LECTURE NOTES

ON

INFORMATION RETRIEVAL SYSTEM

IV B. Tech I semester (JNTU)

UNIT-1

COMPUTER SCIENCE AND ENGINEERING

INTRODUCTION

1.1 Definition:

An Information Retrieval System is a system that is capable of storage, retrieval and maintenance of information.

→ Information in this context can be composed of following:

- **Text** (including numeric and date data)
- Images
- Audio
- Video
- Other multimedia objects

→ Of all the above data types, Text is the only data type that supports full functional processing.

→ The term "item**"** is used to represent the smallest complete unit that is processed and manipulated by the system.

→ The definition of item varies by how a specific source treats information. A complete document, such as a book, newspaper or magazine could be an item. At other times each chapter or article may be defined as an item.

→ The efficiency of an information system lies in its ability to minimize the overhead for a user to find the needed information.

→ Overhead from a user's perspective is the time required to find the information needed, excluding the time for actually reading the relevant data. Thus search composition, search execution, and reading non-relevant items are all
aspects of information retrieval overhead.

1.2 Objectives

The general objective of an Information Retrieval System is to minimize the overhead of a user locating needed information. Overhead can be expressed as
the time a user spends in all of the steps leading to reading an item containing the needed information. For example:

- Query Generation
- Query Execution
  - Scanning results of query to select items to read
- Reading non-relevant items etc ...

→ The success of an information system is very subjective, based upon what information is needed and the willingness of a user to accept overhead.

→ The two major measures commonly associated with information systems are precision and recall.

→Pr *ecision* $\frac{\textit{Number\_Re trieved\_Re levant}}{\textit{Number\_Total\_Re trieved}}$

Re *call* $\frac{\textit{Numbe\_Re trieved\_Re levant}}{\textit{Number\_Possible\_Re levant}}$

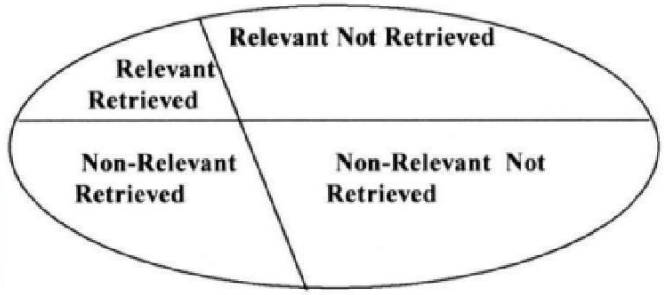
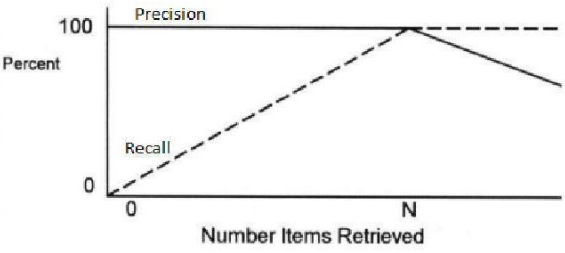


Figure 1.1 Effects of Search on Total Document Space



1.2a Ideal Precision and Recall

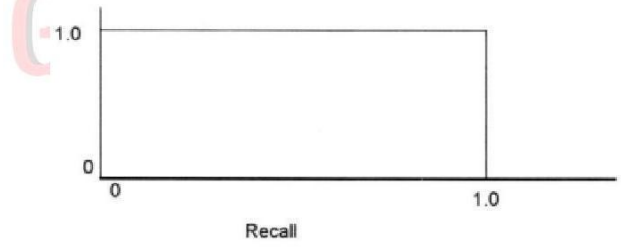


Figure 1.2b Ideal Precision/Recall Graph

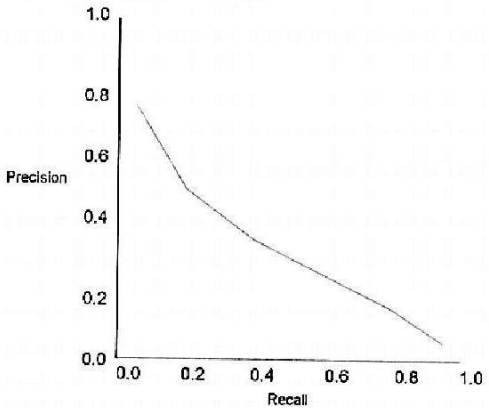


Figure 1.2c Achievable Precision/Recall Graph

1.3 Functional Overview:

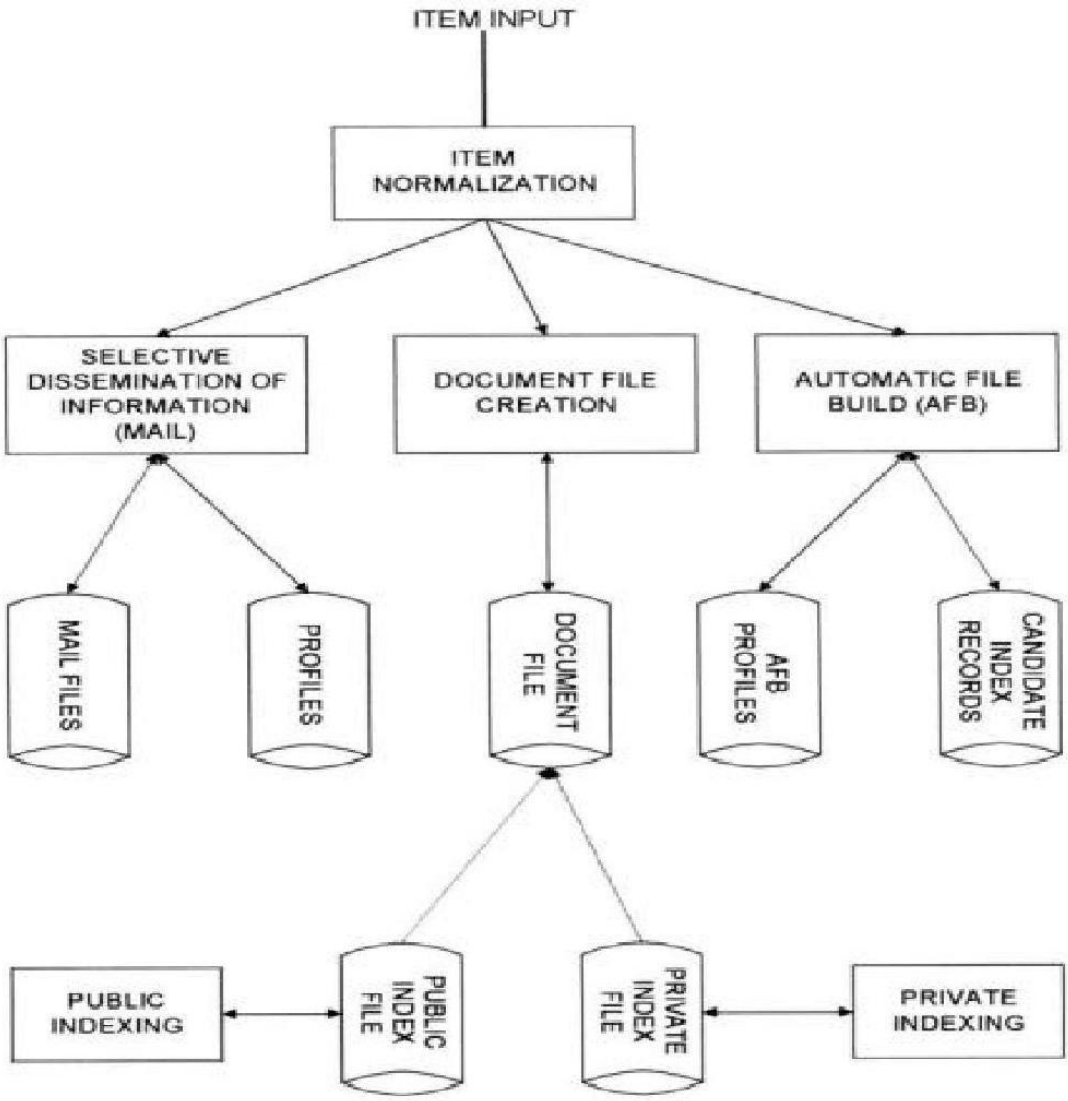


Figure 1.4 Total Information Retrieval System

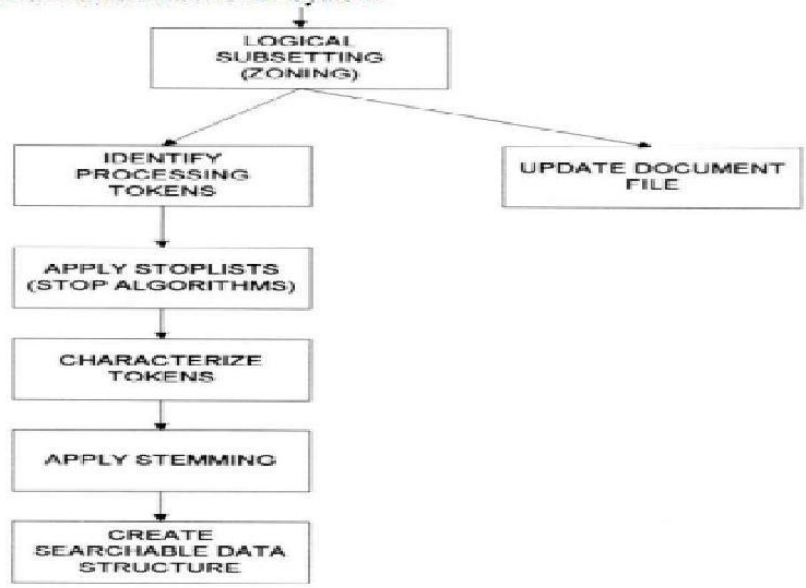


Figure 1.5 The Text Normalization Process

→ Item Normalization: The first step in any integrated system is to normalize the incoming items to a standard format. In addition to translating multiple external formats that might be received into a single consistent data structure that can be manipulated by the functional processes, item normalization provides logical restructuring of the item

1.4 Relationship to DBMS, Digital Libraries and Data Warehouses:

→ An Information Retrieval System is software that has the features and function required to manipulate "information" items where as a DBMS that is optimized to handle "structured" data.

→ All the three systems (IRS, Digital Libraries & Data Warehouses) are repositories of information and their primary goal is to satisfy user information needs.

→ The conversion of existing hardcopy text, images and analog data and the storage and retrieval of the digital version is a major concern to Digital Libraries, where as IRS concentrates on digitized data.

→ Digital Libraries value original sources where as these are of less concern to IRS.

→ With direct electronic access available to users the social aspects of congregating in a library and learning from librarians, friends and colleagues will be lost and new electronic collaboration equivalencies will come into existence.

→ With the proliferation of information available in electronic form, the role of search intermediary

will shift from an expert in search to being an expert in source analysis.

→ Information storage and retrieval technology has addressed a small subset of the issues

associated with digital

libraries.

→ The term Data Warehouse comes more from the commercial sector than academic sources.

→ Its goal is to provide to the decision makers the critical information to answer future direction questions.

→ A data warehouse consists of the following:

- Information Directory (describes the contents & their meaning)
- Input function (captures data and moves it to data warehouse)
- Data search & manipulation tools
- Delivery mechanism to export data

→ Data warehouse is more focused on structured data & decision support technologies.

→ Data mining is a search process that automatically analyzes data and extract relationships and dependencies that were not part of the database design.

1.5 Information Retrieval System Capabilities Search:

→ Major functions that are available in an Information Retrieval System are:

- Searching
- Browsing
- Miscellaneous

→ The objective of the search capability is to allow for a mapping between a user's specified need and the items in the information database that will answer that need.

Based upon the algorithms used in a system many different functions are associate with the system's understanding the search statement. The functions define the relationships between the terms in the search statement

- Boolean Logic: Allows a user to logically relate multiple Concepts together to define what information is needed
- Proximity: Used to restrict the distance allowed within an search between two terms. The semantic concept that the closer two terms found in a text are more likely they are related in the description of a particular concept.
- Continuous Word Phrases :(CWP) is two or more words that are treated as a single semantic unit.
- Fuzzy Searches: provide the capability to locate spellings of words that are similar to the entered search term. Primary used to compensate for errors in

spelling of words. Fuzzy searching increases recall at the expense of decreasing precision

- Term Masking: The ability to expand a query term by masking a portion of the term and accepting as valid any processing token that maps to the unmasked portions of the term

- Numeric and Date Ranges: Allows for specialized numeric or date range processing against those words.

- Concept/ Thesaurus Expansion : Thesaurus is typically a one-level or two-level expansion of a

  term to

  other terms that are similar in meaning where as a concept class is a tree structure that expands each meaning of a word into potential concepts that are related to the initial term.

