning the g_i values to the zones of the documents with respect to query is the intelligent task. The machine learning approach can be used to assign specific values to g_i i.e. assigning the weights to the zones. Following approach is followed which is called as learning the weights :

1. A set of training samples is provided. Each training sample contains the document d, query q and the judgement like the document is *relevant* or *not relevant* to the query.
2. The weight g_i is "learned" from these samples and the value of g_i is defined.

Example :

- Consider the documents in the collection has 2 zones
  - Title zone
  - Body zone
- Following variables are considered during computation:
  - $s_T(d, q)$: It is the Boolean variable which is either 0 or 1 and shows that whether Title zone of the document matches with the query q.
  - $s_B(d, q)$: It is the Boolean variable which is either 0 or 1 and shows that whether Body zone of the document matches with the query q.
  - score(d, q): It is the matching score between the document and the query q.
  - Score can be computed as,

$$score(d, q) = g \cdot s_T(d, q) + (1 - g).s_B(d, q)$$

- Consider we have the training examples in the form of $\Phi_j = (d_j , q_j , r(d_j , q_j))$ where,
  - Φ_j: training example
  - d_j: document

- q_j: query
- $r(d_j , q_j)$: relevance value, either 0 or 1. This relevance value is set by the human editor.
- $s_T(dj , qj)$: Boolean value showing match between Title zone of the document to the query
- $s_B(dj , qj)$: Boolean value showing match between Body zone of the document to the query
- score(dj , qj) : score value is computed using,

$$score(d_j , q_j) = g \cdot s_T(d_j , q_j) + (1 - g).s_B(d_j , q_j).$$

- Fig. 4.8.5 shows the set of training examples.

Example	DocID	Query	S_T	S_B	Judgment
Φ_1	37	Linux	1	1	Relevant
Φ_2	37	Penguin	0	1	Non-relevant
Φ_3	238	System	0	1	Relevant
Φ_4	238	Penguin	0	0	Non-relevant
Φ_5	1741	Kernel	1	1	Relevant
Φ_6	2094	Driver	0	1	Relevant
Φ_7	3191	Driver	1	0	Non-relevant

Fig. 4.8.5 : An illustration of training examples

- To assign the values of g_i, the computed score value $score(d_j , q_j)$ is compared with the relevance value $r(d_j , q_j)$ given by the human editor. The error value is computed between score and the given relevance value as,

$$\varepsilon(g, \Phi_j) = (r(d_j , q_j) - score(d_j , q_j))^2$$

- The total error of the set of training examples is computed as,

$$\sum_j \varepsilon(g, \Phi_j)$$

- The values of g, over the interval [0, 1], are selected in such a way that the above error value reduces near to 0.
- The optimal weight g :

To set the optimal value to g, we consider the values set to $s_T(dj , qj)$ and $s_B(dj , qj)$. Consider the equation of score. In the training example, when $s_T(dj , qj) = 0$ and $s_B(dj , qj) = 1$, the score is computed as (1-g). In the same fashion, for other possible values of $s_T(dj , qj)$ and $s_B(dj , qj)$ we can compute the score. Figure shows the possible score values based on S_T and S_B.

S_T	S_B	Score
0	0	0
0	1	1 - g
1	0	g
1	1	1

Fig. 4.8.6 : The four possible combinations of S_T and S_B

- **Let us consider other parameters :**
  - n_{01r} (respectively, n_{01n}) : The number of training examples for which $s_T(dj, qj) = 0$ and $s_B(dj, qj) = 1$ and editorial judgment is Relevant (respectively, Non-relevant).
  - Then the contribution to the total error from training examples for which $s_T(dj, qj) = 0$ and $s_B(dj, qj) = 1$ is,

$$1 - (1 - g)]^2 n_{01r} + [0 - (1 - g)]^2 n_{01n}$$

  - In the similar fashion, we can compute other possible combinations of S_T and S_B values and its error contribution to the total error value.

$$(n_{01r} + n_{10r})g^2 + (n_{10r} + n_{01n})(1 - g)^2 + n_{00r} + n_{11n}$$

  - Differentiate the equation with respect to g and set the result to 0, we get optimal value of g as,

$$\frac{n_{10r} + n_{01n}}{n_{10r} + n_{00r} + n_{01r} + n_{01n}}$$

4.8.5 Term Frequency and Weighting

- In terms of free text query, user enters the query in the form of collection of words. These words are not necessarily connected with logical operators. The queries can be sentences or just the collection of words. In this case the words. The documents which contains these words are relevant to the query.
- **Term** Before understanding the scoring scheme for the document, let us discuss what is term? Term is the stemming word in the sentence. For example, consider the sentence, "My name is Pratibha." In this case we are not interested in the stop words like 'My', 'is' because they do not reflect the semantic of the sentence. Such stopwords are neglected from the sentences and we concentrate only on important words. For remaining words, we find the original word. For example if the word is "dogs", its stem is "dog". Please refer Porter's algorithm which shows the process to find the stems of the word. These stems are considered as the terms.
- The scoring mechanism of the documents considers the number of occurrences of the terms in the document. For each document, the list of terms is maintained along with its score. The term score with respect to the document is the number of occurrences that the term is present in the document. This value is referred as **term frequency**.
- $tf_{t,d}$:is the term frequency of the term in the document d.

Inverse document frequency :

- Inverse document frequency (idf) of a term t is computed as follows :

$$Idf_t = \log \frac{N}{df_t}$$

Where,

- N : Total number of documents in a collection
- df_t : document frequency be the number of documents in the collection that contain a term t
- idf of a rare term is high, whereas the idf of a frequent term is likely to be low.

4.8.6 Tf-idf Weighting

- The term frequency $tf_{t,d}$ and inverse document frequency (idf) is used to define the weight of each term in the document.

$$tf - idf_{t,d} = tf_{t,d} \times idf_t$$

Where,

- $Tf\text{-}idf_{t,d}$ is the weight assigned to term t with respect to document d. It is,
  1. Highest when t occurs many times within a small number of documents (thus lending high discriminating power to those documents);
  2. Lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
  3. Lowest when the term occurs in virtually all documents.
- Consider we have collection of documents. Initially, the terms related to all documents are listed. Then each document is represented using the vector where there is entry for each term in the collection. If the term is not present in the document, then we save weight is 0 for that term w.r.to the document otherwise the tf-idf value is saved in the vector. Once we save the vectors of all documents in the collection, then the system is ready to accept the user query.
- When user enters the query, we find the terms present in the query. Based on the terms of the query, the relevance of the documents w.r.to. query is computed as,

$$\text{Score}(q, d) = \sum_{t \in q} tf\text{-}idf_{t,d}$$

4.9 The Vector Space Model for Scoring

- As the documents are represented using the vectors, the similarity between the two documents or document-query can be computed using the vector theory.

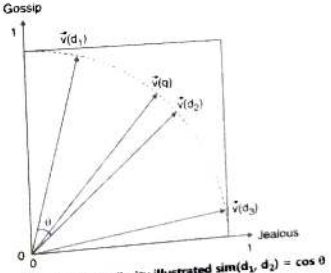


Fig. 4.9.1 : Cosine similarity illustrated $sim(d_1, d_2) = \cos\theta$

- Consider the documents are represented by the vectors which is shown in Fig. 4.9.1. To find the similarity between the documents, we find the cosine similarity of their vector representations $\vec{V}$
- Consider,
  - $\vec{V}(d_1)$: Vector representation of document d_1
  - $\vec{V}(d_2)$: Vector representation of document d_2

- Similarity between the documents is computed using following equation which is the cosine similarity between documents.

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|}$$

- Where the numerator represents the dot product (also known as the inner product) of the vectors $\vec{V}(d_1)$ and $\vec{V}(d_2)$, while the denominator is the product of their Euclidean lengths. The dot product $\vec{x} \cdot \vec{y}$ of two vectors defined as $\sum_{i=1}^{M} x_i y_i$. Let $\vec{V}(d)$ denote the document vector for d, with M components $\vec{V}_1(d) \ldots \vec{V}_M(d)$. The Euclidean length of d is defined to be $\sqrt{\sum_{i=1}^{M} \vec{V}_i^2(d)}$.

Example :

	Doc1	Doc2	Doc3
Car	27	4	24
Auto	3	33	0
insurance	0	33	29
Best	14	0	17

- Consider the collection has 3 documents. Table shows the frequency of the terms car, auto, insurance, best in the documents.
- We now apply Euclidean normalization to the tf values from the table, for each of the three documents in the table.
- The quantity $\sqrt{\sum_{i=1}^{M} \vec{V}_i^2(d)}$ for documents are 30.56, 46.84 and 41.30 respectively for Doc1, Doc2 and Doc3.
- The resulting Euclidean normalized tf values for these documents are shown in following table :

	Doc 1	Doc 2	Doc 3
Car	0.88	0.09	0.58
Auto	0.10	0.71	0
Insurance	0	0.71	0.70
Best	0.46	0	0.41

Euclidean normalized tf values for documents i_1

4.10 Queries as Vectors

- Queries are also represented using the vector in vector model. The similarity between the query and each document can be computed as,

$$score(q, d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}$$

Ex. 4.10.1 : We now consider the query best car insurance on a fictitious collection with N = 1,000,000 documents where the document frequencies of auto, best, car and insurance are respectively 5000, 50000, 10000 and 1000.

Term	Query				Document			Product
	tf	df	idf	$w_{t,q}$	tf	wf	$w_{t,d}$	
Auto	0	5000	2.3	0	1	1	0.41	0
Best	1	50000	1.3	1.3	0	0	0	0
Car	1	10000	2.0	2.0	1	1	0.41	0.82
Insurance	1	1000	3.0	3.0	2	2	0.82	2.46

In this example the weight of a term in the query is simply the idf (and zero for a term not in the query, such as auto); this is reflected in the column header $w_{t,q}$ (the entry for auto is zero because the query does not contain the termauto). For documents, we use tf weighting with no use of idf but with Euclidean normalization. The former is shown under the column headed wf, while the latter is shown under the column headed $w_{t,d}$. Invoking (6.9) now gives a net score of 0 + 0 + 0.82 + 2.46 = 3.28.

4.11 Computing Vector Scores

```
COSINESCORE(q)
1. float Scores [N] = 0
2. Initialize Length [N]
3. for each query term t
4. do calculate w_{t,q} and fetch postings list for t
5. for each pair (d, tf_{t,d}) in postings list
6. do Scores [d] + = wf_{t,d} × w_{t,q}
7. Read the array Length [d]
8. for each d
9. do Scores [d] = Scores [d] / Length [d]
10. return Top K components of scores [ ]
```

Fig. 4.11.1 : The basic algorithm for computing vector space scores

- Figure shows the algorithm to compute the score value between document and the query. Consider the collection contains N documents and each document is represented by vector. Also, the query is represented in terms of vector. We wish to find K relevant documents to the query with highest score. For each document, we compute the similarity value between query and the document with the help of tf-idf value. Finally, first K documents with highest score values are returned to the user.

4.12 Efficient Scoring and Ranking

- In above algorithm, it may possible that two documents get same relevance value with query q. For examp document 4 and docuemnt8 are having 0.707 similarity value with the query. In this situation, we need to def which documents should come first; whether document 4 or document 8. In such cases, we need relative scor (rather than absolute score).
- To simplify this situation, the algorithm is little bit updated :

```
FASTCOSINESCORE( q)
1  float Scores [N] = 0
2  for each d
3  do Initialize Length [d] to the length of doc d
4  for each query term t
5  do calculate w_t,q and fetch postings list for t
6  for each pair (d, tf_t,d) in postings list
7  do add wf_t,d to Scores [d]
8  Read the array Length [d]
9  for each d
10 do Divide Scores[d] by Length[d]
11 return Top K components of Scores[ ]
```

Fig. 4.12.1 : A faster algorithm for vector space scores

- To compute the cosine similarity from each document unit vector $\vec{V}(d)$ to $\vec{V}(q)$ (in which all non-zero components of the query vector are set to 1), rather than to the unit vector $\vec{V}(q)$. For any two documents d_1 and d_2,

$$\vec{V}(q) \cdot \vec{V}(d_1) > \vec{V}(q) \cdot \vec{V}(d_2) \Leftrightarrow \vec{V}(q) \cdot \vec{V}(d_1) > \vec{V}(q) \cdot \vec{V}(d_2)$$

- $\vec{V}(q) \cdot \vec{V}(d)$ is the weighted sum of all the terms in the query, of the weights of those terms in d.
- To define the score of the document, following steps are followed:
  1. For each document, initialize length[d] as the length of document.
  2. For each term t in the query q, compute the weight $w_{t,q}$
  3. For each pair (d, $tf_{t,d}$), add $wf_{t,d}$ value to the score[d], where, d is the document, $tf_{t,d}$ is the term frequency value of t w.r.to d, and wf_{td} is the Weight of term t in document d
  4. Divide the score[d] by length[d]
  5. Finally, return top K document entries to the user.

4.13 Inexact Top K Document Retrieval

- When user enters a query, we find the similarity score between the all the documents in the collection to the query. If the collection contains N documents, then we compute N similarity scores out of which highest K entries are displayed to the user.

- Consider the situation in which the collection contains 1,00,000 documents and we wish to display top 50 documents to the user. In this case the computation time required to find the score between all 1,00,000 and the query is the burden on Information retrieval system.
- Hence, the idea came into picture, and following heuristic approach is suggested :
  1. Find a set A of documents that are contenders, where $K < |A| \ll N$. A does not necessarily contain the K top-scoring documents for the query, but is likely to have many documents with scores near those of the top K.
  2. Return the K top-scoring documents in A.
- Thus, rather than handling all N documents, first K entries from set A are displayed to the user. Such kind of heuristic is termed as **Inexact top K document retrieval**. The result is not exact in this heuristic, but the computation time is less and effectively, the retrieval time also get reduced.