→ Natural Language Queries : Allow a user to enter a prose statement that describes the information that the user wants to find. The longer the prose, the more accurate the results returned

→ Multimedia Queries : The user interface becomes far More complex with introduction of the availability
of multimedia items.

→ All of the previous discussions still apply for search of the textual portions of a multimedia database. But in addition, the user has to be able to specify search terms for the other modalities

1.6 Browse:

→ Once the search is complete, browse capabilities provide the user with the capability to determine which items are of interest and select those to be displayed.

→ There are two ways of displaying a summary of the items that are associated with a query: Line Item Status and Data Visualization.

→ Different browsing capabilities are:

- Ranking: Hits are retrieved in either a sorted order or in time order from the newest to the

oldest item. With the introduction of ranking based upon predicted relevance values, the status summary displays the relevance score associated with item along with a brief descriptor of the

- Zoning: The user wants to see the minimum information needed to determine if the item is relevant. Once the determination is made an item is possibly relevant, the user wants to display the complete item for detailed review
- Highlighting: Another display aid is an indication was selected. This indication frequently highlighting, lets the user quickly focus on the potentially relevant parts of the text to scan for item relevance. Different strengths of highlighting indicates how strongly the highlighted word participated in the selection of the item

1.7 Miscellaneous:

There are many additional functions that facilitates the user's ability to input queries, reducing the time it takes to generate the queries, and reducing a *priori* the probability of entering a poor query.

→ Different ways are:

→ Vocabulary Browse: This provides the capability to display in alphabetical sorted order words from the document database. Logically, all unique words (processing tokens) in the database are kept in sorted order along with a count of the number of unique items in which the word is found.

→ Iterative Search and Search History Log :Frequently a search returns a Hit file containing many more items than the user wants to review. Rather than typing in a complete new query, the results of the previous search can be used as a constraining list to create a new query that is applied against it.

→ Canned Query : The capability to name a query and store it to be retrieved and executed during a later user session is called canned or stored queries.

→ Multimedia : Once a list of potential items that Satisfy the query are discovered, the techniques for displaying them when they are multimedia Introduces new Challenges. To display more aggregate data textual interfaces sometimes allow for clustering of the hits and then use of graphical display to show a higher level view of the information. Neither of these techniques lends themselves well when the information is multimodal. The textual aspect of the multimedia can use to apply all of the techniques described.

UNIT-2

CATALOGING AND INDEXING

- INDEXING: the transformation from received item to searchable data structure is called indexing.
- Process can be manual or automatic.
- Creating a direct search in document data base or indirect search through index files.
- Concept based representation: instead of transforming the input into a searchable format some systems transform the input into different representation that is concept based .Search ? Search and return item as per the incoming items.
- History of indexing: shows the dependency of information processing capabilities on manual and then automatic processing systems .
- Indexing originally called cataloguing : oldest technique to identity the contents of items to assist in retrieval.
- Items overlap between full item indexing , public and private indexing of files
- Objectives : the public file indexer needs to consider the information needs of all users of library system . Items overlap between full item indexing , public and private indexing of files

•Users may use public index files as part of search criteria to increase recall.

•They can constrain there search by private index files

•The primary objective of representing the concepts within an item to facilitate users finding relevant information .

•Users may use public index files as part of search criteria to increase recall.

•They can constrain there search by private index files

•The primary objective of representing the concepts within an item to facilitate users finding relevant information .

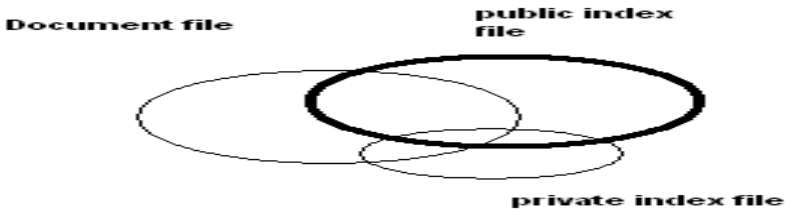
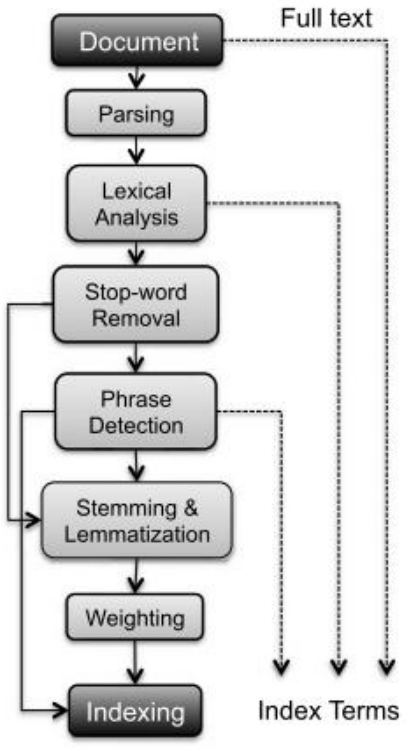


Fig : Indexing process

- 1.Decide the scope of indexing and the level of detail to be provided. Based on usage scenario of users .
- 2.Second decision is to link index terms together in a single index for a particular concept.

TEXT PROCESSING

Fig. 2.3 Text processing phases in an IR system



Text process phases

1. Document Parsing. Documents come in all sorts of languages, character sets, and formats; often, the same document may contain multiple languages or formats, e.g., a French email with Portuguese PDF attachments. Document parsing deals with the recognition and "breaking down" of the document structure into individual components. In this pre processing phase, unit documents are created; e.g., emails with attachments are split into one document representing the email and as many documents as there are attachments.

2. Lexical Analysis. After parsing, lexical analysis tokenizes a document, seen as an input stream, into words. Issues related to lexical analysis include the correct identification of accents, abbreviations, dates, and cases. The difficulty of this operation depends much on the language at hand: for example, the English language has neither diacritics nor cases, French has diacritics but no cases, German has both diacritics and cases. The recognition of abbreviations and, in particular, of time expressions would deserve a separate chapter due to its complexity and the extensive literature in the field For current approaches

3. Stop-Word Removal. A subsequent step optionally applied to the results of lexical analysis is stop-word removal, i.e., the removal of high-frequency words. For example, given the sentence "search engines are the most visible information retrieval applications" and a classic stop words set such as the one adopted by the Snowball stemmer,1 the effect of stop-word removal would be: "search engine most visible information retrieval applications".

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

5. Stemming and Lemmatization. Following phrase extraction, stemming and lemmatization aim at stripping down word suffixes in order to normalize the word. In particular, stemming is a heuristic process that "chops off" the ends of words in the hope of achieving the goal correctly most of the time; a classic rule based algorithm for this was devised by Porter [280]. According to the Porter stemmer, our example sentence "Search engines are the most visible information retrieval applications" would result in: "Search engin are the most visibl inform retriev applic".

- Lemmatization is a process that typically uses dictionaries and morphological analysis of words in order to return the base or dictionary form of a word, thereby collapsing its inflectional forms (see, e.g., [278]). For example, our sentence would result in "Search engine are the most visible information retrieval application" when lemmatized according to a WordNet-based lemmatizer

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

SCOPE OF INDEXING

- When perform the indexing manually, problems arise from two sources the author and the indexer the author and the indexer .
- Vocabulary domain may be different the author and the indexer.
- This results in different quality levels of indexing.
- The indexer must determine when to stop the indexing process.
- Two factors to decide on level to index the concept in a item.
- The exhaustively and how specific indexing is desired.
- Exhaustively of index is the extent to which the different concepts in the item are indexed.
- For example, if two sentences of a 10-page item on microprocessors discusses on-board caches, should this concept be indexed
- Specific relates to preciseness of index terms used in indexing.
- For example, whether the term "processor" or the term "microcomputer" or the term "Pentium" should be used in the index of an item is based upon the specificity decision.
- Indexing an item only on the most important concept in it and using general index terms yields low exhaustively and specificity.
- Another decision on indexing is what portion of an item to be indexed Simplest case is to limit the indexing to title and abstract(conceptual) zone .
- General indexing leads to loss of precision and recall.

PREORDINATION AND LINKAGES

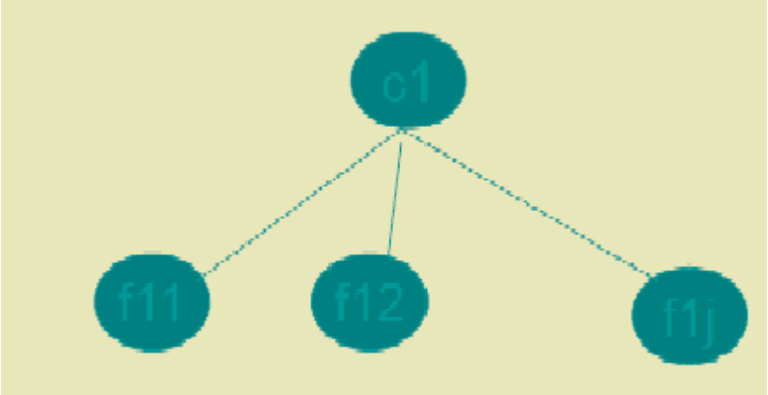
- Another decision on linkages process whether linkages are available between index terms for an item .
- Used to correlate attributes associated with concepts discussed in an item .'this process is called preordination .
- When index terms are not coordinated at index time the coordination occurs at search time. This is called post coordination , implementing by "AND" ing index terms .
- Factors that must be determined in linkage process are the number of terms that can be related.
- Ex., an item discusses 'the drilling of oil wells in Mexico by CITGO and the introduction of oil refineries in Peru by the U.S.'

AUTOMATIC INDEXING

- Case: Total document indexing .

- Automatic indexing requires few seconds based on the processor and complexity of algorithms to generate indexes.'
- Adv. is consistency in index term selection process.
- Index resulting form automated indexing fall into two classes , weighted and un weighted .
- Un weighted indexing system : the existence of an index term in a document and some times its word location are kept as part of searchable data structure .
- Weighted indexing system: a attempt is made to place a value on the index term associated with concept in the document . Based on the frequency of occurrence of the term in the item .
- Values are normalized between 0 and 1.
- The results are presented to the user in order of rank value from highest number to lowest number .
- Indexing By term
- Terms (vocabulary) of the original item are used as basis of index process .
- There are two major techniques for creation of index statistical and natural language.
- Statistical can be based upon vector models and probabilistic models with a special case being Bayesian model(accounting for uncertainty inherent in the model selection process) .
- Called statistical because their calculation of weights use information such as frequency of occurrence of words .
- Natural language also use some statistical information , but perform more complex parsing to define the final set of index concept.
- Other weighted systems discussed as vectorised information system .
- The system emphasizes weights as a foundation for information detection and stores these weights in a vector form.
- Each vector represents a document. And each position in a vector represent a unique word(*processing token*) in a data base..
- The value assigned to each position is the weight of that term in the document.
- 0 indicates that the word was not in the document .
- Search is accomplished by calculating the distance between the query vector and document vector.
- Bayesian approach: based on evidence reasoning(drawing conclusion from evidence)
- Could be applied as part of index term weighing. But usually applied as part of retrieval process by calculating *the relation ship between an item and specific query*.
- Graphic representation each node represents a random variable arch between the nodes represent a probabilistic dependencies between the node and its parents .
- Two level Bayesian network
- " c"" represents concept in a query

- "f" representing concepts in an item



- Another approach is natural language processing.
- DR-LINK(document retrieval through linguistics knowledge)
- Indexing by concept
- Concept indexing determines a canonical set of concept based upon a test set of terms and uses them as base for indexing all items. *Called latent semantics indexing* .
- Ex: match plus system developed by HNC inc
- Uses neural NW strength of the system word relationship (synonyms) and uses the information in generating context vectors.
- Two neural networks are used one to generated stem context vectors and another one to perform query.
- Interpretation is same as the weights.
- Multimedia indexing:
- Indexing video or images can be accomplished at raw data level.
- Positional and temporal (time) search can be done.

INFORMATION EXTRACTION

There are two processes associated with information extraction:

- 1.determination of facts to go into structured fields in a database and
- 2.extraction of text that can be used to summarize an item.
- The process of extracting facts to go into indexes is called Automatic File Build.
- In establishing metrics to compare information extraction, precision and recall are applied with slight modifications.