Review questions

Q. 1 Write a short note on Suffix trees and suffix arrays.

Q. 2 What is inverted file? Give one example of inverted file.

Q. 3 How is the structure of Boolean query?

Q. 4 Explain the process of searching the relevant documents when Boolean query is entered by the user.

Q. 5 Write short note on: Parametric and zone indices.

Q. 6 How to compute score of the terms in vector model?

Q. 7 Explain the term: TF-IDF weighting.

Q. 8 How the documents and queries represented in vector model?

Q. 9 Write an algorithm for efficient scoring and ranking of the documents in Boolean model.

Q. 10 Write short note on : Inexact Top K Document Retrieval.

□□□

5 Evaluation of Information Retrieval Systems

UNIT V

Syllabus

Information retrieval system evaluation, Standard test collections, Evaluation of unranked retrieval sets, Evaluation of ranked retrieval results, Assessing and justifying the concept of relevance
System quality and user utility, System issues, Refining a deployed system

5.1 Information Retrieval System Evaluation

- Information retrieval system is used for searching the relevant documents with respect to the query. It includes following steps
  - Finding internal representation of documents.
  - Deciding the file structure using which the documents are saved inside the system.
  - Selecting model for matching the documents with the query.
  - Finding internal representation of query.
  - Evaluation of the system.
- Once all the decisions are taken regarding the presentation, organization, and search, the final step before implementation of the IR system is an evaluation of the system.
- The type of evaluation depends on the objective of retrieval system. Thus, the first type of evaluation is **functional analysis** and the second is **error analysis**. Both of these are part of testing phase. Various measures are used for evaluating performance of the system.
- The most common measures of system performance are:
  1. Response Time
  2. Space
- The shorter the response time, the smaller the space used, the better the system is considered to be.
- For a system designed for providing data retrieval, two basic metrics are used :
  1. **Response time :** The time between a query is submitted to the system and the answer given by the system.
  2. **Space :** Space required to store documents and handle query.
- The factors which affects above measures are :
  - Performance of the indexing structures.
  - Interaction with the operating system.

  - Delay in communication channels.
  - Overhead introduced by many software layers which are usually present.
- Such form of evaluation can be referred as **performance evaluation**.
- In information retrieval system, other metrics, besides time and space, are also used. In IR system, query or the request given the user can be exact or vague. Hence retrieved documents are not the exact answer and need to be ranked according to the relevance to the query. Thus IR systems require the evaluation, which indicates how precise the answer set is. This type of evaluation is referred as **retrieval performance evaluation**.
- Here we discuss about the retrieval performance evaluation for information retrieval systems. Such an evaluation is usually based on a test reference collection and evaluation measure.
- The ideal test reference collection consists of :
  1. Collection of documents.
  2. A set of example information requests.
  3. A set of relevant documents (provided by specialists) for each example information requests.
- Given a retrieval strategy S, the evaluation measure quantifies (for each example information request) the **similarity** between the set of documents retrieved by S and the set of relevant documents provided by the specialists. This provides an estimation of the **goodness** of the retrieval strategy S.
- Before considering the evaluation, we must consider which type of task the system will handle.
- The task can be :
  1. Query processed in **batch mode** (i.e. user submits the query and gets the result back).
  2. Query is handled in **interactive session**. Here user will submit the query and see what is in the result. Based on the result given by system, user may refine the query.
  3. Combination of batch and interactive mode.
- In an interactive session, different things may affect the performance of the system and show impact on evaluation. The things which may affect the evaluation are :
  1. User effort
  2. Characteristics of the interface design.
  3. Guidance provided by the system.
  4. Duration of the session.

5.2 Standard Test Collections

- Below is the list of the most standard test collections and evaluation series.
  1. The *Cran field* collection. This was the pioneering test collection in allowing precise quantitative measures of information retrieval effectiveness, but is nowadays too small for anything but the most elementary pilot experiments.

2. *Text Retrieval Conference (TREC)* . The U.S. *National Institute of Standards and Technology* (NIST) has run a large IR test bed evaluation series since 1992. Within this framework, there have been many tracks over a range of different test collections, but the best known test collections are the ones used for the TREC Ad Hoc track during the first 8 TREC evaluations between 1992 and 1999. In total, these test collections comprise 6 CDs containing 1.89 million documents and relevance judgments for 450 information needs, which are called *topics* and specified in detailed text passages. In more recent years, NIST has done evaluations on larger document collections, including the 25 million page *GOV2* web page collection. From the beginning, the NIST test document collections were orders of magnitude larger than anything available to researchers previously and GOV2 is now the largest Web collection easily available for research purposes. Nevertheless, the size of GOV2 is still more than 2 orders of magnitude smaller than the current size of the document collections indexed by the large web search companies.
3. **NII Test Collections for IR Systems (*NTCIR*).** The NTCIR project has built various test collections of similar sizes to the TREC collections, focusing on East Asian language and *cross-language information retrieval* , where queries are made in one language over a document collection containing documents in one or more other languages.
4. Cross Language Evaluation Forum (*CLEF*). This evaluation series has concentrated on European languages and cross-language information retrieval.

5.3 Evaluation of Unranked Retrieval Sets

- Recall and precision are the basic measures for retrieval performance evaluation.
- Consider

I	Information request (of a test reference collection)
R	Set of relevant documents
| R |	Number of documents in the set of relevant documents.
A	Answer set generated by the information retrieval system.
| A |	Number of documents in the answer set.
| Ra |	Number of documents in the intersection of the sets R and A.

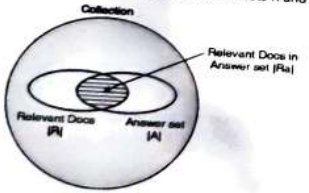


Fig. 5.3.1 : Precision and Recall

Precision and recall for a given example information request

The recall and precision measures are defined as follows

1. **Recall :** It is the fraction of the relevant documents (the set R) which has been retrieved i.e.

$$\text{Recall} = \frac{|Ra|}{|R|}$$

2. **Precision :** It is the fraction of the retrieved documents (the set A) which is relevant i.e.

$$\text{Precision} = \frac{|Ra|}{|A|}$$

- The above measures are defined with the assumption that user will go through all the documents in the answer set. But practically, this happens rarely. In real situations, the documents in the answer set are ranked and then displayed to the user. The user examines the documents from top to bottom.
- In such situations recall and precision vary as the user proceeds with his examination of answer set A. Thus, we require plotting the precision vs recall curve.
- Consider following example :

Consider a reference collection and its set of example information request.

q : The query based on the information request.

R_q : Set of relevant documents to the query q, suggested by the specialist.

Example

$$R_q = \{d_1, d_4, d_{12}, d_{39}, d_{44}, d_{60}, d_{72}, d_{88}, d_{92}, d_{110}\}$$

- Consider the new retrieval system. Assume that the algorithm returns the list of documents relevant to the query q. The ranking of the documents in the answer set as follows :
- Ranking for given query :

1. d_{19}	6. d_{110}	11. d_{50}
2. d_{22}	7. d_{95}	12. d_{60}
3. d_{88}	8. d_{99}	13. d_{77}
4. d_2	9. d_{112}	14. d_{45}
5. d_3	10. d_4	15. d_1

- The list represents the documents which is the outcome of search done by the retrieval system. The actual relevant documents to the query are marked unit the bullets.
- Let us calculate the precision and recall value for these points.

1. Document d_{19}

- It is ranked as number 1 document in the answer set. It is part of the ideal answer set. Hence we can calculate precision at this document as,
- There is single document in the answer set out of which the document itself is actually relevant to the query. Hence precision value is 100% (out of one document one document is relevant)
- Recall can be calculated as the documents which are searched till the first ranked documents are having 10% of actual relevant docs. Hence recall is 10%.

2. Document d_{88}

- Now at these points we are having 3 documents in the answer set i.e. d_{39}, d_{22} and d_{88}. Out of these documents (d_{39} and d_{88}) 2 documents are actually relevant. Hence precision is 66.66% (2 out of 3).
- Recall can be calculated as, till yet we found 2 documents which are part of ideal answer set. The ideal answer set contains 10 documents. Hence recall is 20% (2 out of 10). Thus, the precision and recall values for the are :

Document		Precession	Recall
(d_{39})	(1 out of 1)	100%	10%
(d_{88})	(2 out of 3)	66.66%	20%
(d_{110})	(3 out of 6)	50%	30%
(d_{4})	(4 out of 10)	40%	40%
(d_{1})	(5 out of 15)	33.33%	50%

- We can draw a graph based on the above values.
- Note that precision values drops to 0% for recall value above 50% because further there is no relevant doc present in the list generated by system.
- In the above example, the precision and recall figures are for a single query. Usually, retrieval algorithms are evaluated by running them for several distinct queries. In this case, for each query a distinct precision versus recall curve is generated.

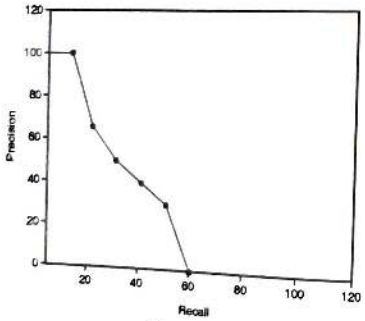


Fig. 5.3.2

- Note that precision values drops to 0% for recall value above 50% because further there is no relevant doc present in the list generated by system.
- In the above example, the precision and recall figures are for a single query. Usually, retrieval algorithms are evaluated by running them for several distinct queries. In this case, for each query a distinct precision versus recall curve is generated.

- To evaluate the retrieval performance of an algorithm over all test queries, we can average the precision figures at each recall level as follows

$$\bar{P}(r) = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q}$$

Where,

$\bar{P}(r)$: Average precision at recall level r

N_q : Number of queries used.

$P_i(r)$: Precision at recall level r for i^{th} query

Ex. 5.3.1 : Explain the term precision and recall and calculate the same for following example. The set of relevant documents in the query Q = {d3, d7, d8, d11, d14, d19, d23, d25}
A new retrieval algorithm returns following answer
set = {d1, d2, d3, d7, d9, d10, d14, d20, d23, d24, d25}

Soln. :

Let R = Set of relevant documents = {d3, d7, d8, d11, d14, d19, d23, d25}

| R | = Number of documents in the set of relevant documents = 8

A = Answer set of algorithm

= {d1, d2, d3, d7, d9, d10, d14, d20, d23, d24, d25}

| A | = Number of documents in the answer set = 11

Ra = Set of documents present in intersection of the sets R and A.

= {d3, d7, d14, d23, d25}

| Ra | = Number of documents in the intersection of the sets R and A. = 5

If we assume that user can see all documents in the answer set at a time, then a single value will be for precision and recall.

1. **Recall :** It is the fraction of the relevant documents (the set R) which has been retrieved i.e.

$$\text{Recall} = \frac{|Ra|}{|R|} = 5/8 = 0.625$$

2. **Precision :** It is the fraction of the retrieved documents (the set A) which is relevant i.e.

$$\text{Precision} = \frac{|Ra|}{|A|} = 5/11 = 0.4545$$

But if our algorithm lists documents one by one then we can define ranking for documents based on the sequence in which documents are listed: Thus the ranking of documents will be

1.	d_1	6.	d_{10}	11.	d_{25}
2.	d_2	7.	d_{14}		
3.	d_3	8.	d_{20}		
4.	d_7	9.	d_{23}		
5.	d_8	10.	d_{24}		

Document	Precision	Precision	Recall	Recall
(d_3)	(1 out of 3)	33.33%	(1 out of 8)	12.5%
(d_7)	(2 out of 4)	50%	(2 out of 8)	25%
(d_{14})	(3 out of 7)	42.85%	(3 out of 8)	37.5%
(d_{23})	(4 out of 9)	44.44%	(4 out of 8)	50%
(d_{25})	(5 out of 11)	45.45%	(5 out of 8)	62.5%

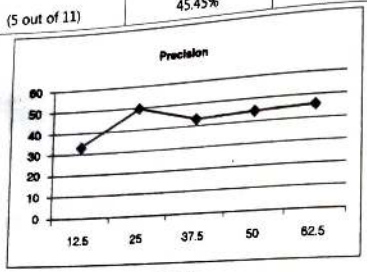


Fig. P. 5.3.1

Ex. 5.3.2 : Consider a reference collection and its set of example information request. If q is the information request and a set Rq = {d3, d5, d9, d25, d39, d44, d50, d70, d80, dl20}.Now consider new retrieval algorithm has been designed and has been evaluated for information request q returns, ranking of the documents in the answer set as.

1. d120 2. d84 3. d50 4. d6 5. d8 6. d9 7. d58 8. d129
9. d143 10. d25 11. d38 12. d48 13. d230 14. d113 15. d3

The documents that are relevant to the query q are underlined. Calculate precision and recall for the documents that are relevant to the query q.

Soln. : To calculate the precision and recall value for the documents that are relevant to the query q, following steps need to be follow :

1. Document d_{120}

- It is ranked as number 1 document in the answer set. It is part of the ideal answer set. Hence we can calculate precision at this document as,
- There is single document in the answer set out of which the document itself is actually relevant to the query. Hence precision value is 100% (out of one document one document is relevant)
- Recall can be calculated as the documents which are searched till the first ranked documents are having 20% of actual relevant docs. Hence recall is 20%.

2. Document d_{50}

- Now at this point we are having 3 documents in the answer set i.e. d_{120}, d_{84} and d_{50}. Out of these 3 documents (d_{120} and d_{50}) 2 documents are actually relevant. Hence precision is 66.66% (2 out of 3).