- Recall refers to how much information was extracted from an item versus how much should have been extracted from the item.
- It shows the amount of correct and relevant data extracted versus the correct and relevant data in the item.
- Precision refers to how much information was extracted accurately versus the total information extracted.
- Additional metrics used are over generation and fallout.
- Over generation measures the amount of irrelevant information that is extracted.
- This could be caused by templates filled on topics that are not intended to be extracted or slots that get filled with non-relevant data.
- Fallout measures how much a system assigns incorrect slot fillers as the number of
- These measures are applicable to both human and automated extraction processes.
- Another related information technology is document summarization.
- Rather than trying to determine specific facts, the goal of document summarization is to extract a summary of an item maintaining the most important ideas while significantly reducing the size.
- Examples of summaries that are often part of any item are titles, table of contents, and abstracts with the abstract being the closest.
- The abstract can be used to represent the item for search purposes or as a way for a user to determine the utility of an item without having to read the complete item.

DATA STRUCTURES

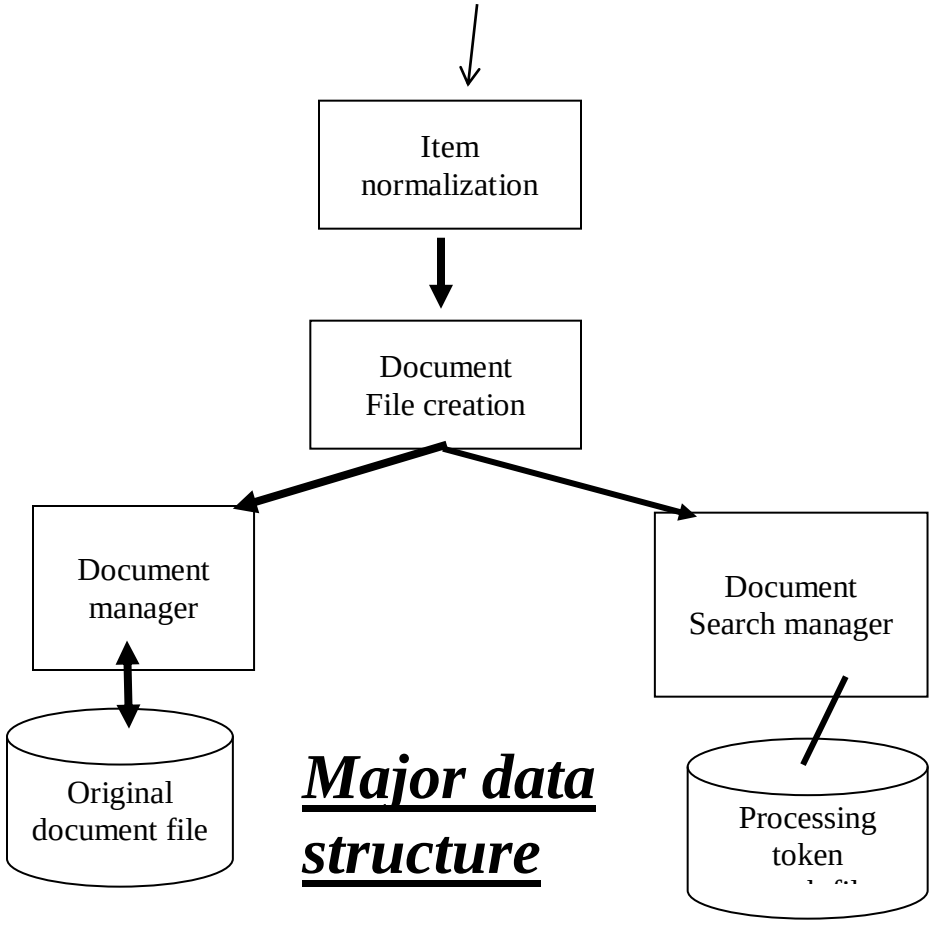
- Introduction to Data Structures
- Stemming Algorithms
- Inverted File Structure
- N-Gram Data Structure
- PAT Data Structure
- Signature File Structure
- Hypertext and XML Data Structures

- Data structure : The knowledge of data structure gives an insight into the capabilities available to the system .
- Each data structure has a set of associated capabilities .

1. Ability to represent the concept and their r/s.
2. Supports location of those *concepts Introduction*

- Two major data structures in any IRS:

1. One structurestores and manages received items in their normalized form is called document manger
2. The other data structure contains processing tokens and associated data to support search.



Result of a search are references to the items that satisfy the search statement which are passed to the document manager for retrieval.

Focus : on data structure that support search function

Stemming : is the transformation *often applied to data* before placing it in the *searchable data structure*

Stemming represents concept(word) to a canonical (authorized; recognized; accepted)morphological (the patterns of word formation in a particular language) representation .

Risk with stemming : concept discrimination information may be lost in the process. Causing decrease in performance.

Advantage : has a potential to increase recall.

STEMMING ALGORITHMS

- Stemming algorithm is used to improve the efficiency of IRS and improve recall.

- Conflation(the process or result of fusing items into one entity; fusion; amalgamation)is a term that is used to refer mapping multiple morphological variants to single representation(stem).
- Stem carries the meaning of the concept associated with the word and the affixes(ending) introduce subtle(slight) modification of the concept.
- Terms with a common stem will usually have similar meanings, for example:
- Ex : Terms with a common stem will usually have similar meanings, for example:
- CONNECT
- CONNECTED
- CONNECTING
- CONNECTION
- CONNECTIONS
- Frequently, the performance of an IR system will be improved if term groups such as this are conflated into a single term. This may be done by removal of the various suffixes -ED, -ING, -ION, IONS to leave the single term CONNECT
- In addition, the suffix stripping process will reduce the total number of terms in the IR system, and hence reduce the size and complexity of the data in the system, which is always advantageous.

❖ Major usage of stemming is to improve recall.

➢ Important for a system to categories a word prior to making the decision to stem.

➢ Proper names and acronyms (A word formed from the initial letters of a name say IARE …) should not have stemming applied.

➢ Stemming can also cause problems for natural language processing NPL systems by causing loss of information .

PORTER STEMMING ALGORITHM

- Based on a set condition of the stem
- A *consonant* in a word is a letter other than A, E, I, O or U, some important stem conditions are

1. The measure m of a stem is a function of sequence of vowels (V) followed by a sequence of consonant (C) .

   $C (VC)^{m}V$. m is number VC repeats

   The case m = 0 covers the null word.

   2. *<X> - stem ends with a letter X

   3.*v* - stem contains a vowel

   4. *d - stem ends in double consonant (e.g. -TT, -SS).

5. *o - stem ends in consonant vowel sequence where the final consonant is not w,x,y(e.g. -WIL, -HOP).

Suffix cond.s takes the form current _suffix = = pattern

Actions are in the form old_suffix ->. New_suffix

Rules are divided into steps to define the order for applying the rule.

Examples of the rules

Step	Condition	Suffix	Replacement	Example
1a	Null	Sses	Ss	Stresses -> stress
1b	*v*	Ing	Null	Making -> mak
1b1	Null	At	Ate	Inflated-> inflate
1c	*v*	Y	I	Happy->happi
2	m>0	aliti	al	Formaliti-> formal
3	m>0	Icate	Ic	Duplicate->duplie
4	m>1	Able	Null	Adjustable -> adjust
5a	m>1	e	Null	Inflate-> inflat
5b	m>1 and *d	Null	Single letter	Control -> control

2. Dictionary look up stemmers

- Use of dictionary look up.

- The original term or stemmed version of the term is looked up in a dictionary and replaced by the stem that best represents it.
- This technique has been implemented in INQUERY and Retrieval ware systems-
- INQUERY system uses the technique called Kstem.
- Kstem is a morphological analyzer that conflates words variants to a root form.
- It requires a word to be in the dictionary
- Kstem uses 6 major data files to control and limit the stemming process.

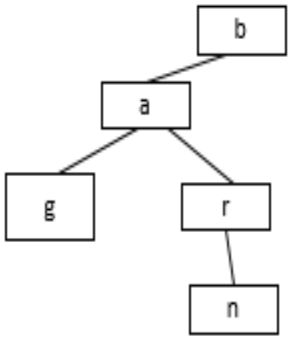
1. Dictionary of words (lexicon)
2. Supplemental list of words for dictionary
3. Exceptional list of words that should retain a 'e' at the end (e.g., "suites" to "suite" but "suited" to "suit").
4. Direct _conflation - word pairs that override stemming algorithm.
5. County_nationality _conflation (British maps to Britain)
6. Proper nouns -- that should not be stemmed

- New words that are not special forms (e.g., dates, phone numbers) are located in the dictionary to determine simpler forms by stripping off suffixes and respelling plurals as defined in the dictionary.

3. Successor stemmers:

- Based on length of prefixes .
- The smallest unit of speech that distinguishes on word from another
- The process uses successor varieties for a word .

Uses information to divide a word into segments and selects on of the segments to stem.



Symbol tree for term bag , barn

Successor variety of words are used to segment a word by applying one of the following four methods.

1. Cutoff method : a cut of value is selected to define the stem length.

2. Peak and plateau: a segment break is made after a character whose successor variety exceeds that of the character.
3. Complete word method: break on boundaries of complete words.
4. Entropy method:uses the distribution method of successor variety letters.

1. Let |Dak| be the number of words beginning with k length sequence of letters a.
2. Let |Dakj| be the number of words in Dak with successor j.
3. The probability that a member of Dak has the successor j is given as |Dakj| / |Dak|

The entropy of |Dak| is

$$Hak = \sum_{p=1}^{26} -(|Dakj| / |Dak|)(\log(|Dakj| / |Dak|))$$

After a word has been segmented the segment to be used as stem must be selected.

Hafer and Weiss selected the following rule

If (first segment occurs in <=12 words in database)

First segment is stem

Else (second segment is stem)

INVERTED FILE STRUCTURE

Inverted file structure

Most common data structure

Inverted file structures are composed of three files

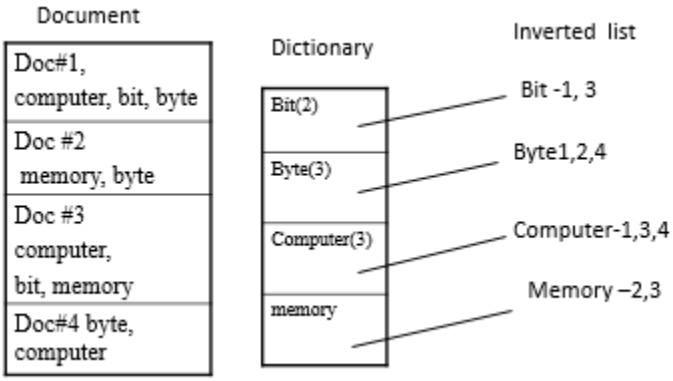
1. The document file
2. The inversion list (Posting List)
3. Dictionary

- The inverted file : based on the methodology of storing an inversion of documents.
- For each word a listof documents in which the word is found is stored(inversion of document)
- Each document is given a unique the numerical identifier that is stored in inversion list .

Dictionary is used to located the inversion list for a particular word.

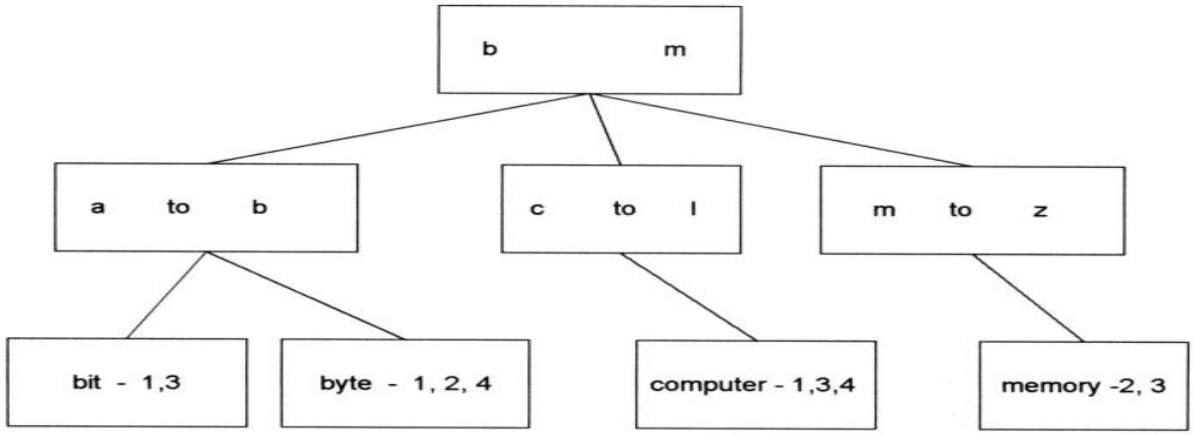
Which is a sorted list(processing tokens) in the system and a pointer to the location of its inversion list.

Dictionary can also store other information used in query optimization such as length of inversion lists to increase the precision.



- Use zoning to improve precision and Restrict entries.
- Inversion list consists of document identifier for each document in which the word is found.