- Recall can be calculated as, till yet we found 2 documents which are part of ideal answer set. The ideal answer set contains 5 documents. Hence recall is 40% (2 out of 5).

3. Document d_9

- Now at these points we are having 6 documents in the answer set i.e. d_{120}, d_{84}, d_{50}, d_6, d_9 and d_8. Out of these 6 documents (d_{120}, d_{50}, and d_9) 3 documents are actually relevant. Hence precision is 50% (3 out of 6).
- Recall can be calculated as, till yet we found 3 documents which are part of ideal answer set. The ideal answer set contains 5 documents. Hence recall is 60% (3 out of 5).

4. Document d_{25}

- Now at this point we are having 10 documents in the answer set. Out of these 10 documents (d_{12}, d_{50}, d_9, andd_{25}) 4 documents are actually relevant. Hence precision is 40% (4 out of 10).
- Recall can be calculated as, till yet we found 4 documents which are part of ideal answer set. The ideal answer set contains 5 documents. Hence recall is 80% (4 out of 5).

5. Document d_3

- Now at this points we are having 15 documents in the answer set. Out of these 15 documents (d_{12}, d_{50}, d_9, d_{25}, and d_3) 5 documents are actually relevant. Hence precision is 33.33% (5 out of 15).
- Recall can be calculated as, up till we found 5 documents which are part of ideal answer set. The ideal answer set also contains 5 documents. Hence recall is 100% (5 out of 5).
- Thus, the precision and recall values for the list are :

Document		Precession	Recall
(d_{120})	(1 out of 1)	100%	20%
(d_{50})	(2 out of 3)	66.66%	40%
(d_9)	(3 out of 6)	50%	60%
(d_{25})	(4 out of 10)	40%	80%
(d_3)	(5 out of 15)	33.33%	100%

5.4 Evaluation of Ranked Retrieval Results

- Average precision versus recall figures are useful for comparing the retrieval performance of distinct retrieval algorithms.
- However there are situations in which we would like to compare the retrieval performance of our retrieval algorithms for the individual queries.
- Reasons are :
  1. Averaging precision over many queries might disguise important anomalies in the retrieval algorithms under study.
  2. When we want to compare two algorithms, investigating whether one of them out performs the other for each query in a given set of example queries.
- In these situations, single value can be used. This single value should be interpreted as a summary of the corresponding precision versus recall curve.

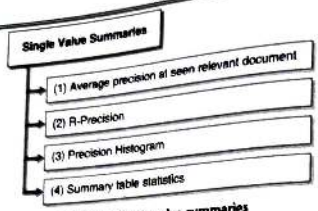


Fig. 5.4.1 : Single value summaries

1. Average precision at seen relevant document

- It is the average of precision figures obtained after each new relevant document is observed.
- For example in above example the precision figures after each new relevant document are 1, 0.66, 0.5, 0.4 and 0.33. Thus, **the average precision of seen documents** is given by,

$$\frac{1 + 0.66 + 0.5 + 0.4 + 0.33}{5} = 0.578$$

- This measure favours systems which retrieve relevant documents quickly (i.e. early in the ranking). An algorithm might present good precision but may have poor performance in terms of recall.

2. R-Precision

- R-precision is the precision value at R^{th} level, where R is the total number of relevant documents for the current query (i.e. number of documents in the set R_q).
- Consider the above example. The value of R-precision is 0.4 for this example (because R = 10 and there are four relevant documents out of first 10 documents). The R-precision measure is useful for checking the behaviour for single query.

3. Precision Histogram

- The R-precision values for several queries can be used to compare the performance of two different algorithms.
- Let, we have two algorithms A and B.

$RP_A(i)$ = R-precision value of algorithm A for i^{th} query.

$RP_B(i)$ = R-precision value of algorithm B for the i^{th} query.

Then calculate, $\mathbf{RP_{A/B}(i) = RP_A(i) - RP_B(i)}$

Let consider three different outcomes :

1. $RP_{A/B}(i) = 0$, indicates both algorithms have equivalent performance.
2. $RP_{A/B}(i) > 0$, indicates better retrieval performance by A compared with B.
3. $RP_{A/B}(i) < 0$, indicates better retrieval performance by B, than A.

For 8 queries algorithm A is better than B and for remaining two queries B algorithm is better than A. This type of bar graph is called as **precision histogram**. It helps to compare performance of two different algorithms through visual inspection.

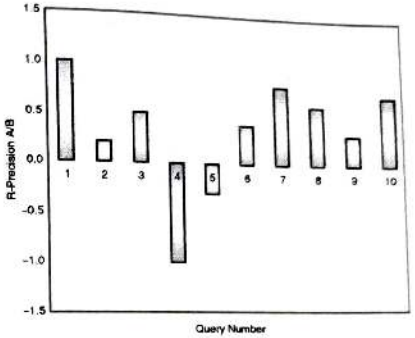


Fig. 5.4.2

4. Summary table statistics

Single value summaries can be stored in a table to provide statistical information about the performance. This table may includes the information like :

1. Number of queries used in the task.
2. Total number of documents.
3. Total number of documents retrieved by all queries.
4. Total number of relevant documents which were effectively retrieved when all queries are considered
5. Total number of relevant documents which could have been retrieved by all queries, etc.

5.5 Assessing and Justifying the Concept of Relevance

- To properly evaluate a system, your test information needs must be germane to the documents in the test document collection, and appropriate for predicted usage of the system. These information needs are best designed by domain experts. Using random combinations of query terms as an information need is generally not a good idea because typically they will not resemble the actual distribution of information needs.
- Given information needs and documents, you need to collect relevance assessments. This is a time-consuming and expensive process involving human beings. For tiny collections like Cranfield, exhaustive judgments of relevance for each query and document pair were obtained. For large modern collections, it is usual for relevance to be assessed only for a subset of the documents for each query. The most standard approach is *pooling*, where relevance is assessed over a subset of the collection that is formed from the top *k* documents returned by a number of different IR systems (usually the ones to be evaluated), and perhaps other sources such as the results of Boolean keyword searches or documents found by expert searchers in an interactive process.

- Observed proportion of the times the judges agreed

$$P(\text{nonrelevant}) = \frac{80 + 90}{400 + 400} = \frac{170}{800} = 0.2125$$

- Pooled marginals

$$P(\text{relevant}) = \frac{320 + 310}{400 + 400} = \frac{630}{800} = 0.7878$$

$$P(\text{nonrelevant}) = \frac{80 + 90}{400 + 400} = \frac{170}{800} = 0.2125$$

- Probability that the two judges agreed by chance

$$P(E) = P(\text{nonrelevant})^2 + P(\text{relevant})^2 = 0.2125^2 + 0.7878^2 = 0.665$$

- Kappa statistic

$$K = \frac{(P(A) - P(E))}{(1 - P(E))} = \frac{(0.925 - 0.665)}{(1 - 0.665)} = 0.776$$

- A human is not a device that reliably reports a gold standard judgment of relevance of a document to a query. Rather, humans and their relevance judgments are quite idiosyncratic and variable. But this is not a problem to be solved : in the final analysis, the success of an IR system depends on how good it is at satisfying the needs of these idiosyncratic humans, one information need at a time.
- Nevertheless, it is interesting to consider and measure how much agreement between judges there is on relevance judgments. In the social sciences, a common measure for agreement between judges is the *kappa statistic*. It is designed for categorical judgments and corrects a simple agreement rate for the rate of chance agreement.

$$\text{Kappa} = \frac{P(A) - P(E)}{1 - P(E)}$$

5.6 System Quality and User Utility

Formal evaluation measures are at some distance from our ultimate interest in measures of human utility: how satisfied is each user with the results the system gives for each information need that they pose? The standard way to measure human satisfaction is by various kinds of user studies. These might include quantitative measures, both objective, such as time to complete a task, as well as subjective, such as a score for satisfaction with the search engine, and qualitative measures, such as user comments on the search interface. In this section we will touch on other system aspects that allow quantitative evaluation and the issue of user utility.

5.7 System Issues

- There are many practical benchmarks on which to rate an information retrieval system beyond its retrieval quality. These include :
  - How fast does it index, that is, how many documents per hour does it index for a certain distribution over document lengths?
  - How fast does it search, that is, what is its latency as a function of index size?
  - How expressive is its query language? How fast is it on complex queries?

  - How large is its document collection, in terms of the number of documents or the collection having information distributed across a broad range of topics?

All these criteria apart from query language expressiveness are straightforwardly *measurable*: we can quantify the speed or size. Various kinds of feature checklists can make query language expressiveness semi-precise.

5.8 Refining a Deployed System

- If an IR system has been built and is being used by a large number of users, the system's builders can evaluate possible changes by deploying variant versions of the system and recording measures that are indicative of user satisfaction with one variant vs. others as they are being used. This method is frequently used by web search engines.
- The most common version of this is *A/B testing*, a term borrowed from the advertising industry. For such a test, precisely one thing is changed between the current system and a proposed system, and a small proportion of traffic (say, 1-10% of users) is randomly directed to the variant system, while most users use the current system. For example, if we wish to investigate a change to the ranking algorithm, we redirect a random sample of users to a variant system and evaluate measures such as the frequency with which people click on the top result, or any result on the first page.
- The basis of A/B testing is running a bunch of single variable tests (either in sequence or in parallel): for each test only one parameter is varied from the control (the current live system). It is therefore easy to see whether varying each parameter has a positive or negative effect. Such testing of a live system can easily and cheaply gauge the effect of a change on users, and, with a large enough user base, it is practical to measure even very small positive and negative effects. In principle, more analytic power can be achieved by varying multiple things at once in an uncorrelated (random) way, and doing standard multivariate statistical analysis, such as multiple linear regression.
- In practice, though, A/B testing is widely used, because A/B tests are easy to deploy, easy to understand, and easy to explain to management.

Review Questions

Q. 1 Write note on Information retrieval system evaluation

Q. 2 What is precision and recall

Q. 3 What is evaluation of unranked retrieval sets

Q. 4 What is evaluation of ranked retrieval results

Q. 5 How to assess and justify the concept of relevance

Q. 6 Write short note on

a) System quality and user utility

b) System issues.

c) Refining a deployed system

□□□

6 Applications of Information Retrieval Systems

Unit - VI

Syllabus

Introduction to Multimedia Information Retrieval

Introduction to Distributed Information

6.1 Distributed Information Retrieval - Introduction

- Distributed computing is the application of multiple computers connected by a network to solve a single problem.
- A distributed computing system can be viewed as a MIMD parallel processor with a relatively slow inter-processor communication channel and the freedom to employ a heterogeneous collection of processors in the system.
- In fact, a single processing node in the distributed system could be a parallel computer in its own right. Moreover, if they all support the same public interface and protocol for invoking their services, the computers in the system may be owned and operated by different parties.
- Distributed systems typically consist of a set of server processes, each running on a separate processing node, and a designated broker process responsible for accepting client requests, distributing the requests to the servers, collecting intermediate results from the servers, and combining the intermediate results into a final result for the client.
- This computation model is very similar to the MIMD parallel processing model but the difference here is that :
  1. The subtasks run on different computers and the communication between the subtasks is performed using a network protocol such as TCP/IP rather than the shared memory-based inter-process communication mechanisms.
  2. In a distributed system it is more common to employ a procedure for selecting a subset of the distributed servers for processing a particular request rather than broadcasting every request to every server in the system.
- Applications that lend themselves well to a distributed implementation usually involve computation and data that can be split into coarse-grained operations with relatively little communication required between the operations. Parallel information retrieval based on document partitioning fits this profile well.
- Moreover, documents are always grouped into collections, either for administrative purposes or to combine related documents into a single source. Collections, therefore, provide a natural granularity for distributing data across servers and partitioning the computation. Note that since term partitioning imposes greater communication overhead during query processing, it is rarely employed in a distributed system.
- To build a distributed IR system, we need to consider both engineering issues common to many distributed systems and algorithmic issues specific to information retrieval.

- The critical engineering issues involve defining a search protocol for transmitting requests and results; designing a server that can efficiently accept a request, initiate a sub process or thread to service the request, and exploit any locality inherent in the processing using appropriate caching techniques; and designing a broker that can submit asynchronous search requests to multiple servers in parallel and combine the intermediate results into a final and user response.
- The algorithmic issues include how to distribute documents across the distributed search servers, how to select which servers should receive a particular search request, and how to combine the results from the different servers.
- The search protocol specifies the syntax and semantics of messages transmitted between clients and servers, the sequence of messages required to establish a connection and carry out a search operation, and the underlying transport mechanism for sending messages (e.g., TCP/IP).
- At a minimum, the protocol should allow a client to:
  - Obtain information about a search server, e.g., a list of databases available for searching at the server and possibly statistics associated with the databases.
  - Submit a search request for one or more databases using a well defined query language.
  - Receive search results in a well defined format.
  - Retrieve items identified in the search results.
- For closed systems consisting of homogeneous search servers, a custom search protocol may be most appropriate, particularly if special functionality (e.g., encryption of requests and results) is required.
- Alternatively, a standard protocol may be used, allowing the system to interoperate more easily with other search servers.
- The Z39.50 standard for client/server information retrieval defines a widely used protocol with enough functionality to support most search applications.
- Another proposed protocol for distributed, heterogeneous search, called STARTS (Stanford Proposal for Internet Meta Searching), was developed at Stanford University in cooperation with a consortium of search product and service vendors.
- STARTS was designed from scratch to support distributed information retrieval and includes features intended to solve the algorithmic issues related to distributed Information Retrieval, such as merging results from heterogeneous sources.