  Ex: bit 1(10),1(12) 1(18) is in 10,12, 18 position of the word bit in the document #1.
- When a search is performed, the inversion lists for the terms in the query are locate and appropriate logic is applied between inversion lists.
- Weights can also be stored in the inversion list.
- Inversion list are used to store concept and their relationship.
- Words with special characteristics can be stored in their own dictionary. Ex: Date… which require date ranging and numbers.
- Systems that support ranking are re-organized in ranked order.
- B trees can also be used for inversion instead of dictionary.
- The inversion lists may be at the leaf level or referenced in higher level pointers.
- A B-tree of order m is defined as:
- A root node with between 2 and 2m keys
- All other internal nodes have between m and 2m keys
- All keys are kept in order from smaller to larger.
- All leaves are at the same level or differ by at most one level.



N-GRAM DATA STRUCTURE

- N-Grams can be viewed as a special technique for conflation (stemming) and as a unique data structure in information systems.
- N-Grams are a fixed length consecutive series of "n" characters.
- Unlike stemming that generally tries to determine the stem of a word that represents the semantic meaning of the word, n-grams do not care about semantics.
- The searchable data structure is transformed into overlapping n-grams, which are then used to create the searchable database.
- Examples of bigrams, trigrams and pentagrams for the word phrase "sea colony."

  se ea co ol lo on ny Bigrams (no interword symbols)

  sea col olo lon onyTrigrams (no interword symbols)

  #se sea ea# #co col olo lon ony ny# Trigrams

  (with interword symbol #)

  #sea# #colo colon olony lony# Pentagrams (with interword symbol #)

- The symbol # is used to represent the interword symbol which is anyone of a set of symbols (e.g., blank, period, semicolon, colon, etc.).

➢ The symbol # is used to represent the interword symbol which is anyone of a set of symbols (e.g., blank, period, semicolon, colon, etc.).

➢ Each of the n-grams created becomes a separate processing tokens and are searchable.

➢ It is possible that the same n-gram can be created multiple times from a single word.

- Uses :

➢ Widely used as cryptography in world war II

➢ Spelling errors detection and correction

- Use bigrams for conflating terms.
- N-grams as a potential erroneous words.
- Damerau specified 4 categories of errors:

Error Category	Example
single char insertion	compuuter
single char deletion	compter
single char substitution	compiter
Transposition of 2 adjacent	comp*tu*er chars

- Zamora showed trigram analysis provided a viable data structure for identifying misspellings and transposed characters.
- This impacts information systems as a possible basis for identifying potential input errors for correction as a procedure within the normalization process.
- Frequency of occurrence of n-gram patterns can also be used for identifying the language of an item.
- Trigrams have been used for text compression and to manipulate the length of index terms.
- To encode profiles for the Selective Dissemination of Information.
- To store the searchable document file for retrospective search databases.

Advantage:

They place a finite limit on the number of searchable token

$MaxSeg_n=(\lambda)^n$ maximum number of unique n grams that can be generated.

" n" is the length of n-grams

λ number of process able symbols

Disadvantage: longer the n gram the size of inversion list increase.

Performance has 85 % precision .

PAT data structure (practical algorithm to retrieve information coded in alphanumeric)

- PAT structure or PAT tree or PAT array : continuous text input data structures(string like N-Gram data structure).
- The input stream is transformed into a searchable data structure consisting of substrings, all substrings are unique.
- Each position in a input string is a anchor point for a sub string.

- In creation of PAT trees each position in the input string is the anchor point for a sub-string that starts at that point and includes all new text up to the end of the input.
- Binary tree, most common class for prefix search,But Pat trees are sorted logically which facilitate range search, and more accurate then inversion file .
- PAT trees provide alternate structure if supporting strings search.

Text Economics for Warsaw is complex.

---

sistring 1 Economics for Warsaw is complex.

sistring 2 conomics for Warsaw is complex.

sistring 5 omics for Warsaw is complex.

sistring 10 for Warsaw is complex.

sistring 20 w is complex.

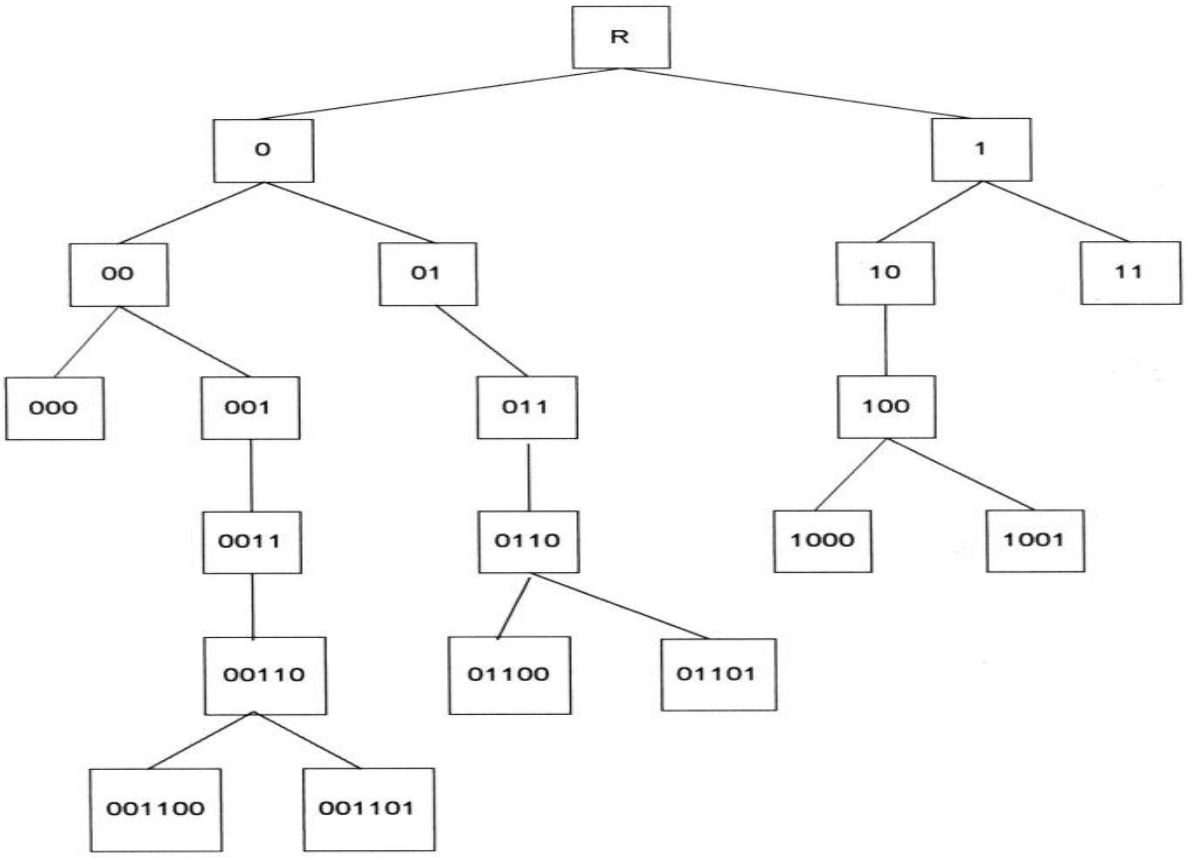
sistring 30 ex.

Examples of sistrings

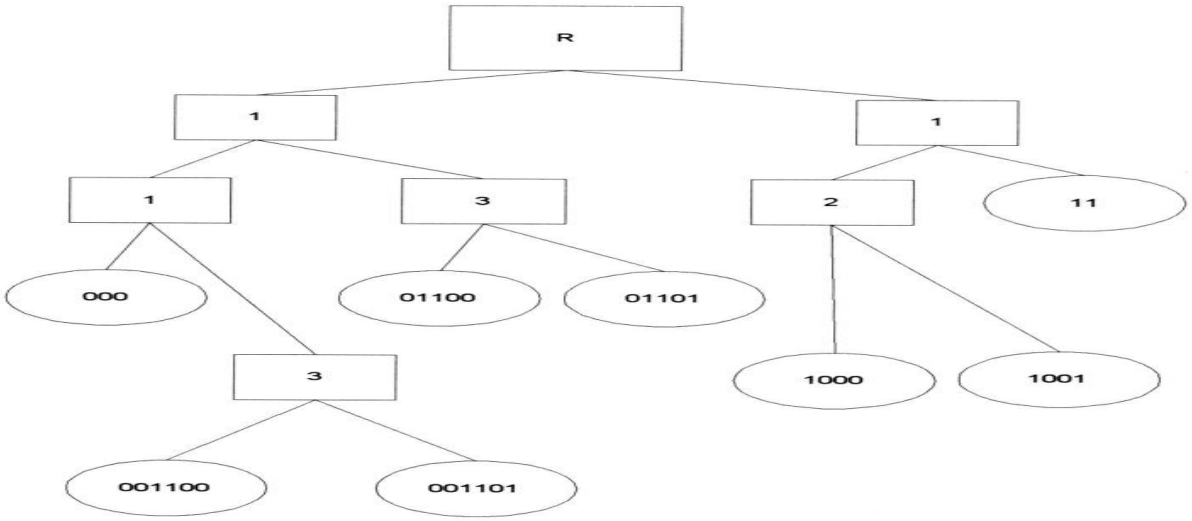
- The key values are stored at the leaf nodes (bottom nodes) in the PAT Tree.
- For a text input of size "n" there are "n" leaf nodes and "n-1" at most higher level nodes.
- It is possible to place additional constraints on sistrings for the leaf nodes.
- If the binary representations of "h" is (100), "o" is (110), "m" is (001) and "e" is (101) then the word "home" produces the input 100110001101.... Using the sistrings.

INPUT	100110001101
sistring 1	1001....
sistring 2	001100...
sistring 3	01100....
sistring 4	11.....
sistring 5	1000...
sistring 6	000.....
sistring 7	001101
sistring 8	01101

The full PAT binary tree is



The value in the intermediate nodes (indicated by rectangles) is the number of bits to skip until the next bit to compare that causes differences between similar terms.



Skipped final version of PAT tree

Signature file structure

- The coding is based upon **words** in the code.
- The words are mapped into word signatures .
- A word signature is fixed length code with a fixed number of bits set to 1.
- The bit positions that are set to one are determined via a hash function of the word.
- The word signatures are **Or**ed together to create signature of an item..
- Partitioning of words is done in block size ,Which is nothing but set of words, Code length is 16 bits .
- Search is accomplished by template matching on the bit position .
- Provide a practical solution applied in parallel processing , distributed environment etc.

- To avoid signatures being too dense with "1"s, a maximum number of words is specified and an item is partitioned into blocks of that size.
- The block size is set at five words, the code length is 16 bits and the number of bits that are allowed to be "1" for each word is five.
- TEXT: Computer Science graduate students study (assume block size is five words)

WORD	Signature
computer	0001 0110 0000 0110
Science	1001 0000 1110 0000
graduate	1000 0101 0100 0010
students	0000 0111 1000 0100
study	0000 0110 0110 0100
Block Signature	1001 0111 1110 0110

Superimposed Coding

Application(s)/Advantage(s)

- Signature files provide a practical solution for storing and locating information in a number of different situations.
- Signature files have been applied as medium size databases, databases with low frequency of terms, WORM devices, parallel processing machines, and distributed environments

HYPERTEXT AND XML DATA STRUCTURES

- The advent of the Internet and its exponential growth and wide acceptance as a new global information network has introduced new mechanisms for representing information.
- This structure is called hypertext and differs from traditional information storage data structures in format and use.
- The hypertext is Hypertext is stored in HTML format and XML .
- Bot of these languages provide detailed descriptions for subsets of text similar to the zoning.
- Hypertext allows one item to reference another item via a embedded pointer .
- HTML defines internal structure for information exchange over WWW on the internet.
- XML: defined by DTD, DOM, XSL, etc.

\

UNIT-3

AUTOMATIC INDEXING

- Classes of Automatic Indexing
- Statistical Indexing
- Natural Language
- Concept Indexing
- Hypertext Linkages

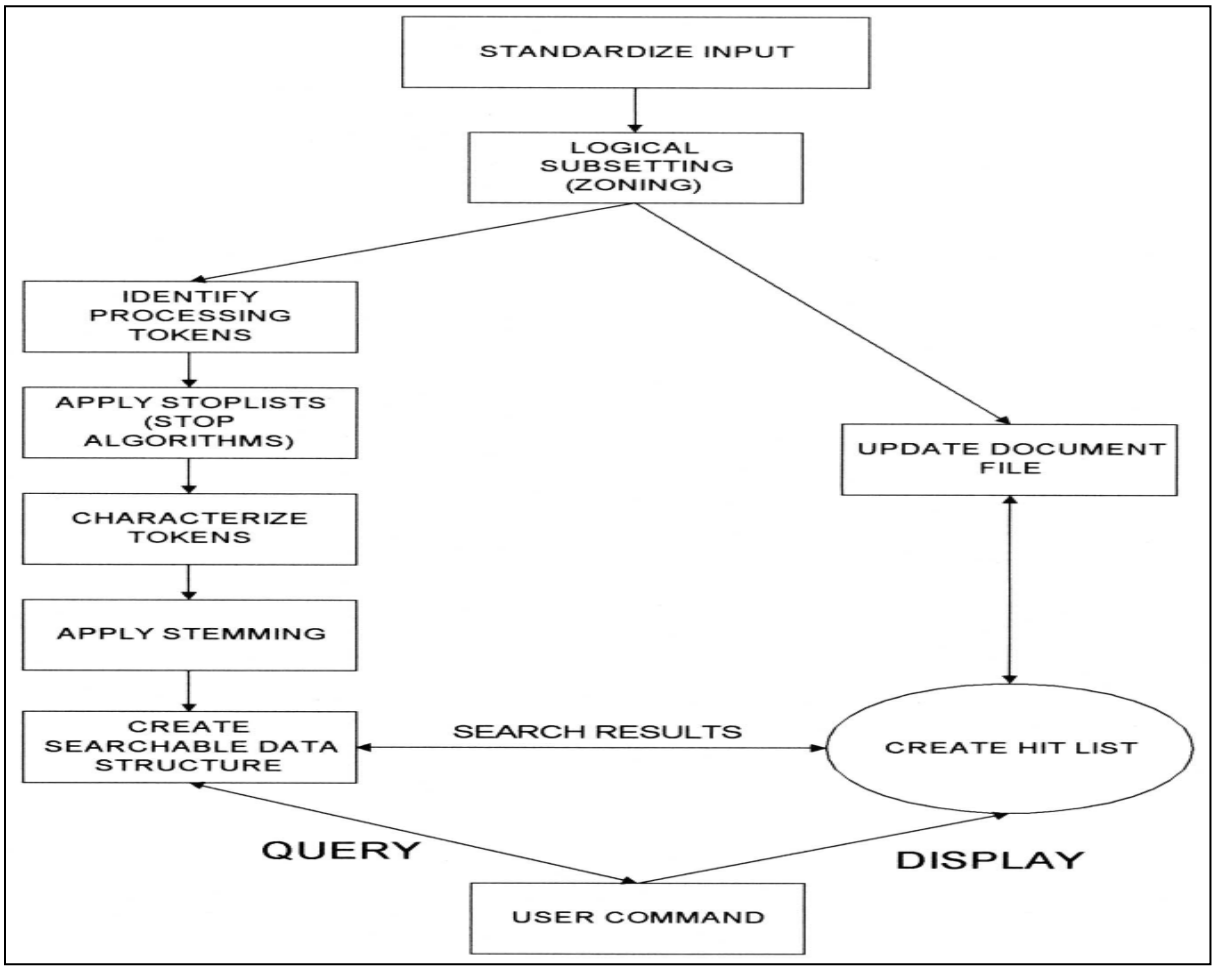
CLASSES OF AUTOMATIC INDEXING

❖ Automatic indexing is the process of analyzing an item to extract the information to be permanently kept in index.

❖ This process is associated with the generation of the searchable data structures associated with an item.

❖ The left hand side of the figure includes identifying processing tokens, apply stop list algorithm , characterize tokens, apply stemming, and creating searchable data structure is part of indexing process.

Data Flow in Information Processing System (Overall fig.)



- ✓ All systems go through the initial stage of zoning and identify the processing tokens to create index.
- ✓ Some systems automatically divide the document up into fixed length passages or localities, which become the item unit that is indexed.
- ✓ Filters such as stop list algorithm and stemming algorithms -to reduce the processing tokens.
- ✓ An index is the *data structure* created to support search strategy.
- ✓ Search strategy – classified as statistical , natural language, and concept.
- ✓ **Statistical strategy** – covers the broadest range of indexing techniques and common in commercial systems .
- ✓ Basis for statistical is – frequency of occurrence of processing tokens(words/ phrases) within documents and within data base.
- ✓ The words/phrases are the domain of searchable values.
- ✓ Statistics that are applied to the event data are probabilistic, Bayesian, vector spaces, neural net.
- ✓ **Statistic approach** – stores a single statistics, such as how often each word occurs in an item – used for generating relevance scores after a Boolean search .
- ✓ **Probabilistic indexing -**stores the information that are used in calculating a probability that a particular item satisfies (i.e., is relevant to) a particular query.
- ✓ **Bayesian and vector space** – stores the information that are used in generating a relative confidence level of an items relevancy to a query.
- ✓ **Neural networks –**dynamic learning structures– comes under concept indexing – that determine concept class.
- ✓ **Natural language** approach perform the similar processing token identification as in statistical techniques - additional level of parsing of the item (present, past, future action) enhance search precision.
- ✓ Concept indexing uses the words within an item to correlate to concepts discussed in the index item.
- ✓ When generating the concept classes automatically, there may not be a name applicable to the concept but just a statistical significance.
- ✓ Finally, a special class of indexing can be defined by creation of hypertext linkages.
- ✓ These linkages provide virtual threads of concepts between items versus directly defining the concept within an item.
- ✓ Each technique has its own strengths and weaknesses.

STATISTICAL INDEXING

- Uses frequency of occurrence of events to calculate the number to indicate potential relevance of an item.
- The documents are found by normal Boolean search and then statistical calculation are performed on the hit file, ranking the out put(e.g. The term- frequency algorithm)

1. Probability weighting
2. Vector Weighting

1. Simple Term Frequency algorithm
2. Inverse Document Frequency algorithm
3. Signal Weighting
4. Discrimination Value
5. Problems with the **weighting schemes** and **vector model**

3. Bayesian Model

1. **Probabilistic weighting**

- Probabilistic systems attempt to calculate a probability value that should be invariant to both calculation method and text corpora(large collection of written/spoken texts).
- The probabilistic approach is based on direct application of theory of probability to information retrieval system.
- Advantage: uses the probability theory to develop the algorithm.
- This allows easy integration of the final results when searches are performed across multiple databases and use different search algorithms.
- The use of probability theory is a natural choice – the basis of evidential reasoning (drawing conclusions from evidence).
- This is summarized by PRP(probability ranking principle) and Plausible corollary (reasonable result)
- PRP –hypothesis –if a reference retrieval systems response to each request is a **ranking** of the documents in order of **decreasing probability** of usefulness to the user, the overall effectiveness of the system to the users is best obtainable on the basis of the data available.
- **Plausible corollary** : the techniques for estimating the probabilities of usefulness for output**ranking in IR** is standard **probability theory and statistics.**
- probabilities are based on binary condition the item is relevant or not.
- IRS the relevance of an item is a continuous function from non- relevant to absolutely useful.
- **Source of problems:** in application of probability theory come from lack of accurate data and simplified assumptions that are applied to mathematical modeling.
- cause the results of probabilistic approaches in ranking items to be less accurate than other approaches.

- *Advantage of probabilistic approach* is that it can identify its weak assumptions and work to strengthens them.
- Ex : **logistical regression**

➢ Approach starts by defining a model 0 system.

- In retrieval system there exits a **query q ¡**and a document term **d_i** which has a set **of attributes (Vi…Vn)** from the query (e.g., counts of term frequency in the query), from the document (e.g., counts of term frequency in the document) and from the database (e.g., total number of documents in the database divided by the number of documents indexed by the term).
- The logistic reference model uses a random sample of query documentterms for which binary relevance judgment has been made from judgment from samples.

o Logarithm O is the logarithm **of odds** of relevance for terms Tk which is present in document Dj and query Qi

o The logarithm that the i^{th} Query is relevant to the j^{th} document is the sum of the logodds for all terms:

- The inverse logistic transformation is applied to obtain the probability of relevance of a document to a query:

o The coefficients of the equation for logodds is derived for a particular database using a random sample of query-document-term-relevance quadruples and used to predict odds of relevance for other query-document pairs.

o Additional attributes of relative frequency in the query (QRF), relative frequency in the document (DRF) and relative frequency of the term in all the documents (RFAD) were included, producing the logodds formula:

o QRF = QAF\ (total number of terms in the query), DRF = DAF\(total number of words in the document) and RFAD = (total number of term occurrences in the database)\ (total number of all words in the database).

o Logs are used to reduce the impact of frequency information; then smooth out skewed distributions.

o A higher max likelihood is attained for logged attributes.

o The coefficients and log (O(R)) were calculated creating the final formula for ranking for query vector Q', which contains q terms:

$$\log(O(R \mid \vec{Q})) = -5.138 + \sum_{k=1}^{q} (Z_j + 5.138)$$

O The logistic inference method was applied to the test database along with the Cornell SMART vector system, inverse document frequency and cosine relevance weighting formulas.

- The logistic inference method outperformed the vector method.
- Attempts have been made to combine different probabilistic techniques to get a more accurate value.
  - This combination of logarithmic odds has not presented better results.
  - The objective is to have the strong points of different techniques compensate for weaknesses.
  - To date this combination of probabilities using averages of Log-Odds has not produced better results and in many cases produced worse results.

2 . Vector weighing

- Earliest system that investigated statistical approach is SMART system of Cornell university. – system based on vector model.
- Vector is one dimension of set of values, where the order position is fixed and represents a domain .
- Each position in the vector represents a processing token
- Two approaches to the **domain values** in the vector – *binary or weighted*
- Under binary approach the domain contains a value of **1 or 0**
- Under weighted - domain is set of positive value – the value of each processing token represents the **relative importance of the item**.
- Binary vector requires a decision process to determine if the degree that a particular token the semantics of item is sufficient to include in the vector.
- Ex., a five-page item may have had only one sentence like "Standard taxation of the shipment of the oil to refineries is enforced."
- For the binary vector, the concepts of "Tax" and "Shipment" are below the threshold of importance (e.g., assume threshold is 1.0) and they not are included in the vector.

Binary and Vector Representation of an Item

	Petroleum	Mexico	Oil	Taxes	Refineries	Shipping
Binary	(1	, 1	, 1	, 0	, 1	, 0)
Weighted	(2.8	, 1.6	, 3.5	, .3	, 3.1	, .1)

- A Weighted vector acts same as the binary vector but provides a range of values that accommodates a variance in the value of relative importance of processing tokens in representing the item .
- The use of weights also provides a basis for determining the rank of an item.
- **The vector approach allows for mathematical and physical representation using a vector space model.**
- Each processing token can be considered another dimension in an item representation space.

- 3D vector representation assuming there were only three processing tokens, Petroleum Mexico and Oil.

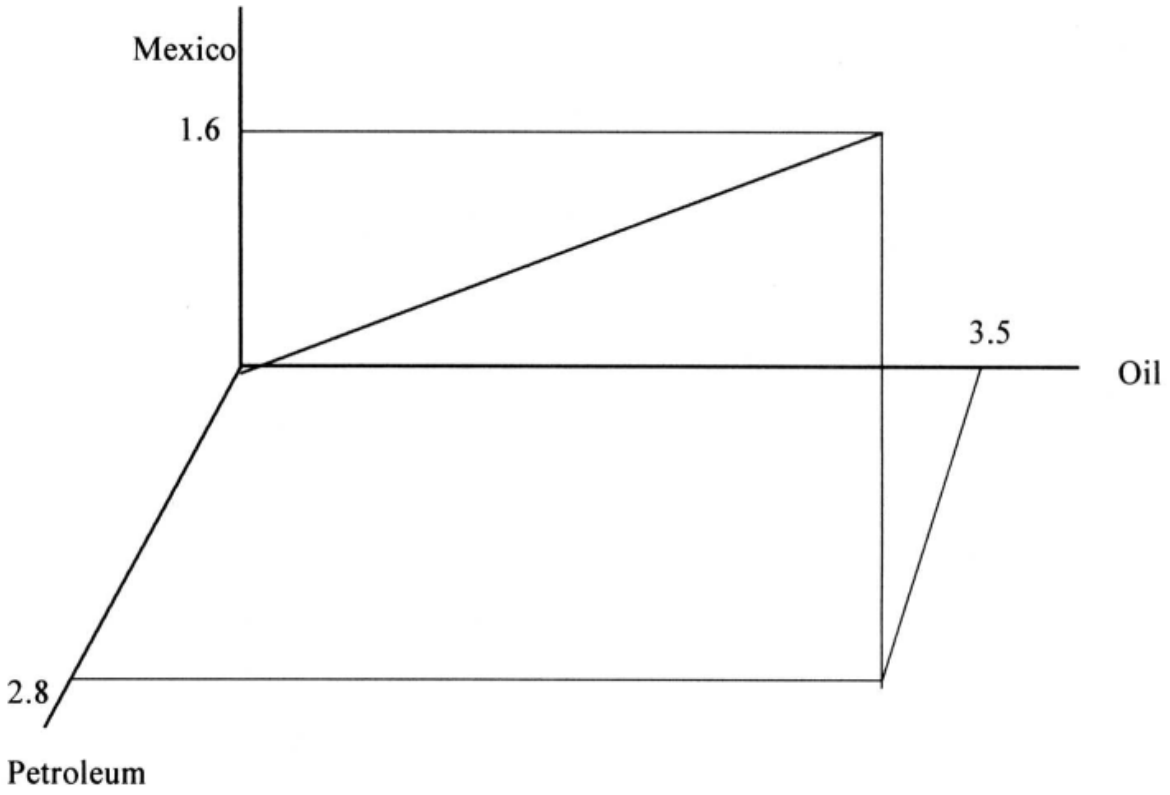


Fig : Vector Represenation

2. 1 .Simple term frequency algorithms

- In both **weighted and un weighted approaches** an automatic index processing implements an algorithm to determine the weight to be assigned to a processing token .
- **In statistical system** : the data that are potentially available for calculating a weight are the frequency of occurrence of the processing token in an existing item (i.e., term frequency - TF), the frequency of occurrence of the processing token in the existing database (i.e., total frequency -TOTF) and the number of unique items in the database that contain the processing token (i.e., item frequency - IF, frequently labeled in other document frequency - DF).
- Simplest approach is to have the weight **equal to the term frequency.**
- If the word "computer" occurs 15 times within an item it has a weight of 15.
- The term frequency weighting formula: **(1+log(TF))/1+log(average(TF)) / (1-slope) * pivot + slope*no. of unique terms**