6.2 Collection Partitioning

- The procedure used to assign documents to search servers in a distributed IR system depends on a number of factors. First, we must consider whether or not the system is centrally administered.
- In a system comprising independently administered, heterogeneous search servers, the distributed document collections will be built and maintained independently.
- In this case, there is no central control of the document partitioning procedure and the question of how to partition the documents is essentially debatable. It may be the case, however, that each independent search server is focused on a particular subject area, resulting in a semantic partitioning of the documents into distributed collections focused on particular subject areas. This situation is common in Meta search systems that provide centralized access to a variety of back-end search service providers.
- When the distributed system is centrally administered, more options are available.
  - **The first option** is simple replication of the collection across all of the search servers. This is appropriate when the collection is small enough to fit on a single search server, but high availability and query processing throughput are required. In this scenario, the parallelism in the system is being exploited via multitasking and the broker's job is to route queries to the search servers and balance the loads on the servers.

- Indexing the documents is handled in one of the following two ways.
  1. In the first method, each search server separately indexes its replica of the documents.
  2. In the second method, each server is assigned a mutually exclusive subset of documents to index and the index subsets are replicated across the search servers. A merge of the subsets is required at each search server to create the final indexes.
- In either case, document updates and deletions must be broadcast to all servers in the system. Document additions may be broadcast, or they may be batched and partitioned depending on their frequency and how quickly updates must be reflected by the system.
- **The second option** is random distribution of the documents. This is appropriate when a large document collection must be distributed for performance reasons but the documents will always be viewed and searched as if they are part of a single, logical collection. The broker broadcasts every query to all of the search servers and combines the results for the user.
- **The final option** is explicit semantic partitioning of the documents. Here the documents are either already organized into semantically meaningful collections, such as by technical discipline, or an automatic clustering or categorization procedure is used to partition the documents into subject specific collections.

6.3 Source Selection

- Source selection is the process of determining which of the distributed document collections are most likely to contain relevant documents for the current query, and therefore should receive the query for processing.
- One approach is to always assume that every collection is equally likely to contain relevant documents and simply broadcast the query to all collections.
- This approach is appropriate when documents are randomly partitioned or there is significant semantic overlap between the collections.
- When document collections are partitioned into semantically meaningful collections or it is prohibitively expensive to search every collection every time, the collections can be ranked according to their likelihood of containing relevant documents.
- The basic technique is to treat each collection as if it were a single large document, index the collections, and evaluate the query against the collections to produce a ranked listing of collections. We can apply a standard cosine similarity measure using a query vector and collection vectors.
- To calculate a term weight in the collection vector using tf-idf style weighting term frequency tfi,j is the total number of occurrences of term i in collection j, and the inverse document frequency idfi for term i is log(N/ni), where N is the total number of collections and ni is the number of collections in which term i appears.
- Problem of this approach is that although a particular collection may receive a high query relevance score, there may not be individual documents within the collection that receive a high query relevance score, essentially resulting in a false drop and unnecessary work to score the collection.
- To avoid this problem Moffat and Zobel proposed a solution by indexing each collection as a series of blocks, where each block contains B documents.
- When B equals 1, this is equivalent to indexing all of the documents as a single, monolithic collection.
- When B equals the number of documents in each collection, this is equivalent to the original solution. By varying B, a tradeoff is made between collection index size and likelihood of false drops.
- An alternative to searching a collection index was proposed by Voorhees using training queries to build a content model for the distributed collections. When a new query is submitted to the system, its similarity to the training queries is computed and the content model is used to determine which collections should be searched and how many hits from each collection should be returned.

6.4 Query Processing

- Query processing in a distributed information retrieval system is proceeds as follows :
  1. Select collections to search.
  2. Distribute query to selected collections.
  3. Evaluate query at distributed collections in parallel.
  4. Combine results from distributed collections into final result.
- As described in the previous section, Step 1 may be eliminated if the query is always broadcast to every document collection in the system. Otherwise, one of the previously described selection algorithms is used and the query is distributed to the selected collections.
- Each of the participating search servers then evaluates the query on the selected collections using its own local search algorithm. Finally, the results are merged.
- There are a number of scenarios as follows :
  1. If the query is Boolean and the search servers return Boolean result sets, all of the sets are simply unioned to create the final result set.
  2. If the query involves free-text ranking, a number of techniques are available ranging from simple/naive to complex/accurate.
- The simplest approach is to combine the ranked hit-lists using round robin interleaving. This is likely to produce poor quality results since hits from irrelevant collections are given status equal to that of hits from highly relevant collections. An improvement on this process is to merge the hit-lists based on relevance score.
- As with the parallel process described for Document Partitioning, unless proper global term statistics are used to compute the document scores, we may get incorrect results. If documents are randomly distributed such that global term statistics are consistent across all of the distributed collections, the merging based on relevance score is sufficient to maintain retrieval effectiveness.
- If the distributed document collections are semantically partitioned or maintained by independent parties, then reranking must be performed.
- Callan proposes reranking documents by weighting document scores based on their collection similarity computed during the source selection step. The weight for a collection is computed as :

$$\omega = 1 + |C| \cdot (s - \bar{s}) / \bar{s}$$

Where,

$|C|$ - The number of collections searched,

s - The collection's score,

$\bar{s}$ - The mean of the collection scores.

- The most accurate technique for merging ranked hit-lists is to use accurate global term statistics. This can be accomplished in one of a variety of ways one of this is discussed in following paragraph.
- If the collections have been indexed for source selection, that index will contain global term statistics across all of the distributed collections. The broker can include these statistics in the query when it distributes the query to the remote search servers. The servers can then account for these statistics in their processing and produce relevance scores that can be merged directly.
- If a collection index is unavailable, query distribution can proceed in two rounds of communication. In the first round, the broker distributes the query and gathers collection statistics from each of the search servers. These statistics are combined by the broker and distributed back to the search servers in the second round.

- Finally, the search protocol can require that search servers return global query term statistics and per-document query term statistics. The broker is then free to rerank every document using the query term statistics and a ranking algorithm of its choice.
- The end result is a hit-list that contains documents from the distributed collections ranked in the same order as if all of the documents had been indexed in a single collection.

6.5 Web Issues

Challenges

1. **Dynamically Evolving and Expanding Data**
   - Today the data is in order of peta-bytes size.
   - Everyday its size is increasing in multi folds.
   - This scenario is a great challenge for Web based IR.
2. **Highly Relevant Results**
   - Users need highly effective and relevant result for the queries.
   - Queries vary in their keywords and languages.
3. **Fast Response Time**
   - Queries are expected to be processed in very short time.
   - Results need to be compiled after retrieval in short time.
4. **Scalability**

   The retrieval system should be ready to scale as per requirement
5. **Compiling Results**
   - Results from various sources and on various systems needs to be handled, to generate
   - A single list of results for the user. Huge number of duplicates in different lists is a challenge.

6.6 Multimedia Information Retrieval - Introduction

- Multimedia information system is widely recognized as one of the most promising and growing field in the area of information management as it is rapidly used in many application environment like offices, CAD/CAM.
- The most important characteristic of multimedia information systems is the variety of data it supports. Multimedia system should have the capability to store, retrieve, transport and present data with very heterogeneous characteristics such as text, images, graphs and sound.
- And because of these reasons the development of multimedia system is more complex than traditional information system.

6.6.1 Multimedia Information System vs. Traditional System

- The data model, the query language, the access and storage mechanisms of a Multimedia information system supports object with very complex structure. While traditional system or conventional system deals with simple data type such as strings or integers.
- Multimedia system handles multimedia data while traditional system handles textual unstructured data. Traditional systems are unable to support the mix of unstructured and structured data and different kinds of media.

- Traditional system does not support metadata information such as that provided by data base schema which is fundamental component in a database management system. A multimedia information retrieval system requires some form of database schema because multimedia applications need to structure their data at least partially.
- Multimedia information retrieval system requires handling metadata which is crucial for data retrieval. Whereas traditional information retrieval system don't have such requirement.
- Traditional system handles attribute bases queries i.e. set of attributes. But multimedia information retrieval system answer attribute based as well as content based queries i.e. set of features.
- In traditional system object retrieved by query processing are exact and precise. While in multimedia information retrieval system exact match cannot be applied means there is no guarantee that the object retrieved by this type of predicate are 100% correct or precise.
- Queries in traditional type system are 'Retrieve all names having Ids between EMP10 to EMP100'. While in multimedia information retrieval system queries are of the type 'Retrieve all the cars manufactured by same company and with different colour'.

6.6.2 Data Modelling

- To ensure fast retrieval a multimedia information retrieval system should be able to represent and store multimedia objects.
- The system should be therefore able to deal with different kinds of media and with semi-structured data i.e. data whose structure may not match or only partially match, the structure prescribed by the data schema.
- The system must extract some feature from multimedia objects to represent semi-structured data.

6.6.3 Data Retrieval

- The main goal of a multimedia information retrieval system is to efficiently perform retrieval based on user requests by exploiting not only data attributes but also content of multimedia objects.
- Data retrieval depends on the following steps as shown in Fig. 6.6.1.

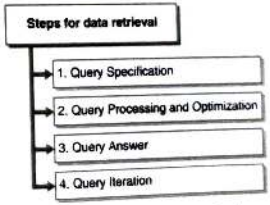


Fig. 6.6.1 : Steps for data retrieval

1. **Query Specification**
   - In this step, the user specifies the request. The query interface should allow the user to express.
   - Fuzzy predicates for proximity searches for example find all images similar to a car.
   - Content based predicates for example find multimedia objects containing an apple.
   - Conventional predicates on the object attributes for example conditions on the attribute 'color' of an image such as 'find all red images'.
   - Structural predicates for example find all multimedia objects containing a video clip.

2. **Query Processing and Optimization**
   - Like traditional system in multimedia information retrieval system query is parsed, compiled and optimized using best evaluation plan to generate internal representation. Due to the fuzzy terms, content-based predicates and structural predicates, query processing is a very complex activity. As compare with traditional and spatial databases a very little work is done on query processing strategies for multimedia information system.
   - The main problem is the heterogeneity of data: different query processing strategies one for each data type should be combined together in some way.
3. **Query Answer**
   - The retrieved objects are returned to the user in decreasing order of relevance.
   - Relevance is the distance function from the query object to the stored object.
4. **Query Iteration**
   - In traditional system when the system returns answer to the user the query process ends.
   - While in multimedia information retrieval system due to the inevitable lack of precision in the user request the query execution is iterated until the user is satisfied.
   - At each iteration, the user supplies the system with additional information by which the request is refined reducing or increasing the number of returned answers.

6.7 Data Modelling in Multimedia Information Retrieval

The integration of multimedia data in a traditional DBMS is not an easy task due to :

- Multimedia data is inherently different from conventional data. The main difference is that information about the content of multimedia data are usually not encoded into attributes provided by the data schema.
- As text, image, video and audio data are typically unstructured; specific methods are needed to identify and represent content features and semantic structures of multimedia data.
- Multimedia system requires large storage; one single image requires several Kbytes of storage where as single second of video requires several Mbytes of storage.
- The content of multimedia data is difficult to analyze and compare in order to be actively used during query processing.
- In order to resolve data modelling issues the framework of multimedia information retrieval systems entails two main tasks :
  1. A data model should be defined for the user to specify the data to be stored into the system. Such data model should be able to give integrated support for both conventional and multimedia data types. It also able to provide methods to analyze, retrieve and query such data.
  2. The system should provide model for the internal representation of multimedia data.
- As far as the first task is concerned, a promising technology with respect to the modelling requirements of multimedia data is the object oriented.
- The richness of data model provided by OODBMSs makes them more suitable than relational DBMSs for modelling both multimedia data types and their semantic relationships.
- The concept of class can be naturally used to define adhoc data types for multimedia data in that a class is characterized by both a set of attributes and a set of operations that can be performed on these attributes.
- As the classes are related into inheritance hierarchies it helps to allow the definition of a multimedia class as specialization of one or more super classes.

- In terms of storage techniques, query processing and transaction management the performance of OODBMS is not comparable to that of relational DBMS.
- One more drawback of OODBMS is that it is non standard.
- Even though a standard has been defined by Object Database Management Group a very few system support it.
- Because of the above mentioned reasons a lot of work has been devoted to the extension of the relational model with capabilities for modelling complex objects.
- The goal of object relational technology is to extend the relational model with ability to represent complex data types and to maintain the performance and simplicity of relational DBMS and related query languages.
- The possibility of defining abstract data types inside the relational model allows one to define ad hoc data types for multimedia data. And such data types provide support for content-dependant queries.
- The second problem which is related with data modelling is how to represent multimedia data inside the system.
- It is not sufficient to describe multimedia data through a set of attributes but some information should be extracted from the objects and used during query processing due to particular nature of multimedia data.
- The extracted information is typically represented as a set of features; each multimedia object is therefore internally represented as a list of features, each of which represents a point in a multidimensional space.
- Multi-attribute access methods can then be used to index and search for them. Features can be assigned to multimedia objects either manually by the user or automatically by the system.
- In general a hybrid approach is used by which the system determines some of the values and the user corrects or augments them.
- In both the cases, values assigned to some specific features such as the shape of an image or the style of an audio object are assigned to the object by comparing the object with some previously classified objects.
- To establish whether an image represents a car or a house the shape of the image is compared with the shapes of already classified cars and houses before taking a decision.
- A weight is usually assigned to each feature value representing the uncertainty of assigning such a value to that feature. For example if we are 80% sure that a shape is a square we can store this value together with the recognized shape.
- It follows that data modelling in multimedia information retrieval system is must take into account the complex structure of data and the need of representing features extracted from multimedia objects.
- Next we will give an overview of the support for multimedia data provided by commercial DBS then an example of the data model developed in the context of the MULTOS project.