- where slope was set at .2 and the pivot was set to the average no of unique terms occurring in the collection.
- Slope and pivot are constants for any document/query set.
- This leads to the final algorithm that weights each term by the above formula divided by the pivoted normalization:

$$((1 + \log(TF))/(1 + \log(\text{average}(TF))/(\text{slope})(\text{No. unique terms}) + (1\text{-slope})*(\text{pivot})$$

2. Inverse document frequency(IDF)

- Enhancing the weighting algorithm : the weights assigned to the term should be inversely proportional to the frequency of term occurring in the data base.
- The term "computer" represents a concept used in an item, but it does not help a user find the specific information being sought since it returns the complete DB.
- This leads to the general statement enhancing weighting algorithms that the weight assigned to an item should be inversely proportional to the frequency of occurrence of an item in the database.
- This algorithm is called inverse document frequency (IDF).
- The un-normalized weighting formula is:

$$WEIGHT = TF_{ij} * [Log_2(n) – Log_2+(IF_{ij})+1]$$

- where WEIGHTij is the vector weight that is assigned to term "j" in item "i,"
- Tf_{ij} (term frequency) is the frequency of term "j" in item "i", "n" is the number of items in the database and
- IF_{ij} (item frequency or document frequency) is the number of items in the database that have term "j" in them.
- Ex., Assume that the term "oil" is found in 128 items, "Mexico" is found in 16 items and "refinery" is found in 1024 items.
- If a new item arrives with all three terms in it, "oil" found 4 times, "Mexico" found 8 times, and "refinery found 10 times and there are 2048 items in the total database.
- weight calculations using inverse document frequency.
- with the resultant inverse document frequency item vector = (20, 64, 20).
- The value of "n" and IF vary as items are added and deleted from the database.

$$Weight_{oil} = 4 * (Log_2(2048) - Log_2(128) + 1) = 4 * (11 - 7 + 1) = 20$$

$$Weight_{Mexico} = 8 * (Log_2(2048) - Log_2(16) + 1) = 8 * (11 - 4 + 1) = 64$$

$$Weight_{refinery} = 10 * (Log_2(2048) - Log_2(1024) + 1) = 10 * (11 - 10 + 1) = 20$$

3. Signal weighting

- IDF adjusts the weight of a processing token for an item based upon the number of items that contain the term in the existing database.
- It does not account for is the term frequency distribution of the processing token in the items that contain the term - can affect the ability to rank items.
- For example, assume the terms "SAW" and "DRILL" are found in 5 items with the following frequencies:

Item Distribution	SAW	DRILL
A	10	2
B	10	2
C	10	18
D	10	10
E	10	18

- In Information Theory, the information content value of an object is inversely proportional to the probability of occurrence of the item.
- An instance of an event that occurs all the time has less information value than an instance of a seldom occurring event.
- This is typically represented as INFORMATION = -Log2(p), where p is the probability of occurrence of event "p."
- The information value for an event that occurs

  .5 %of the time is: occurs 50 %of the time is:
- If there are many independent occurring events then the calculation for the average information value across the events is:

$$\text{INFORMATION} = -\text{Log}_2(.0005) = -(-10) = 10$$

$$\text{INFORMATION} = -\text{Log}_2(.50) = -(-1) = 1$$

$$\text{AVE\_INFO} = -\sum_{k=1}^{n} p_k \text{Log}_2(p_k)$$

- Its value decreases proportionally to increases in variances in the values of can be defined as
- formula for calculating the weighting factor called Signal (Dennis-67) can be used:
- producing a final formula of:

$$\text{Weight}_{ik} = TF_{ik} * \text{Signal}_k$$

$$\text{Weight}_{ik} = TF_{ik} * [\text{Log}_2(TOTF_k) - \sum_{i=1}^{n} TF_{ik}/TOTF_k \text{Log}_2(TF_{ik}/TOTF_k)]$$

Item Distribution	SAW	DRILL
A	10	2
B	10	2
C	10	18
D	10	10
E	10	18

$$Signal_{SAW} = LOG_2(50) - [5 * \{10/50LOG_2(10/50)\}]$$

$$Signal_{DRILL} = LOG_2(50) - [2/50LOG_2(2/50) + 2/50LOG_2(2/50) + 18/50LOG_2(18/50) + 10/50LOG_2(10/50) + 18/50LOG_2(18/50)$$

- The weighting factor for term "DRILL" that does not have a uniform distribution is larger than that for term "SAW" and gives it a higher weight.
- This technique could be used by itself or in combination with inverse document frequency or other algorithms.
- The overhead of the additional data needed in an index and the calculations required to get the values have not been demonstrated to produce better results.
- It is a good example of use of Information Theory in developing information retrieval algorithms.

4. Discrimination value

- Creating a **weighting algorithm** based on the **discrimination of value of the term**.
- all items appear the same, the harder it is to identify those that are needed.
- Salton and Yang proposed a weighting algorithm that takes into consideration the ability for a search term to discriminate among items.
- They proposed use of a discrimination value for each term "i":
- where AVESIM is the average similarity between every item in the database and AVESIMi is the same calculation except that term "i" is removed from all items.

$$DISCRIM_i = AVESIM_i - AVESIM$$

- DISCRIMi value being positive, close to zero/negative.
- A positive value indicates that removal of term "i" has increased the similarity between items.
- In this case, leaving the term in the database assists indiscriminating between items and is of value.
- A value close to zero implies that the term's removal or inclusion does not change the similarity between items.

- If the value is negative, the term's effect on the database is to make the items appear more similar since their average similarity decreased with its removal.
- Once the value of DISCRMi is normalized as a positive number, it can be used in the standard weighting formula as:

$$Weight_{ik} = TF_{ik} * DISCRIM_k$$

Problems With Weighting Schemes

- Often weighting schemes use information that is based upon processing token distributions across the database.
- Information databases tend to be dynamic with new items always being added and to a lesser degree old items being changed or deleted.
- Thus these factors are changing dynamically.
- There are a number of approaches to compensate for the constant changing values.

  a. Ignore the variances and calculate weights based upon current values, with the factors changing over time. Periodically rebuild the complete search database.

  b. Use a fixed value while monitoring changes in the factors. When the changes reach a certain threshold, start using the new value and update all existing vectors with the new value.

  c. Store the invariant variables (e.g., term frequency within an item) and at search time calculate the latest weights for processing tokens in items needed for search terms.

- First approach, Periodically the database and all term weights are recalculated based upon the most recent updatesto the database.
- For large databases in the millions of items, the overhead of rebuilding the database can be significant.
- In the second approach, there is a recognition that for the most frequently occurring items, the aggregate values are large.
- As such, minor changes in the values have negligible effect on the final weight calculation.
- Thus, on a term basis, updates to the aggregate values are only made when sufficient changes not using the current value will have an effect on the final weights and the search/ranking process.
- This process also distributes the update process over time by only updating a subset of terms at any instance in time.
- The third approach is the most accurate. The weighted values in the database only matter when they are being used to determine items to return from a query or the rank order to return the items.
- This has more overhead in that database vector term weights must be calculated dynamically for every query term.
- SOLUTIONS:

- If the system is using an inverted file search structure, this overhead is very minor.
- The best environment would allow a user to run a query against multiple different time periods and different databases that potentially use different weighting algorithms, and have the system integrate the results into a single ranked Hit file.

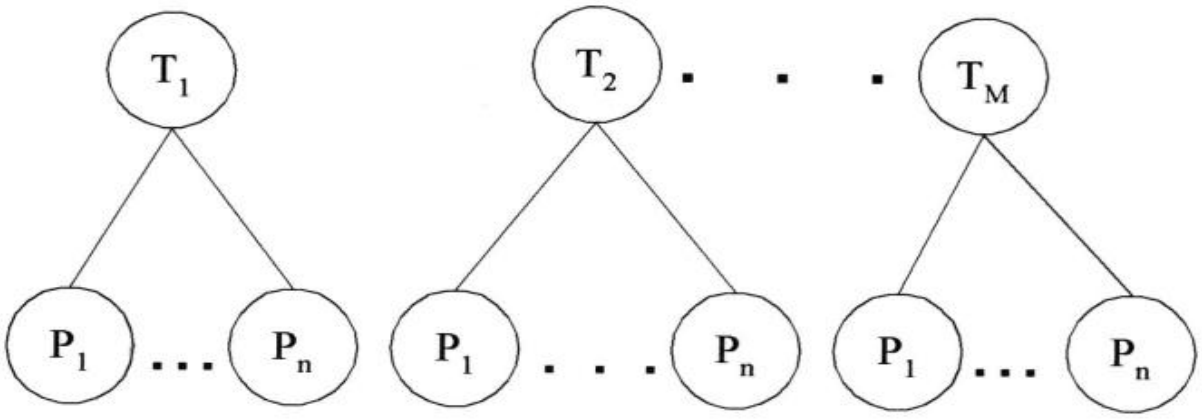
Problems With the Vector Model

- A major problem comes in the vector model when there are multiple topics being discussed in a particular item.
- For example, assume that an item has an in-depth discussion of "oil" in "Mexico" and also "coal" in "Pennsylvania." The vector model does not have a mechanism to associate each energy source with its particular geographic area.
- There is no way to associate correlation factors between terms since each dimension in a vector is independent of the other dimensions.
- Thus the item results in a high value in a search for "coal in Mexico."
- Another major limitation of a vector space is in associating positional information with a processing term.
- The concept of a vector space allows only one scalar value to be associated with each processing term for each item.