6.7.1 Multimedia Data Support in Commercial DBMS

- Most of the current relational DBMSs support variable length data types which are useful to represent multimedia data. But the way by which data is supported is not standard.
- Each DBMSs vendor uses different names for such data types and provides support for different operations.
- For example -
- The Oracle DBMS provide the VARCHAR2 data type to represent variable length character strings with maximum length of data is 4000 bytes.
- The RAW and LONG RAW data types are used for data that is not interpreted by Oracle. These data types can be used to store sounds, graphics or unstructured objects.
- LOB (Large Object) data types can be used to store large unstructured data objects up to 4 GB in size.
- BLOB (Binary Object) are used to store unstructured binary large objects.
- CLOB (Character Large Object) are used to store character large object data.

- The Sybase SQL server supports IMAGE and TEXT data types to store images and unstructured text and also provides a limited set of functions for searching and manipulation.
- But the support provided by the above mentioned data type is very limited and operations can be performed on such data by means of the built in functions provided by the DBMS are very simple.
- The major improvement have been provided by SQL3 with respect to its predecessor SQL-92 like :
  - Support for an extensible type system. Extensibility is achieved by providing constructs to define user-dependent abstract data types like object oriented manner.
  - Each type specification consists of both attribute and function specifications.
  - A strong encapsulation is provided in that attribute values can only be accessed by using some system functions.
  - User defined functions can be either visible from any object or only visible in the object they refer to.
  - Both single and multiple inheritances can be defined among user-defined types and dynamic late binding is provided.
  - Provides three types of collections data types as: sets, multisets and lists.
  - Several system defined operations are provided to deal with collections.
  - Provides a restricted form of object identifier that supports sharing and avoid data duplication.

Examples of SQL3

Most of the commercial products have already implemented their proprietary versions of SQL3. For example :

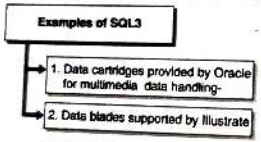


Fig. 6.7.1 : Examples of SQL3

1. **Data cartridges provided by Oracle for multimedia data handling**
   - Oracle provide data cartridge for text, spatial data, image, audio and video data.
   - Oracle8 provides a ConText cartridge which is text management solution combining data management capabilities of traditional system with advanced retrieval and natural-language process technology.
   - ConText supports the document formats like ASCII, MSWord and HTML.
   - ConText has ability to find document of specific type.
2. **Data blades supported by Illustrate**
   - Provides support for 3D and 2D spatial data blades for modelling spatial data.
   - Provides boxes, vectors, quadrangles data types
   - Provides INTERSECT, CONTAINS, OVERLAPS, CENTER operations.
   - Implements R-tree for performing efficient spatial queries.
   - Supports a data blade which can be used to query images by content.
   - The object relational technology and its extensive type system are now widely used in industrial and research projects for example LA Scala. The goal of this project is the development of the multimedia archive of Teatro alla Scala.

6.7.2 MULTOS Data Model

- MULTOS i.e. MULTimedia office server is a multimedia document server with advanced document retrieval capabilities.
- Developed in the context of an ESPRIT project in the area of office systems.
- Based on client-server architecture.
- Supports different type of servers i.e. current servers, dynamic servers and archive servers. Each supported server differs in storage capacity and document retrieval speed.
- Server's supports filling and retrieval of multimedia objects based on document collections, document types, document attributes, document text and document images.
- Allows the representation of high level concepts present in the documents contained in the database, the grouping of documents into classes of documents having similar content and structure and the expression of conditions on free text.
- Each document is described by MULTOS data model.

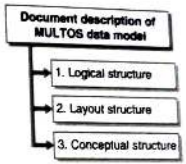


Fig. 6.7.2 : Document description of MULTOS Data model

1. **Logical structure**
   - Determines arrangement of logical components for example title, introduction, chapter, section.
   - Allows syntax oriented description of the document content.
2. **Layout structure**
   - Deals with the layout of the document content.
   - Contains components such as pages, frames etc.
   - Allows syntax oriented description of the document content.
3. **Conceptual structure**
   - Allows semantic oriented description of the document content as opposed to the syntax oriented description provided by the logical and layout structure.
   - Added to provide support for document retrieval by content.
   - For document retrieval it plays the role of the database schema which enables the use of efficient access structure.
   - Are basis for formulating queries at an abstract level.
   - MULTOS provides a formal model based on a data structuring tool available in semantic data models, to define the document conceptual structure.
   - The logical and layout structures are defined according to the ODA document representations.
   - Documents having similar conceptual structures are grouped into conceptual types.

- Conceptual types are maintained in a hierarchy of generalization to handle types. Subtype inherits from its super types the conceptual structure and then refines it.
- Types can be strong and weak. A strong type completely specifies the structure of its instances.
- While weak type partially specifies the structure of its instances. Components of unspecified type called as spring component type can also appear in document definition.
- The conceptual structure of the type Generic Letter is as shown in Fig. 6.7.3.

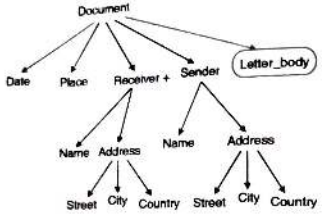


Fig. 6.7.3 : Conceptual Type Structure of Generic_Letter

- The node Letter_Body is a spring conceptual component.
- The complete conceptual structure corresponds to the type Business_Product_Letter is as shown in Fig. 6.7.4

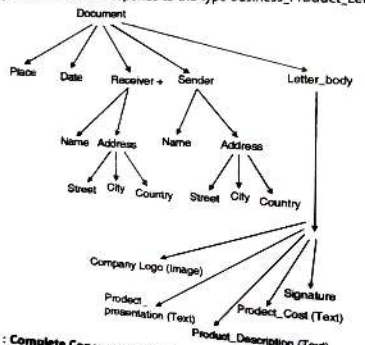


Fig. 6.7.4 : Complete Conceptual Structure of the Type Business_Poduct_Letter

- This type has been obtained from Generic_Letter by specialization of Letter_Body into a complex conceptual component, defined as an aggregation of five conceptual components.
- According to the conceptual model the document type Business_Product_Letter is linked to the document type Generic_Letter by an is-a relationship.
- In this example the '+' symbol attached to the Receiver component means that it is mutivalued. The Name and the Address appear in two subtrees having as roots the conceptual components Receiver and Sender.

- To deal with image data MULTOS provides sophisticated approach-
  - An image analysis process is performed, consisting of two phases low level image analysis and high level image analysis.
  - Two types of index are constructed- object index and cluster index.

a. Low level image analysis

During this phase the basic objects composing a given image and their relative positions are identified.

b. High level image analysis

- This phase deals with image interpretation according to the Dempster-Shafter theory evidence.
- At the end of image analysis process images are described in terms of the objects recognized with associated belief and possibility values and the classes to which they belong.
- The information is then exploited in image access. Image access information is stored in an image header associated with the image file. Access structures are then built for a fast access to image headers.

c. Object index

- For each object a list is maintained.
- Each element of the list is a pair BI, IMH. Where, BI is the associated belief interval representing the probability that image considered really contains the object. And IMH is a pointer o the header of the image containing the object.

d. Cluster index

- For each image class a list of pairs MF, IMH is maintained.
- IMH is pointer to an image header corresponding to an image with a non null degree membership to the class.
- MF is the value of the membership degree.
- Membership degree of an image to a given class is computed by comparing the image interpretation resulting from the analysis phase with the class description.

6.8 Techniques to Represent Audio and Visual Document

- Multimedia stands for applications that handle different types of digital data originating from distinct types of media.
- The most common types of media in multimedia applications are text, sound, images, and video (which is an animated sequence of images) as well as binary data.
- The digital data originating from each of these four types of media is quite distinct in volume, format, and processing requirements (for instance, video and audio impose real time constraints on their processing).
- Different types of formats are necessary for storing each type of media.
- In this section we cover formats for multimedia applications. In contrast with text formats, most formats for multimedia are partially binary and hence can only be processed by a computer.
- Also, the presentation style is almost completely defined, perhaps with the exception of some spatial or temporal attributes.

6.8.1 Format for Images

- There are several formats for images. The simplest formats are direct representations of a bit-mapped (or pixel-based) display such as XBM, BMP, or PCX. But those formats consume too much space.

- For example, a typical computer screen which uses 256 colours for each pixel might require more than 1 Mb (one megabyte) in storage just for describing the content of a single screen frame.
- In practice, images have a lot of redundancy, and can be compressed efficiently. So, most popular image formats incorporate compression such as Compuserve's Graphic Interchange Format (GIF).
- GIF is good for black and white pictures, as well as pictures that have a small number of colours or gray levels (say 256).
- To improve compression ratios for higher resolutions, lossy compression was developed. That is, uncompressing a compressed image does not give the original. This is done by the Joint Photographic Experts Group (JPEG) format which tries to eliminate parts of the image that have less impact on the human eye. This format is parametric, in the sense that the loss can be tuned.
- Another common image format is the Tagged Image File Format (TIFF). This format is used to exchange documents between different applications and different computer platforms. TIFF has fields for metadata and also supports compression as well as different numbers of colors.
- Yet another format is Truevision Targa image file (TGA), which is associated with video game boards.
- There are many more image formats, many of them associated to particular applications ranging from fax (bi-level image formats such as JBIG) to fingerprints (highly accurate and compressed formats such as WSQ) and satellite images (large resolution and full-color images).
- In 1996 a new bit-mapped image format was proposed for the Internet : Portable Network Graphics (PNG).

6.8.2 Format for Audio

- In order to store properly Audio must be digitalized.
- The most common formats for small pieces of digital audio are AU, MIDI, and WAVE.
- MIDI is a standard format to interchange music between electronic instruments and computers.
- For audio libraries other formats are used such as RealAudio or CD formats.

6.8.3 Format for Video

- There are several formats for animations or moving images (similar to video or TV), but here we mention only the most popular ones.
- The main one is MPEG (Moving Pictures Expert Group) which is related to JPEG. MPEG works by coding the changes with respect to a base image which is given at fixed intervals. In this way, MPEG profits from the temporal image redundancy that any video has. Higher quality is achieved by using more frames and higher resolution. MPEG specifies different compression levels, but usually not all the applications support all of them. This format also includes the audio signal associated with the video.
- Other video formats are AVI, FLI, and QuickTime. AVI may include compression (CinePac), as well as QuickTime which was developed by Apple. As for MPEG, audio is also included.

6.9 Query Languages

- In relational or object database systems queries are based on the exact match mechanism.
- With this mechanism system will be able to return only those tuples or objects that satisfy some well specified criteria given in the query expressions i.e. query predicates specify which values the object attributes must contain.
- User should be able to query the content of multimedia objects by specifying values of semantic attributes and also able to specify additional conditions about the content o multimedia data.
- Thus exact match is only one of the possible ways of querying multimedia objects.

- A similarity based approach is applied that considers both the structure and the content of the objects.
- A query of the later type is called as content based queries. These queries retrieve multimedia object depending upon their global content.
- Information on the global content of an object is not represented as attribute values in the database.
- Instead of that a set of information called as features is extracted and maintained for each object.
- When the user submit query, the features of the query object are matched with respect to the feature of the object stored in the database and only the objects that are more similar to the query are returned to the user.
- While designing any multimedia query language following three aspects should be considered.
- Interfaces to be provided to the user to enter his/her queries.
- Conditions on multimedia objects can be specified in the user request.
- The impact of uncertainty, proximity and weights on the design of query language.

6.9.1 Request Specification

To query multimedia objects, two different interfaces are presented to the user.

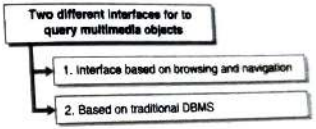


Fig. 6.9.1 : Interfaces for to query multimedia objects

1. Interface based on browsing and navigation

- Users can browse and navigate inside the structure of multimedia objects to locate the desired object. This approach is used in CAD/CAM/CASE environment due to the complex structure of the objects under consideration.
- But navigation is not always the best way to find multimedia objects because they may be heavily time consuming when the object desired is deeply nested.

2. Based on traditional DBMS

- Specify the conditions that objects of interest must satisfy.
- Queries can be specified in two different ways :

i. Traditional database context.

User can enter query by using a specific query language.

ii. Query by example

- When there is image and audio data then this way is preferred. In this approach queries are specified by using actual data inside a visual environment. The user provides the system with an object example that is then used to retrieve all the stored objects similar to the given one.
- This approach requires the use of GUI environment where user can pick examples and compose the query objects. To pick examples, the system must supply some domain i.e. for each object feature a set of typical values.
- Example of this type is "Retrieve all houses of the same shape and different color".

6.9.2 Conditions on Multimedia Data

- Multimedia query language should provide predicates for expressing conditions on the attribute, the content and the structure of multimedia objects.
- Query predicate can be classified as shown in Fig. 6.9.2.

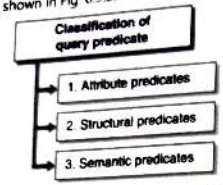


Fig. 6.9.2 : Classification of query predicate

1. Attribute predicates

- Concern the structured content attribute of the multimedia objects.
- These are predicates against traditional attribute, i.e. attribute for which an exact value is supplied for each object.
- Examples of attribute are the speaker of an audio object, the size of an object and the type of an object.
- By querying these predicates the system applies exact-match retrieval.

2. Structural predicates

- Concern the structure of data being considered.
- Predicate are answered by using some form of metadata and information about the database schema.
- Example of this type is "Find all multimedia objects containing at least one image and a video clip".

3. Semantic predicates

- Concern the semantic content of the queried data depending on the features that have been extracted and stored for each multimedia object.
- Example of this type is "Find all the object containing the word OFFICE."
- The query "Find all the blue houses" is a query on the image content. This query can be executed only if color and shape are features that have been previously extracted from the image.
- Current systems support semantic predicates only with respect to specific features, such as the color, the shape, the texture and sometimes the motion.
- The main difference between attribute predicates and semantic predicates is that in semantic predicate an exact match cannot be applied. I.e. there is no guarantee that object retrieved are 100% correct or precise. The result of a query involving semantic predicates is a set of objects which has an associated degree of relevance with respect to the query.