Bayesian Model

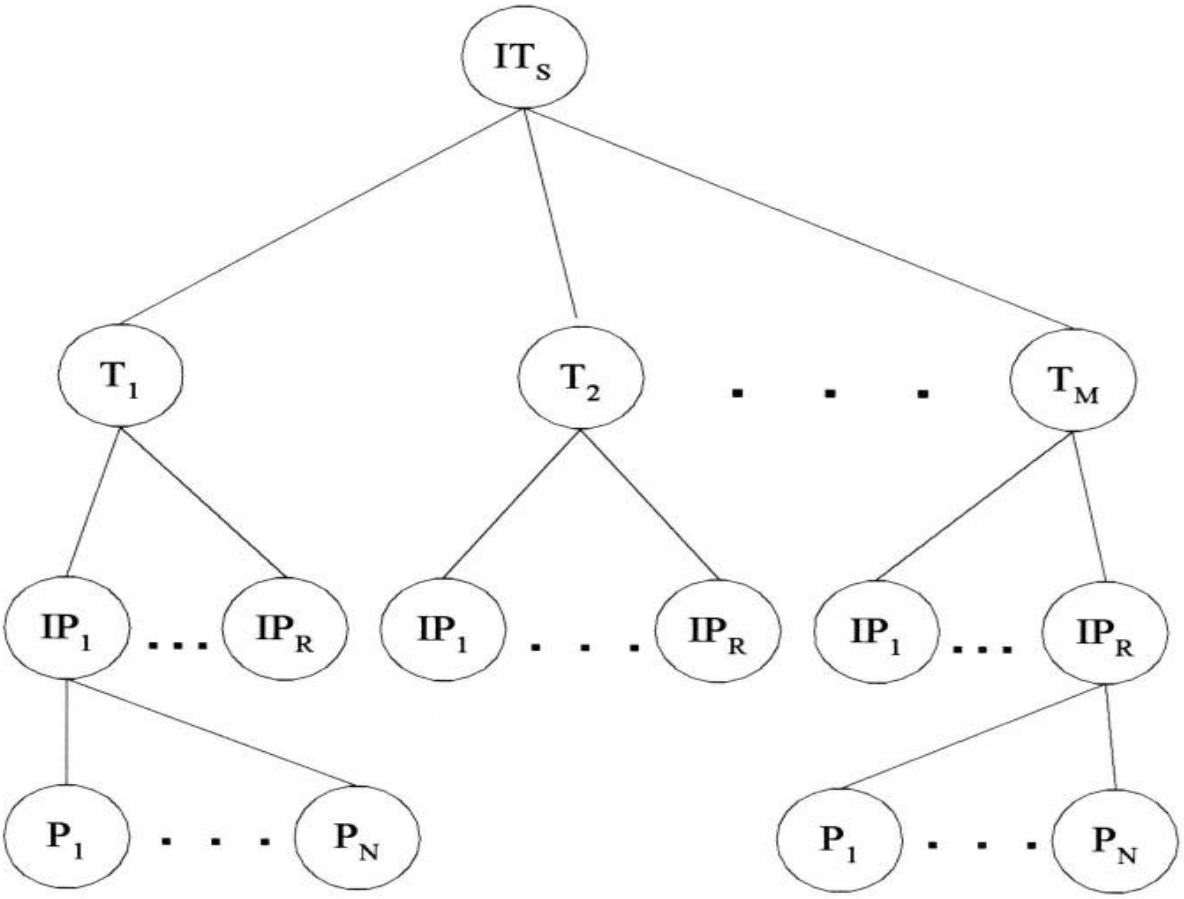
- One way of overcoming the restrictions inherent in a vector model is to use a Bayesian approach to maintaining information on processing tokens.
- The Bayesian model provides a conceptually simple yet complete model for information systems.
- The Bayesian approach is based upon conditional probabilities (e.g., Probability of Event 1 given Event 2 occurred).
- This general concept can be applied to the search function as well as to creating the index to the database.
- The objective of information systems is to return relevant items.
- Thus the general case, using the Bayesian formula, is P(REL/DOCi , Queryj) which is interpreted as the probability of relevance (REL) to a search statement given a particular document and query.
- In addition to search, Bayesian formulas can be used in determining the weights associated with a particular processing token in an item.
- The objective of creating the index to an item is to represent the semantic information in the item.
- A Bayesian network can be used to determine the final set of processing tokens (called topics) and their weights.

- A simple view of the process where Ti represents the relevance of topic "i" in a particular item and Pj represents a statistic associated with the event of processing token "j" being present in the item.



- "m" topics would be stored as the final index to the item.
- The statistics associated with the processing token are typically frequency of occurrence.
- But they can also incorporate proximity factors that are useful in items that discuss multiple topics.
- There is one major assumption made in this model:Assumption of Binary Independence : the topics and the processing token statistics are independent of each other. The existence of one topic is not related to the existence of the other topics. The existence of one processing token is not related to the existence of other processing tokens.
- In most cases this assumption is not true. Some topics are related to other topics and some processing tokens related to other processing tokens.
- For example, the topics of "Politics" and "Economics" are in some instances related to each other and in many other instances totally unrelated.
- The same type of example would apply to processing tokens.
- There are two approaches to handling this problem.
- The first is to assume that there are dependencies, but that the errors introduced by assuming the mutual independence do not noticeably effect the determination of relevance of an item nor its relative rank associated with other retrieved items.
- This is the most common approach used in system implementations.
- A second approach can extend the network to additional layers to handle interdependencies.
- Thus an additional layer of Independent Topics (ITs) can be placed above the Topic layer and a layer of Independent Processing Tokens (IPs) can be placed above the processing token layer.

Extended Bayesian Network



- The new set of Independent Processing Tokens can then be used to define the attributes associated with the set of topics selected to represent the semantics of an item.
- To compensate for dependencies between topics the final layer of Independent Topics is created.
- The degree to which each layer is created depends upon the error that could be introduced by allowing for dependencies between Topics or Processing Tokens.
- Although this approach is the most mathematically correct, it suffers from losing a level of precision by reducing the number of concepts available to define the semantics of an item.

NATURAL LANGUAGE

- The goal of natural language processing is to use the semantic information in addition to the statistical information to enhance the indexing of the item.
- This improves the precision of searches, reducing the number of false hits a user reviews.

- The semantic information is extracted as a result of processing the language rather than treating each word as an independent entity.
- The simplest output of this process results in generation of phrases that become indexes to an item.
- More complex analysis generates thematic representation of events rather than phrases.
- Statistical approaches use proximity as the basis behind determining the strength of word relationships in generating phrases.
- For example, with a proximity constraint of adjacency, the phrases "venetian blind" and "blind Venetian" may appear related and map to the same phrase.
- But syntactically and semantically those phrases are very different concepts.
- Word phrases generated by natural language processing algorithms enhance indexing specification and provide another level of disambiguation.
- Natural language processing can also combine the concepts into higher level concepts sometimes referred to as thematic representations.

1. Index Phrase Generation

- The goal of indexing is to represent the semantic concepts of an item in the information system to support finding relevant information.
- Single words have conceptual context, but frequently they are too general to help the user find the desired information.
- Term phrases allow additional specification and focusing of the concept to provide better precision and reduce the user's overhead of retrieving non-relevant items.
- Having the modifier "grass" or "magnetic" associated with the term "field" clearly disambiguates between very different concepts.
- One of the earliest statistical approaches to determining term phrases using of a COHESION factor between terms (Salton-83):

$$COHESION_{k,h} = SIZE\text{-}FACTOR * (PAIR\text{-}FREQ_{k,h} / TOTF_k * TOTF_H)$$

- where SIZE-FACTOR is a normalization factor based upon the size of the vocabulary and is the total frequency of co-occurrence of the pair $Term_k$, $Term_h$ in the item collection.
- Co-occurrence may be defined in terms of adjacency, word proximity, sentence proximity, etc.
- This initial algorithm has been modified in the SMART system to be based on the following guidelines
- any pair of adjacent non-stop words is a potential phrase
- any pair must exist in 25 or more items
- phrase weighting uses a modified version of the SMART system single term algorithm
- normalization is achieved by dividing by the length of the single-term sub-vector.
- Statistical approaches tend to focus on two term phrases.

- The natural language approaches is their ability to produce multiple-term phrases to denote a single concept.
- If a phrase such as “industrious intelligent students” was used often, a statistical approach would create phrases such as “industrious intelligent” and “intelligent student.”
- A natural language approach would create phrases such as “industrious student,” “intelligent student” and “industrious intelligent student.”
- The first step in a natural language determination of phrases is a lexical analysis of the input.
- In its simplest form this is a part of speech tagger that, for example, identifies noun phrases by recognizing adjectives and nouns.
- Precise part of speech taggers exist that are accurate to the 99 per cent range.
- Additionally, proper noun identification tools exist that allow for accurate identification of names, locations and organizations since these values should be indexed as phrases and not undergo stemming.
- The Tagged Text Parser (TTP), based upon the Linguistic String Grammar (Sager-81), produces a regularized parse tree representation of each sentence reflecting the predicate-argument structure (Strzalkowski-93).
- The tagged text parser contains over 400 grammar production rules. Some examples of

CLASS	EXAMPLES
determiners	a, the
singular nouns	paper, notation, structure, language
plural nouns	operations, data, processes
preposition	in, by, of, for
adjective	high, concurrent
present tense verb	presents, associates
present participal	multiprogramming

- The TTP parse trees are header-modifier pairs where the header is the main concept and the modifiers are the additional descriptors that form the concept and eliminate ambiguities.
- Ex., ‘The former Soviet President has been a local hero’ regularized parse tree structure generated for the

|assert

perf[HAVE]

verb[BE]

subject

np

noun[President]

t_pos[The]

adj[former]

adj[Soviet]

object

np

noun[hero]

t_pos[a]

adj[local]

- To determine if a header-modifier pair warrants indexing,Strzalkowski calculates a value for Informational Contribution (IC) for each element in the pair.
- The basis behind the IC formula is a conditional probability between the terms. The formula for IC between two terms (x,y) is:
- Where is the frequency of (x,y) in the database, is the number of pairs in which "x" occurs at the same Position as in (x,y) and D(x) is the dispersion parameter which is the number of distinct words with which x is paired. When IC= 1, x occurs only with the
- following formula for weighting phrases:
- is 1 for i<N and 0 otherwise C1, C2 normalizing factors.

2 Natural Language Processing

- Natural language processing not only produces more accurate term phrases, but can provide higher level semantic information identifying relationships between concepts.
- System adds the functional processes Relationship Concept Detectors, Conceptual Graph Generators and Conceptual Graph Matchers that generate higher level linguistic relationships including semantic and is course level relationships.
- During the first phase of this approach, the processing tokens in the document are mapped to Subject Codes.
- These codes equate to index term assignment and have some similarities to the concept-based systems.
- The next phase is called the Text Structurer, which attempts to identify general discourse level areas within an item.
- The next level of semantic processing is the assignment of terms to components, classifying the intent of the terms in the text and identifying the topical statements.
- The next level of natural language processing identifies interrelationships between the concepts.
- The final step is to assign final weights to the established relationships.
- The weights are based upon a combination of statistical information and values assigned to the actual words used in establishing the linkages.

CONCEPT INDEXING

- Natural language processing starts with a basis of the terms within an item and extends the information kept on an item to phrases and higher level concepts such as the relationships between concepts.
- Concept indexing takes the abstraction a level further.
- Its goal is use concepts instead of terms as the basis for the index, producing a reduced dimension vector space.
- Concept indexing can start with a number of unlabeled concept classes and let the information in the items define the concepts classes created.
- A term such as "automobile" could be associated with concepts such as "vehicle," "transportation," "mechanical device," "fuel," and "environment."
- The term "automobile" is strongly related to "vehicle," lesser to "transportation" and much lesser the other terms.
- Thus a term in an item needs to be represented by many concept codes with different weights for a particular item.
- The basis behind the generation of the concept approach is a neural network model.
- Special rules must be applied to create a new concept class.
- Example demonstrates how the process would work for the term "automobile."

TERM: automobile

	Weights for associated concepts:
Vehicle	.65
Transportation	.60
Environment	.35
Fuel	.33
Mechanical Device	.15

Vector Representation Automobile: (.65,..., .60, ..., .35, .33, ... , .15)

HYPERTEXT LINKAGES

- It's a new class of information representation is evolving on the Internet.
- Need to be generated manually Creating an additional information retrieval dimension.
- Traditionally the document was viewed as two dimensional .
- **Text** of the item as one dimension and **references** as second dimension.
- Hypertext with its linkages to additional electronic items , can be viewed as networking between the items that extend contents, i.e by embedding linkage allows the user to go immediately to the linked item.

- **Issue :** how to use this additional dimension to locate relevant information **.**

- At the internet we have three classes of mechanism to help find information.

1. Manually generated indexes ex: www.yahoo.com were information sources on the home page are indexed manually into hyperlink hierarchy.

- The user navigates through hierarchy by expanding the hyper link.

- At some point the user starts to see the end of items.

2. **Automatically generated indexes -** sites like lycos.com and altavista.com automatically go to other internet sites and return the text , ggogle.com.

3. **Web Crawler's** : A **web crawler** (also known as a **Web spider** or **Web robot**) is a program or automated script which browses the World Wide Web in a methodical, automated manner.

- Are tools that that allow a user to define items of interest and they automatically go to various sites on the net and search for the desired information.' know as search tool rather than indexing.
- **What is needed is an indexalgorithm for items that look at hypertext linkages as an extension of the concept where the link exits .**
- Current concept is defined by the proximity of information .
- **Attempts** have been made to achieve automatically generate hyper link between items . But they suffer from the **dynamic growing data bases.**
- Significant portion of errors have come from **parsing** rather than algorithm problems.

UNIT-4

DOCUMENT AND TERM CLUSTERING

CLUSTERING

Cluster analysis is a technique for multivariate analysis that assigns items to automatically created groups based on a calculation of the degree of association between items and groups. In the information retrieval (IR) field, cluster analysis has been used to create groups of documents with the goal of improving the efficiency and effectiveness of retrieval, or to determine the structure of the literature of a field. The terms in a document collection can also be clustered to show their relationships. The two main types of cluster analysis methods are the non-hierarchical, which divide a data set of N items into M clusters, and the hierarchical, which produce a nested data set in which pairs of items or clusters are successively linked. The non-hierarchical methods such as the single pass and reallocation methods are heuristic in nature and require less computation than the hierarchical methods. However, the hierarchical methods have usually been preferred for cluster-based document retrieval. The commonly used hierarchical methods, such as single link, complete link, group average link, and Ward's method, have high space and time requirements. In order to cluster the large data sets with high dimensionality that are typically found in IR applications, good algorithms (ideally $O(N^2)$ time, $O(N)$ space) must be found. Examples are the SLINK and minimal spanning tree algorithms for the single link method, the Voorhees algorithm for group average link, and the reciprocal nearest neighbour algorithm for Ward's method.