Properties of multimedia objects

The structural and semantic predicates can also refer to following two properties of multimedia objects :

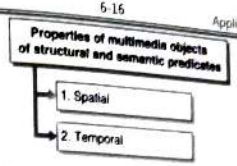


Fig. 6.9.3 : Properties of multimedia objects

1. Spatial

- Spatial semantic predicates specify conditions about the relative positions of a set of objects in an image or a video.
- Examples of spatial semantic predicates are : contain, intersect, is contained in, is adjacent to.
- Spatial structural predicates are used to specify spatial layout properties for the presentation of multimedia objects.
- Spatial structural predicates are also used to impose a condition on the spatial layout of the retrieved objects.
- Example of spatial structural predicate is query of the form "Find all the object containing an image overlapping the associated text.

2. Temporal

- Temporal semantic predicates are related to continuous media like audio and video.
- Temporal semantic predicates all to express temporal relationship among the various frames of a single audio or video.
- Example of temporal semantic predicates is "find all the objects that contain an audio component where the hint of the discussion is first policy, and then economy"
- Temporal structural predicates are used to specify temporal synchronization properties for the presentation of multimedia objects.
- Example of structural temporal predicate is query of form "Find all the objects in which a jingle is played for the duration of an image display".
- The temporal and spatial predicates can be combined to express more articulated requirements. Example is the query of the form "find all the objects in which the logo of a car company is displayed and when it disappears, a graphics showing the increase in the company sales is shown in the same position where the logo was."

6.9.3 Uncertainty, Proximity and Weights in Query Expressions

One of the aspects in designing a query language is how to specify the degree of relevance of the retrieved objects. And this can be done by -

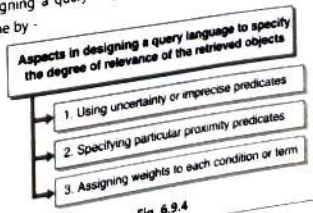


Fig. 6.9.4

1. **Using uncertainty or imprecise predicates**
   - Some imprecise terms and predicates such as normal, unacceptable, typical are used.
   - Each of these terms represents a set of possible acceptable values and not the precise values. And these possible acceptable values are with respect to which the attribute or the features has to be matched
2. **Specifying particular proximity predicates**
   - The predicate does not represent a precise relationship between objects and values or between attribute/feature and values. Instead of that the relationship is based on the computation of a semantic distance between the query object and the stored objects on the basis of the extracted features.
   - Example of proximity predicate is nearest object search, in this the user requests all the objects which are closest or within a certain distance of a given object.
3. **Assigning weights to each condition or term**
   - Weight specifies the degree of precision by which a condition must be verified by an object.
   - Example of this type is query of the form "Find all the objects containing an image representing a screen and a keyboard"
   - The use of imprecise terms and relationship, the use of weights allows the user to drive the similarity-based selection of relevant objects.
   - The corresponding query is executed by assigning some importance and preference values to each predicate and term. Then the object are retrieved and presented to the user as an ordered list.
   - This ordering is given by a score associated with each object, giving a measure of the matching degree between the object and the query. And the computation of score is based on probabilistic models using the preference values assigned to each predicate.

6.9.4 Query Languages to Support Retrieval of Multimedia Objects

- In this section we will see some query languages supporting retrieval of multimedia objects.
- There are two query languages for retrieval of multimedia objects as shown in Fig. 6.9.5.

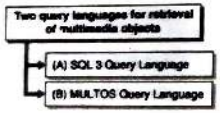


Fig. 6.9.5 : Query language for retrieval

6.9.4 (A) SQL 3 Query Language

- In this section we will see the facilities provided by the standard SQL 3 to support retrieval of multimedia objects.
- Due to the extensibility to deal with complex objects make SQL 3 suitable for modelling multimedia data.
- From the query language point of view there are major improvements of SQL 3 with respect to SQL - 92 as :

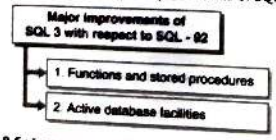


Fig. 6.9.6 : Improvement of SQL3 with respect to SQL-92

1. **Functions and stored procedures**
   - SQL 3 allows the user to integrate external functionalities with data manipulation.
   - As SQL 3 allows functions of an external library to be introduced into database system as external functions.
   - Such external functions can be either implemented by using an external language or directly implemented by SQL 3.
   - If external function is implemented by external language then SQL 3 only specifies which the language is and where the function can be found.
   - By this way the impedance mismatch between two different programming languages and type systems is avoided.
   - But this approach requires an extension of SQL with imperative programming languages constructs.
2. **Active database facilities**
   - Database are able to react to some system or user-dependent events by executing specific actions, using active rules supported by SQL 3.
   - Active rules or triggers are very useful to enforce integrity constraints.
   - The ability to deal with external functions and user-defined data types enables the language to deal with objects with a complex structure as multimedia objects.
   - Without this characteristic, it is not possible to deal with BLOB (Binary Large OBject) as it reduces the view of multimedia data to single large uninterested data values, which are not a liquate for the rich semantics of multimedia data.
   - Spatial and temporal constraints can be enforced by using triggers. And it also preserves the database consistency.
   - SQL 3 also allows to model multimedia objects in the framework of a well understood technology.

Limitations of SQL 3

- Does not provide retrieval support and optimization.
- NO information retrieval techniques are integrated into the SQL 3 query processor.
- The ability to perform consent-based search is application dependent.
- Objects are not ranked and therefore are returned the application as a unique set.
- Specialized indexing techniques can be used but they are not transparent to the user.

6.9.4 (B) MULTOS Query Language

- The development of the MULTOS query language has been driven by then requirements such that it should be possible to easily navigate through the document structure using path-names.
- Path name identifies either one component or several components.
- If path name identifies only one component then it is termed as total.
- If path name identifies more than one component then it is termed as partial.
- Path names are similar to object oriented path expressions but queries both on the content and on document structure must be supported.
- Query predicates on complex components must be supported. In this case, the predicate applies to all the document subcomponents that have a type compatible with the type required by the query.
- This possibility is very useful when a user does not recall the structure of a complex component.

- In general the form of MULTOS query is as follows :

  FIND DOCUMENTS VERSION version-clause

  SCOPE scope - clause

  TYPE type - clause

  WHERE condition - clause

  WITH component

  Where,

- The version-clause specifies which versions of the documents should be considered by the query.
- The scope-clause restricts the query to a particular set of documents.
- This set of documents is either a user-defined document collection or a set of documents retrieved by a previous query.
- The type-clause allows the restriction of a query to documents belonging to prespecified set of types.
- The condition expressed by the condition clause is applicable to the document belonging to these types and their subtypes.
- When no type is specified, the query is applied to all the document types.
- The condition-clause is a Boolean combination of simple conditions i.e. predicates on documents components.
- Predicates are expressed on conceptual components of documents conceptual components are referenced by pet-names the general form of a predicate is – component restriction.
- Where, component is a path-name and restriction is an operator followed by an expression.
- The with-clause allows one to express structural predicates.
- Component is a path-name and the clause looks for all documents structuring containing such a component.
- Different types to conditions can be specified in order to query different types of media.
- MULTOS supports three main classes of predicates as shown in Fig. 6.9.7.

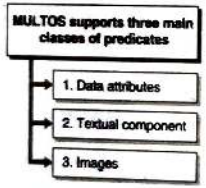


Fig. 6.9.7 : Classes of predicate

1. **Data attributes**

   An exact match search is performed.

2. **Textual component**

   By determining all objects containing some specific strings.

3. **Images**

   - By specifying conditions on the image content.
   - Image predicates allow one to specify conditions on the class to which an image should belong or conditions on the existence of a specified object within an image and on the number of occurrences of an object within an image.
   - The following example illustrates the basic features of the MULTOS query language.

Example

- Here we will consider the conceptual structure Generic-letter.
- Consider the following query.

```
FIND DOCUMENT VERSIONS LAST WHERE
    Document . Date > 1/1/1998 AND
    * Product_Description CONTAINS
    ( * Sender . Name = "Olivetti" OR
    * Product_presentation CONTAINS "Olivetti") AND
    "Personal Computer"AND
    (* Address . Country = "Italy" OR
        TEXT CONTAINS "Italy") AND
        WITH * Company_Logo.
```

- According to this query user search for the last version of all documents dated after January 1998. Containing a company logo, having the word "olivetti" either as sender name or in the product presentation (is textural component) with the word 'Personal computer' in the product description section (is another textual component) and with the word 'Italy' either constituting the country in the address or contained in any part of the entire document.
- '*' symbol indicates that the path-name is not complete, that is it could identify more than one component.
- The query language provided by MULTOS also supports the specifications of imprecise queries that can be used when the user has an uncertain knowledge about the consent of the documents he or she is seeking.
- By associating both a preference and an importance value with the attributes in the query uncertainty can be expressed. Such values are then used for ranking the retrieved documents.
- This is as discussed in following example.
- The query is as follows :

```
FIND DOCUMENT VERSION LAST WHERE
(Document . Date BETWEEN (12/31/1998, 1/31/1998) PREFERED
BETWEEN (2/1/1998, 2/15/98) ACCEPTABLE) HIGH AND
(* Sender . Name = "olivetti" OR
* product_Presentation CONTAINS "olvietti") HIGH AND
(* Product_Description CONTAINS "Personal computer") HIGH AND
(* Product_Description CONTAINS "good ergonomics") LOW AND
(* Address . Country = "Italy") OR
TEXT CONTAINS "Italy") HIGH AND
WITH * Company_logo HIGH
(IMAGE MATCHES
screen HIGH
keyboard HIGH
AT LEAST 2 floppy_drives LOW) HIGH
```

- This query finds the last versions of all documents written in January, but possibly even at the beginning of February 1998. Containing a company logo, having the word "Olivetti" either as sender name or in the product presentation with the word "personal computer" in the product description section and with the word "Italy" either constituting the country in the address or contained in any part of the entire document.
- Personal computers are described in the product description section as products having good ergonomics.
- The document should contain the picture of the personal computer. Complete with screen and keyboard, with at least two floppy drives.
- The value 'LOW' associated with the condition on 'good ergonomics' indicates that the user formulating the query is not completely sure about this description of PC but associated value is HIGH.

6.10 Multimedia Information Retrieval : Indexing and Searching

6.10.1 Introduction

- The searching methods will search a database of multimedia object to locate objects that match a query object exactly and so problem is how to design such fast searching methods.
- Object can be two-dimensional colour images gray scale medical images in 2D or 3D, one dimensional time series, digitized voice or music, video clip etc.
- Example of query by content is "in a collection of colour photograph : Find ones with the same colour distribution as a sunset photograph."
- Some specific applications include image databases (like financial, marketing, production time series), scientific databases with vector fields. Audio and video database, DNA databases. Such databases contains queries like, "find all the companies whose stock prices move similarly" or "Find medical X-rays that contain something that has the texture of a tumor."
- Searching for similar patterns in such kind of databases is essential as it helps in predictions. Computer-aided medical diagnosis and teaching, hypothesis testing, data mining and rule discovery.
- The distance of two objects has to be quantified and domain expert will supply such distance function D ( ).
- If two object O_1 and O_2 are given, then the distance i.e. dissimilarity of the two objects is given by D (O_1, O_2)
- For example if the objects are two equal-length time series, then the distance D ( ) can be Euclidean distance i.e. the root of the sum of squared differences.
- Similarity queries can be classified into following two categories as shown in Fig. 6.10.1.

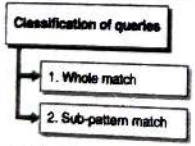


Fig. 6.10.1 : Classification of queries

1. Whole match

- If the collection of N objects O1, O2, ..., ON and a query Q is given we have to find those objects that are within distance ϵ from Q.
- In this match query and objects are of the same type.
- For example is the objects are 512 × 512 gray-sale images then only we can have query of whole match type.

2. Sub-pattern match

- The query is allowed to specify only part of the object.
- Given N data objects O1, O2, ..., ON and a query (sub) object Q and a tolerances ϵ, we want to identify the parts of the object that match the query.
- For example if the objects are 512 × 512 gray-sale images like medical x-rays then the query could be 48 × 48 sub pattern e.g. a typical x-ray of tumour.

Additional types of queries includes

- Nearest neighbours queries :

  E.g. find the five most similar stocks to IBM's stock.
- All pair queries or spatial joins :

  E.g. Report all the pairs of stocks that are within distance ϵ from each other.

The ideal method should fulfill the following requirement for all the types of queries

- As sequential scanning and distance calculation with each and every object will be too slow for large databases so it should be fast.
- It should be correct that means it should return all the qualifying objects, without missing any 'False alarm is' are acceptable as they can be discarded easily through a post processing step but try to keep the number low so that the total response time is minimized.
- It should require a small space overhead.
- The method should be dynamic. If method is dynamic then it is easy to insert, delete and update the objects.

6.10.2 Spatial Access Methods

- In 'GEMINI' approaches if feature extraction functions are used to map objects into points in f-dimensional space.
- To accelerate the search this we can use highly fine-tuned database spatial access methods.
- The spatial access methods form three classes :
  1. R* - trees and the R - tree family
  2. Linear quad trees
  3. Grid - files
- Several of these methods explode exponentially with the dimensionality eventually reducing to sequential scanning.
- For linear quad trees, the effort is proportional to the hyper surface of the grows exponentially with the dimensionality.
- Grid files also face the similar problem as they a directory that grows exponentially with the dimensionality.
- Grid files also face the similar problem as they require a directory that grows exponentially with the dimensionality.
- The R-tree based method seems to be more robust for higher dimensions, provided that the famous of the R-tree node remain greater than two.
- As one of the typical representatives of spatial access method in following section we discuss the R-tree method and its variants.

R-trees

- Represents a spatial object by its Minimum Bounding Rectangle (MBR).
- Parent nodes are formed by grouping data rectangles and parent nodes are recursively grouped to form grandparent nodes. And eventually a tree is formed.

- The MBR of parent node completely contains the MBR's of its children. And MBR are allowed to overlap. Modes of the tree correspond to disk space.
- Disk pages or disk blocks are consecutive byte positions on the surface of the disk that are typically fetched with one disk access.
- The goal of the insertion split and deletion routines is to give trees that will have good clustering with few light parents MBR.
- Fig. 6.10.2 illustrates data rectangles organized in an R-tree with fanout 3.

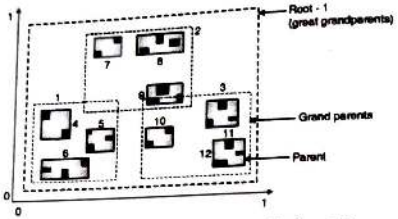


Fig. 6.10.2 : Organization of R-tree with fanout 3

- Fig. 6.10.3 shows the file structure for the same R-tree where nodes corresponds to disk pages.

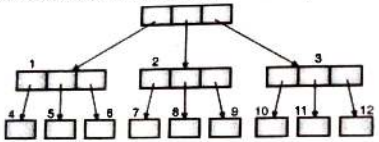


Fig. 6.10.3 : The file structure for the R-tree with fanout 3

- A range query specifies a region of interest requiring all the data regions that interest it.
- To answer this type of query perform following steps :
  1. First retrieve a superset of the qualifying data region.
  2. Compute the MBR of the data region.
  3. Then recursively descend the R-tree excluding the branches whose MBR do not intersect the query MBR.
- Thus the R-tree will give you quickly the data regions whose MBR intersects, the MBR of the query region.
- Then the retrieved data regions will be further examined for intersection with the query region.

6.11 Generic Multimedia Indexing Approach

For whole match queries the problem is defined as follows :

- We have collection of N objects : $O_1, O_2, ..., O_N$
- The distance or dissimilarity between two objects (O_i, O_j) is given by the function $D(O_i, O_j)$. This function is implemented as program.
- The user has to specify the query object Q and a tolerance ϵ.
- Here we have to find the objects in the collection of N objects $O_1, O_2, ..., O_N$ that are within distance ϵ from the query object.

- One solution to find such objects is to apply sequential scanning i.e. for each and every object O_i where $1 \le i \le N$ we have to compute its distance from Q and report the objects with distance $D(Q, O_i) \le \epsilon$.
- But the sequential scanning may be slow because of following reasons :
  1. The computation of distance might be expensive.
     For example the editing distance in DNA strings requires a dynamic programming algorithm which grows (in the range of hundreds or thousands) like the product of the string lengths.
  2. The data base size H may be huge.
- In order to avoid above mentioned disadvantages of sequential scanning we need faster alternative and that is nothing but GEMINI.
- The GEMINI i.e. GEneric Multimedia INdexIng approach is based on following two ideas :
  1. 'Quick and dirty' test is performed to discard quickly the vast majority of non qualifying objects.
  2. To achieve fasters than sequential searching use of spatial access method.
- We will illustrate the case with an example. Consider a database of time series, such as yearly stock price movements with one price per day.
- Assume that the distance function between two such series S and Q is the Euclidian distance and given as :

$$D(S, Q) = \left(\sum_i (S[i] - Q[i])^2\right)^{1/2}$$

Where,

S [i] : The value of stocks on the i^{th} day.

- In this example computing the distance of two stocks will take 365 subtractions and 365 squarings.
- In case of quick and dirty test we have to characterize a sequence with a single number.
- This characterization will help us to discard many non-qualifying sequences.
- And such a number could be the average stock price over the year, if we consider the above example.
- If two stocks differ in their average by a large margin then it is impossible that they will be similar.
- But the converse is not true and because of that we have the false alarms i.e. non-qualifying objects.
- With good feature (numbers that contain some information about a sequence is referred as features) we will have a quick test, which will discard many stocks with a single numerical comparison for each sequence.
- Using two or more features might be better if using one feature is good, because it will reduce the number of false alarms. But it will make quick and dirty test a bit more elaborate and expensive and because of that cost will be increased.
- In stock price example one feature is Eudidian distance and additional features can be standard deviation, Discrete Fourier Transform (DFT) coefficients.
- So finally we can map each object joint a point in f-dimensional space by using f features for each of objects. And this mapping is referred as F( ). (Here F stands for feature).
- Mapping F( ) is defined as - the mapping of objects to f-dimensional points i.e. F (0) is the f - D point that corresponds to object 0.
- This mapping helps to improve the second drawback of sequential scanning.
- We can do this by organizing f - D points into a spatial access method and clustering them in a hierarchical structure like the R* tree.
- So depending upon the query, we can prune out the large portion of the database that are not promising by exploiting the R* tree.
- Fig. 6.11.1 illustrates the basic idea. It shows a database of sequences of $S_1, S_2, ..., S_N$.

- It shows each sequence is mapped to point in feature space.
- And a query with tolerance ϵ becomes a sphere of radius ϵ.

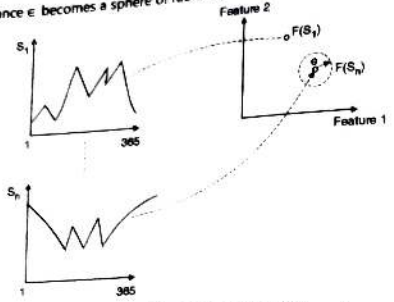


Fig. 6.11.1 : Illustration of basic idea

- Here the objects are mapped into 2D points. If we consider the stock price example then objects are time series that are 365 points long are mapped into 2D points and points are using the average and the standard deviation as features.
- Consider the 'whole match' query that requires all the objects that are similar to S_N within tolerance ϵ then this query becomes an f - D sphere in feature space. Centred on the image F (S_N) of S_N.
- Then the search algorithm for a 'whole match' query is as follows :
  1. In feature space map a query object Q into a point F (Q).
  2. Retrieve all the points which are within the desired tolerance by using a spatial access method.
  3. Then retrieve the corresponding objects and then compute their actual distance from Q and discard the false alarms.
- So the method has the potential to relieve both problems of the sequential scanning and result in faster searches.
- But we have to be careful at the time of mapping F () from objects to F - D points such that the distance should not get distorted.
- Let D () be the distance function of two objects and $D_{feature}$ ( ) be the Eudidian distance of the corresponding feature vector, then the mapping should preserve the distances exactly, in which case the spatial access method will have neither false alarms nor false dismissals.
- But requiring perfect distance preserve distance preservation might be difficult. Because for example, it is not obvious which features we have to use to match the editing distance between two DNA strings.
- Even if the features are obvious, there might be practical problems, for example in the stock price example, we could treat every sequence as a 365 - dimensional vector, although in theory a spatial access method can support an arbitrary number of dimensions and in practice they all suffer from the 'dimensionality curse'.
- If the distance in feature space matches or underestimates the distances between two objects then we can guarantee that there will be no false dismissals.
- If $O_1 \times O_2$ be two objects (e. g. same-length sequence) with distance function D ( ) (e.g. the Euclidean distance) and F (O_1), F(O_2) be their feature vectors (e.g. their first few Fourier Coefficients) with distance function $D_{feature}$ ( ) (e.g the Euclidean distance) the we have "To guarantee no false dismissals for whole-match queries the feature extraction function F ( ) should satisfy the following formula : $D_{feature}$ (F (O_1), F(O_2)) $\leq$ D (O_1, O_2)
- Lower-bounding the distance works correctly for range queries.

- Now question is whether it works for other queries like 'all pair' and 'nearest neighbour'.
- An 'all pair' query can easily be handled by a spatial join on the points of the feature space.
- For 'nearest neighbour' query the following algorithm is used to guarantee no false dismissals.
  1. Find the nearest neighbor that is F (P) to the query point F (Q).
  2. Issue a range query with query object Q and radius ϵ = D (Q, P) i.e. the actual distance between the query object Q and data object P.
- Finally the GEMINI approach to indexing multimedia objects for fast similarity searching is as follows
  1. Find the distance function D ( ) between two objects.
  2. To provide a 'quick and dirty' test find out one or more numerical feature extraction functions.
  3. To guarantee correctness, prove that the distance in feature space lower bounds the actual distance D ( ).
  4. To store and retrieve the f-D feature vectors use a spatial access method (e. g. an R-tree).
- To find the distance function D ( ) between two objects involves a domain expert.
- The methodology focuses on the speed of search but the quality of the results is completely depends on the distance function that the expert will provide.
- Finding numerical feature extraction function will require intuition and imagination.
- It starts by trying to answer the question called as 'feature-extraction question. The feature extracting questions can be of - "If we are allowed to use only a numerical feature to describe each data object, what should this feature be ?"
- The successful answer to the above question will provide two goals :
  1. They should facilitate step 3 i.e. the distance lower bounding.
  2. They should capture most of the characteristics of the objects.
- Thus, GEMINI will return exactly the same quality of output that would be returned by a sequential scanning of the database.
- And the GEMINI is faster than sequential scanning.

6.12 One-Dimensional Time Series

- We will give case studies of steps 2 and 3 of the GEMINI algorithm using
  1. One dimensional time series
  2. Two dimensional color images
- For each case study, first we will describe the objects and the distance function, and then we will show how to apply lower-bounding lemma and finally give experimental results on real or realistic data.
- In this section we will discuss the 1-D time series.
- In one dimensional time series, we have to find series which are similar to a desirable series from the collection of time series.
- For example "In a collection of yearly stock price of movements. Find the ones that are similar to IBM"

6.12.1 Distance Function

- The first step in the GEMINI algorithm is we have to find the distance measure between tow time series.
- And a typical distance functions the Euclidean distance which is of reply used in financial and forecasting application. Additional functions include time - warping.

6.12.2 Feature Extraction and Lower - Bounding

- After deciding the distance function D ( ) between two objects i.e. Euclidean distance function. Next we have to find some features that can lower bound it.
- We have to find out the set of features that will satisfy the two requirements :
  1. They should lower bound the distance.
  2. They should carry much information about the corresponding time series so that the false alarms are less.
- The above second requirement suggests that we use 'good' features that have much discriminatory power.
- In the stock price example a 'bad' feature is 'the first day's value' because two stocks might have similar first-day values though they may differ significantly from them.
- Even though if we use the values of all 365 days as features, this would perfectly match the actual distance but it would lead to the 'dimensionality curse' problem. So we need better features.
- In the second step of GEMINI algorithm, we have to ask feature - extracting question of the form If we are allowed to use only one feature from each sequence, what would this feature be ?. And answer to this type of question is the average.
- By the same token additional features could be the average of the first half, of the second half of the first quarter etc.
- That means we could use the coefficient of the Discrete Fourier Transform (DFT).
- For a signal $\vec{x} = [x_i]$, $i = 0, \ldots n - 1$, and x_F denote the n - point DFT coefficient at the F^{th} frequency ($F = 0, 1, \ldots n - 1$).
- In the third step of GEMINI algorithm we have to show that the distance in feature space lower-bounds the actual distance.
- This solution is provided by Parsevals theorem. It states that, the DFT preserves the energy of a signal as well as the distances between two signals.

$$D(\vec{x}, \vec{y}) = D(\vec{X}, \vec{Y})$$

Where,

$\vec{X} \times \vec{Y}$ are Fourier transforms of $\vec{x}$ and $\vec{y}$ resp.

- Thus by keeping the first f ($f \leq n$) coefficients of the DFT as the features we can lower bound the actual distance as :

$$D_{feature}(F(\vec{x}), F(\vec{y})) = \sum_{F=0}^{f-1} |X_F - Y_F|^2$$

$$\leq \sum_{F=0}^{n-1} |X_f - Y_F|^2$$

$$= \sum_{i=0}^{n-1} |x_i - y_i|^2$$

And finally :

$$D_{feature}(F(\vec{x}), F(\vec{y})) = D(\vec{x}, \vec{y})$$

Thus there will be no false dismissals.

- GEMINI algorithm can be applied with and orthonormal transform. Such as Discrete Cosine Transform (DCT), the wavelet transforms because they all preserve the distance between the original and the transformed space.

- The response time will improve with the ability of the transform to concentrate the energy, the fewer the coefficients that contain most of the energy, more accurate will be the estimates for the actual distance, the fewer the false alarms and faster the response time.
- Thus, better transforms will achieve the better response times.
- The DFT concentrates the energy in the first few coefficients for a large class of signals i.e. the colored noises. These signals have a skewed energy spectrum ($O(F^{-b})$) as follows
  1. For the value of b = 2 we will have random walks or brown noise. And such signals model successfully stock movements and exchange rates.
  2. With the value b > 2 i.e. more skewed spectrum we will have black noises. And such kind of signals models successfully, the water level of river and the rainfall patterns as they vary over time.
  3. With the b = 1 value we will have pink signals. Musical scores and other works of art, consists of pink noise. Whose energy spectrum follows $O(F^{-1})$.
- In general white noise with $O(F^0)$ energy spectrum is completely unpredictable, brown noise with $O(F^{-2})$ energy spectrum is too predictable and 'boring' and the energy the spectrum of pink noise lies in bean.
- Fig. 6.12.1 shows the movement of the exchange rate between the Swiss France and the US dollar starting 7^{th} August 1990 to 18^{th} April 1991 and first 3000 values out of 30,000.

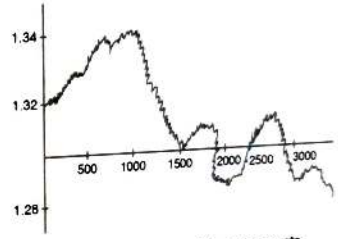


Fig. 6.12.1 : Movement of the exchange rate

- Fig. 6.12.2 shows the amplitude of the Fourier coefficients as a function of the frequency F, as well as the 1/F line in a logarithmic - logarithmic plot.
- As it is successfully modelled as a random walk, the amplitude of the Fourier coefficients follows the 1/F line.

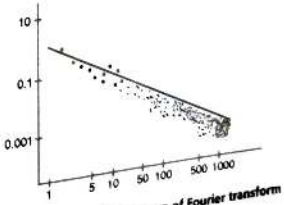


Fig. 6.12.2 : Amplitude spectrum of Fourier transform

- In addition to 1D signals, several families of real n - D signals belong to the family of colored noise, with skewed spectrum.

6.12.3 Experiments

- Performance results with the GEMINI approach on time series are collected in this section from the reference "Efficient similarity search in sequence database, by Rakesh Agrawal, Christos Faloutsos and Aram swami."
- In that the method is compared with the sequential scanning method.

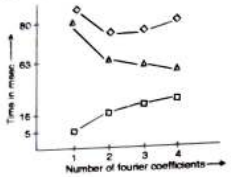


Fig. 6.12.3 : Breakup of the execution time for range query

- The R* - tree is used for the spatial access method within GEMINI.
- The sequences were artificially generated random walks, with length n = 1024 and number N varied from 50 to 400.
- Fig. 6.12.3 shows the break-up of the response time as a function of the number f of DFT coefficient.
- In the Fig. 6.12.3.

  ◊ (Diamond) – Total time

  Δ (Triangle) – Post - processing time

  (Square) – R* tree time.
- If we keep more features F_1 the R* tree becomes bigger and slower but more accurate. And if it is and shorter post processing time.
- This trade off reaches an equilibrium for f = 2 or 3. For the remaining experiments the f = 2 Fourier coefficients were kept for indexing, resulting in a four - dimensional R* tree.
- Fig. 6.12.4 shows the response time for the two methods GEMINI and sequential scan, as a function of the number of sequences H.

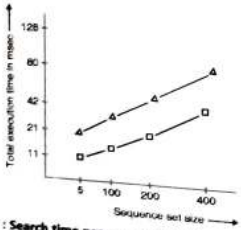


Fig. 6.12.4 : Search time per query VS number N of sequences

- In the Fig. 6.12.4 search time per query VS number N of sequences for whole match queries is shown. In the Fig. 6.12.4 the line with ◊ represents GEMINI approach and the line with Δ shows sequential scanning.

- And from the Fig. 6.12.4 it is clear that GEMINI approach outperforms the sequential scanning.
- From the application of GEMINI approach on time series. Following concretions are drawn.
  1. GEMINI approach can be successfully applied to the time series and specifically to the ones that behave like 'colored images' for example stock price movements, currency exchange rates, water level in rivers etc.
  2. For signals with skewed spectrum the minimum response time is achieved for a small number of Fourier coefficients i.e. f = 1, 2, 3.

     It means that a suboptimal choice for f will give search time that is close to the minimum.

     With the help of the lower bounding and the energy. Concentrating properties of the DFT we can avoid the 'dimensionality curse'.
  3. The success in 1D series suggests that GEMINI is promising for 2D or higher dimensionality signals. If those signals also have skewed spectrum.
  4. The success of JPEG images which uses DCT indicates that real images indeed have a skewed spectrum.
  5. The method is extended to handle sub pattern matching for time sequences.
- Assuming that query pattern have length of at least w we pre-process every sequence of the database by allowing a sliding window of length w at each and every possible position and by extracting the features f for a given positioning of the window.
- Thus, every sequence becomes a trail in the f - dimensional feature space which can be further approximated by a set of few MBR's that cover it. Representing each sequence by a few MBR's in feature space may allow false alarms but no false dismissals.
- The same approach can be generalized for sub pattern matching in 2D signals and in general for n-dimensional vector fields.

6.13 Two-Dimensional Colour Images

- GEMINI has also been applied for colour images within the QBIC (Query by Image Content) project of IBM. This project studies methods to query large online images databases using the images content as the basis of the queries.
- Examples of the content include colour, texture, shape position and dominant edges of image items and regions.
- In this section we will discuss methods on databases of still images with two main data types.
  1. Images or Scenes : A scene is a color image.
  2. Items : A item is a part of a scene.
- For example a person, a piece of outlined texture or an apple.
- Each scene has zero or more items.
- In this section we will give an overview of the indexing aspects of QBIC and specifically the distance functions and the application of the GEMINI approach.

6.13.1 Image Features and Distance Function

- Here we will focus on the color features, which can be resolves by the GEMINI approach.
- For color we compute a K-element color histogram for each item and scene where K = 256 or 64 colors.
- Each component in the color histogram is the percentage of lixers that are most similar to that.
- Fig. 6.13.1 gives an example of such histogram of a fictitious photograph of a sensed, there are many red, pink, orange and purple pixels but only few white and green ones.

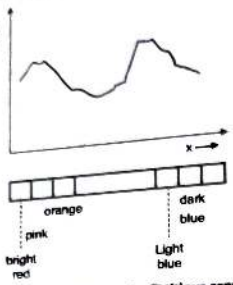


Fig. 6.13.1 : Example of a color histogram of a fictitious sensed photographs

- Once these histograms are computed. One method to measure the distance between two histograms (K + 1 vectors) $\vec{x}$ and $\vec{y}$ is given by

$$d^2_{hist}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^t A (\vec{x} - \vec{y})$$

$$d^2_{hist}(\vec{x}, \vec{y}) = \sum_{i}^{K} \sum_{j}^{K} a_{ij} (x_i - y_i)(x_j - y_j)$$

- Where the superscript t indicates matrix transposition and the color-to-color similarity matrix A has entries a_{ij} which describe the similarity between color i and color j.

6.13.2 Lower Bounding

- For applying the GEMINI method for color indexing there are two obstacles :
  1. Dimensionality curse
  2. Quadratic nature of the distance function.
- The distance function in the feature space is a full quadratic form which involves all cross terms.
- Such distance function is expensive to compute than a Euclidean distance and it also precludes efficient implementation of commonly used spatial access method.
- Fig. 6.13.2 shows the illustration of the 'cross-talk' between two color histogram.

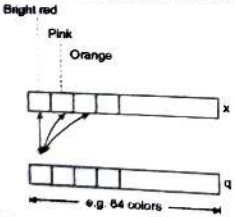


Fig. 6.13.2 : Cross-talk between two color histograms

- To compute the distance between two color histograms $\vec{x}$ and $\vec{q}$, the bright-red component of $\vec{x}$ has to be compared not only to the bright-red component of $\vec{q}$ but also to the pink and orange components of $\vec{q}$.
- To resolve the cross-talk problem, we will try to apply the GEMINI approach as follows.
  1. The first step of the algorithm is to calculate the distance function between two objects.

     We have done the distance function calculation between two color images and is given by the equation

$$D(\,) = d_{hist}(\,) = (\vec{x} - \vec{y})^t A (\vec{x} - \vec{y})$$

$$= \sum_{i}^{K} \sum_{j}^{K} a_{ij} (x_i - y_i)(x_j - y_j)$$

  2. The second step is to find one or more numerical features whose Euclidean question is like. If we are allowed to use only one numerical feature to describe each color image, what should this feature be ?

     Again we can consider some average value or the first few coefficients of the two - dimensional DFT transform. As we have three color components i.e. Red, Green and Blue we will consider the average amount of red, green and blue in a given color image. So color of an individual pixel is described by the tripled (R, G, B) i.e. 'R'ed , 'G'reen and 'B'lue.

- The average color vector of an image or item $\vec{x} = (R_{avg}, G_{avg}, B_{avg})^t$ is defined as

$$R_{avg} = (1/P) \sum_{p=1}^{P} R(p)$$

$$G_{avg} = (1/P) \sum_{p=1}^{P} G(p)$$

$$B_{avg} = (1/P) \sum_{p=1}^{P} B(p)$$

Where,

P → The number of pixels in the item

R(p), G(p), B(p) → Red, Green and Blue components and in the range 0-255 of the p^{th} pixel.

p → Pixels from 1 to P.

- Given the average colors $\vec{x}$ and $\vec{y}$ of two items, we can define $d_{avg}(\,)$ as the Euclidean distance between the three - dimensional average color vectors.

$$d^2_{avg}(\vec{x}, \vec{y}) = (\vec{x} - \vec{y})^t (\vec{x} - \vec{y})$$

  3. The third step of the GEMINI algorithm is to prove that the simplified distance $d_{avg}(\,)$ lower bounds the actual distance $d_{hist}(\,)$. And this D true because of the application Quadratic Distance Bounding Theorem.

     So given a color query retrieval is proceeded by first filtering the set of images based on their average (R, G, B) color and then doing a final more accurate matching using their full K-element histogram.

6.13.3 Experiments using Bounding Theorem

- In this section we will present experimental results with GEMINI on color using the bounding theorem.
- The experiment compares the relative performance between simple sequential evaluation of d_{hist} for all database vectors and GEMINI approach in terms of CPU time and disk accesses.

- The experiments performs the simulations on a database of N = 924 color image histograms each of K = 256 colors, of assorted natural images and report the total and CPU times required by the methods.
- Results are shown in Fig. 6.13.2.
- Fig. 6.13.3 presents the total response time as a function of the selectivity.

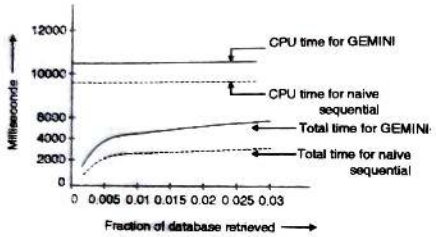


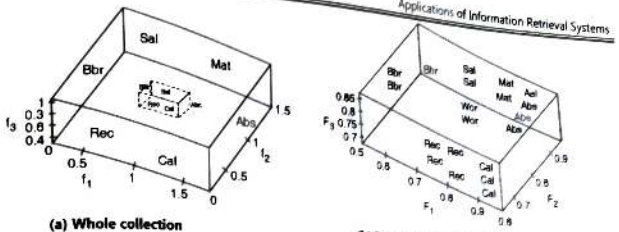
Fig. 6.13.3 : Response time VS selectivity for noise sequential and GEMINI

- Selectivity is the ratio of actual hits over the database size N.
- Fig. 6.13.3 also shows the CPU time for each method.
- So from the Fig. 6.13.3 it is clear that for a selectivity of 5% which would return = 50 images to the user, the GEMINI method is much faster than the sequential computation of the histogram distances i.e. GEMINI method requires from a fraction of second up to = 4 seconds while the naive sequential method requires consistently = 10 seconds.
- For larger databases, the naive method will have a linearly increasing response time.
- Thus, finally following conclusions are drawn

1. GEMINI approach motivated a fast method using the average RGB distance.
2. GEMINI approach motivated a strong theorem i.e. QDB theorem to guarantee the correctness.
3. GEMINI approach resolves the cross-talk problem.
4. GEMINI approach resolves the dimensionality curse problem with no extra cost. If requires only f = 3 features as opposed to K = 64 or 256 that d_{hist} ( ) required.

6.14 Automatic Feature Extraction

- GEMINI approach is useful for any setting that we can extract features from.
- There are different algorithms for automatic feature extraction methods like the Multi-Dimensional Scaling (MDS) and Fast Map.
- Extracting features allows facilitating the use of off-the-shelf spatial access methods and visual data mining.
- He can plot a 2D or 3D projection of the data set and inspect it for correlations, clusters and other patterns.
- Fig. 6.14.1 shows the results of Fast Map on 35 documents of seven classes, after deriving K = 3 features or dimensions.
- In the Fig. 6.14.1, the classes included are basketball reports (Bbr), abstracts of computer science technical reports (Abs), Cooking recipes, and so on.



(a) Whole collection **(b) Magnification of the dashed box**

Fig. 6.14.1 : Collection of documents after Fast Map in 3-D space

- The distance function was a decreasing function of the cosine similarity.
- Fig. 6.14.1 shows the 3D scatter-plot in its entirety (a) and (b) after zooming into the center to highlight the clustering abilities of Fast Map.
- In Fig. 6.14.1 the seven classes are separated in only K = 3 dimensions.

Review Questions

Q. 1 What is multimedia IR? Discuss steps on which data retrieval relies.

Q. 2 Explain collection partitioning and source selection with respect to distributed IR.

Q. 3 Discuss the generic multimedia index approach.

Q. 4 Discuss SQL and MULTOS Query language.

Q. 5 Explain Query specification and Processing.

Q. 6 Explain one dimensional time series.

Q. 7 Write a short note on :
i. MULTOS ii. GEMINI iii. SQL3 query language.

Q. 8 Explain distributed IR with the help collection partitioning, source selection and query processing.

Q. 9 Discuss technique to represent audio and visual documents.

Q. 10 Define multimedia IR? Explain steps of multimedia IR.

Q. 11 What is feature extraction in multimedia IR? How is it helpful for data retrieval.

Q. 12 Describe the architecture of distributed IR.

Q. 13 Describe Collection partitioning in distributed IR.

Q. 14 What do you mean by collection partitioning in Distributed IR ?

Q. 15 Explain source selection with respect to distributed IR.

Q. 16 Explain distributed IR with the help source selection.

Q. 17 Describe Source Selection in distributed IR.

Q. 18 Explain query processing with respect to distributed IR.

Q. 19 How are queries processed in distributed IR?