4.1 CLUSTER ANALYSIS

4.1.1 Introduction

Cluster analysis is a statistical technique used to generate a category structure which fits a set of observations. The groups which are formed should have a high degree of association between members of the same group and a low degree between members of different groups (Anderberg 1973). While cluster analysis is sometimes referred to as automatic classification, this is not strictly accurate since the classes formed are not known prior to processing, as classification implies, but are defined by the items assigned to them.

Because there is no need for the classes to be identified prior to processing, cluster analysis is useful to provide structure in large multivariate data sets. It has been described as a tool of discovery because it has the potential to reveal previously undetected relationships based on complex data (Anderberg 1973). An early application of cluster analysis was to determine taxonomic relationships among species. Psychiatric profiles, medical and clinical data, census and survey data, images, and chemical structures and properties have all been studied using cluster analytic methods, and there is an extensive and widely scattered journal literature on the subject.

Basic texts include those by Anderberg (1973), Hartigan (1975), Everitt (1980), Aldenderfer and Blashfield (1984), Romesburg (1984), Spath (1985), Jain and Dubes (1988) and Kaufman (1990). Taxonomic applications have been described by Sneath and Sokal (1973), and social science applications by Lorr (1983) and Hudson and Associates (1982). Comprehensive reviews by Lee (1981), Dubes and Jain (1980) and Gordon (1987) are also recommended.

Because cluster analysis is a technique for multivariate analysis that has application in many fields, it is supported by a number of software packages which are often available in academic and other computing environments. Most of the methods and some of the algorithms described in this chapter are found in statistical analysis packages such as SAS, SPSSX, and BMDP and cluster analysis packages such as CLUSTAN and CLUSTAR/CLUSTID. Brief descriptions and sources for these and other packages are provided by Romesburg (1984).

4.1.2 Applications in Information Retrieval

The ability of cluster analysis to categorize by assigning items to automatically created groups gives it a natural affinity with the aims of information storage and retrieval. Cluster analysis can be performed on documents in several ways:

- Documents may be clustered on the basis of the terms that they contain. The aim of this approach has usually been to provide more efficient or more effective retrieval, though it has also been used after retrieval to provide structure to large sets of retrieved documents. In distributed systems, clustering can be used to allocate documents for storage. A recent review (Willett 1988) provides a comprehensive summary of research on term-based document clustering.

- Documents may be clustered based on co-occurring citations in order to provide insights into the nature of the literature of a field (e.g., Small and Sweeney [1985]).

- Terms may be clustered on the basis of the documents in which they co-occur, in order to aid in the construction of a thesaurus or in the enhancement of queries (e.g., Crouch [1988]).

Although cluster analysis can be easily implemented with available software packages, it is not without problems. These include:

- Selecting the attributes on which items are to be clustered and their representation.

- Selecting an appropriate clustering method and similarity measure from those available .

- Creating the clusters or cluster hierarchies, which can be expensive in terms of computational resources.

- Asessing the validity of the result obtained.

- If the collection to be clustered is a dynamic one, the requirements for update must be considered.

- If the aim is to use the clustered collection as the basis for information retrieval, a method for searching the clusters or cluster hierarchy must be selected.

The emphasis in this chapter will be on the range of clustering methods available and algorithms for their implementation, with discussion of the applications and evaluation that have been carried out in an information retrieval environment. The following notation will be used: N for the numberof items D_i in a data set, L for its dimensionality, and M for the number of clusters C_i created. It will be assumed that the items to be clustered are documents, and each document D_i (or cluster representative C_i) is represented by (w$eight_i$1, . . ., $weight_{iL}$), where $weight_{ik}$ is the weight assigned to $term_k$ in D_i or C_i. The choice of an appropriate

document representation is discussed elsewhere in this text; a summary of research on term-weighting approaches is provided by Salton and Buckley (1988).

4.2 MEASURES OF ASSOCIATION

4.2.1 Introduction

In order to cluster the items in a data set, some means of quantifying the degree of association between them is required. This may be a distance measure, or a measure of similarity or dissimilarity. Some clustering methods have a theoretical requirement for use of a specific measure (Euclidean distance for Ward's method, for example), but more commonly the choice of measure is at the discretion of the researcher.

While there are a number of similarity measures available, and the choice of similarity measure can have an effect on the clustering results obtained, there have been only a few comparative studies (summarized by Willett [1988]). In cluster-based retrieval, the determination of interdocument similarity depends on both the document representation, in terms of the weights assigned to the indexing terms characterizing each document, and the similarity coefficient that is chosen. The results of tests by Willett (1983) of similarity coefficients in cluster-based retrieval suggest that it is important to use a measure that is normalized by the length of the document vectors. The results of tests on weighting schemes were less definitive but suggested that weighting of document terms is not as significant in improving performance in cluster-based retrieval as it is in other types of retrieval. Sneath and Sokal (1973) point out that simple similarity coefficients are often monotonic with more complex ones, and argue against the use of weighting schemes. The measures described below are commonly used in information retrieval applications. They are appropriate for binary or real-valued weighting scheme.

4.2.2 Similarity Measures

A variety of distance and similarity measures is given by Anderberg (1973), while those most suitable for comparing document vectors are discussed by Salton (1989). The Dice, Jaccard and cosine coefficients have the attractions of simplicity and normalization and have often been used for document clustering.

Dice coefficient:

$$S_{D_i,D_j} = \frac{2\sum_{k=1}^{L}(weight_{ik}\,weight_{jk})}{\sum_{k=1}^{L} weight_{ik}^2 + \sum_{k=1}^{L} weight_{jk}^2}$$

If binary term weights are used, the Dice Coefficient reduces to:

$$S_{D_i,D_j} = \frac{2C}{A + B}$$

where C is the number of terms that D_i and D_j have in common, and A and B are the number of terms in D_i and D_j.

Jaccard coefficient:

$$S_{D_i,D_j} = \frac{\sum_{k=1}^{L} (weight_{ij} weight_{jk})}{\sum_{k=1}^{L} weight_{ik}^2 + \sum_{k=1}^{L} weight_{jk}^2 - \sum_{k=1}^{L} (weight_{ik} weight_{jk})}$$

Cosine coefficient:

$$S_{D_i,D_j} = \frac{\sum_{k=1}^{L} (weight_{ij} weight_{jk})}{\sqrt{\sum_{k=1}^{L} weight_{ik}^2 \sum_{k=1}^{L} weight_{jk}^2}}$$

4.2.3 The Similarity Matrix

Many clustering methods are based on a pairwise coupling of the most similar documents or clusters, so that the similarity between every pair of points must be known. This necessitates the calculation of the*similarity matrix*; when the similarity measure is symmetric ($S_{ij} = S_{ji}$), the lower triangular matrix is sufficient (Figure 4.1).

The inverted file algorithm is particularly useful in limiting the amount of computation required to calculate a similarity matrix, if the similarity measure used is one that results in a 0 value whenever a document-document or document-cluster pair have no terms in common (Willett 1980; Perry and Willett 1983). The document term list is used as an index to the inverted index lists that are needed for the similarity calculation. Only those document/cluster pairs that share at least one common term will have their similarity calculated; the remaining values in the similarity matrix are set to 0. The inverted file algorithm is as follows:

$$S = \begin{vmatrix} S_{21} & & & & \\ S_{31} & S_{32} & & & \\ S_{41} & S_{42} & S_{43} & & \\ \vdots & \vdots & \vdots & \ddots & \\ S_{N1} & S_{N2} & S_{N3} & \cdots & S_{N(N-1)} \end{vmatrix}$$

Figure 4.1: Similarity matrix

```
for ( docno = 0; docno < n; docno++ )
{
for ( i = 0; i < doclength; i++ )
{
retrieve_inverted_list ( term[i] );
for ( j = 0; j < invlength; j++ ) counter[doc[j]]++;
}
for ( doc2 = 0; doc2 < n; doc2++ )
{
if (counter [doc2]) calc_similarity( docno, doc2 );
}
}
```

The inverted file algorithm can be effectively incorporated in the clustering algorithms described in this chapter when the calculation of the similarity matrix, or a single row of it, is required for a document collection.

It should be noted that the similarity matrix can be the basis for identifying a nearest neighbor (NN), that is, finding the closest vector to a given vector from a set of N multidimensional vectors. The identification of an NN arises in many clustering algorithms, and for large data sets makes a significant contribution to the computational requirement. Calculating and storing the similarity matrix, or recalculating it when needed, provides a brute force approach to nearest neighbor identification. Therefore, if an efficient NN-finding algorithm can be incorporated into the clustering algorithm, considerable savings of processing time may be achieved. However, although there are a number of techniques available for introducing efficiency into the NN-finding process (Bentley et al. 1980; Murtagh 1985), these techniques are generally inappropriate for data sets with the high dimensionality typical of information retrieval applications. Use of the inverted file algorithm to calculate the similarity matrix or a row of it seems to be the best optimization technique available in these circumstances.

16.3 CLUSTERING METHOS

16.3.1 Methods and Associated Algorithms

There are a very large number of ways of sorting N objects into M groups, a problem compounded by the fact that M is usually unknown. Most of the possible arrangements are of no interest; it is the role of a clustering method to identify a set of groups or cluster that reflects some underlying structure in the data. Moreover, there are many clustering methods available, which have differing theoretical or empirical bases and therefore produce different cluster structures. For a given clustering *method*, there may be a choice of clustering *algorithm* or means to implement the method. The choice of clustering method will determine the outcome, the choice of algorithm will determine the efficiency with which it is achieved. In this section, an overview of the clustering methods most used in information retrieval will be provided. The associated algorithms that are best suited to the processing of the large data sets found in information retrieval applications are discussed in sections 4.4 and 4.5.

16.3.2 Computational and Storage Requirements

In cases where the data set to be processed is very large, the resources required for cluster analysis may be considerable. A major component of the computation required is the calculation of the document-document or document-cluster similarity. The time requirement will be minimally $O(NM)$, where M is the number of clusters, for the simpler reallocation methods; where the similarity matrix must be constructed, the proportionality is at least N^2. Most of the preferred clustering methods have time requirements of at least $O(N^2)$. The storage requirement will be $O(N)$ if the data set is stored, or $O(N^2)$ if the similarity matrix is stored. For large N this may be unacceptable, and disk accesses may make processing time too large if the similarity matrix is stored on disk. An alternative is to recalculate the similarity matrix from the stored data whenever it is needed to identify the current most similar pair, but this increases the time requirement by a factor of N^2.

Because of the heavy demands of processing and storage requirements, much of the early work on cluster analysis for information retrieval was limited to small data sets, often only a few hundred items. However, improvements in processing and storage capacity and the introduction of efficient algorithms for implementing some clustering methods and finding nearest neighbors have made it feasible to cluster increasingly large data sets. Salton and Bergmark (1981) have pointed out that there is a high degree of parallelism in the calculation

of a set of similarity values, and parallel hardware also offers the potential for increased processing efficiency (Willett and Rasmussen 1990).

16.3.3 Survey of Clustering Methods

Clustering methods are usually categorized according to the type of cluster structure they produce. The simple nonhierarchical methods divide the data set of *N* objects into *M* clusters; where no overlap is allowed, these are known as partitioning methods. Each item has membership in the cluster with which it is most similar, and the cluster may be represented by a centroid or cluster representative that is indicative of the characteristics of the items it contains. The more complex hierarchical methods produce a nested data set in which pairs of items or clusters are successively linked until every item in the data set is connected. The hierarchical methods can be either agglomerative, with $N - 1$ pairwise joins beginning from an unclustered data set, or *divisive*, beginning with all objects in a single cluster and progressing through $N - 1$ divisions of some cluster into a smaller cluster. The divisive methods are less commonly used and few algorithms are available; only agglomerative methods will be discussed in this chapter.

Nonhierarchical methods

The nonhierarchical methods are heuristic in nature, since a priori decisions about the number of clusters, cluster size, criterion for cluster membership, and form of cluster representation are required. Since the large number of possible divisions of *N* items into *M* clusters make an optimal solution impossible, the nonhierarchical methods attempt to find an approximation, usually by partitioning the data set in some way and then reallocating items until some criterion is optimized. The computational requirement $O(NM)$ is much lower than for the hierarchical methods if $M << N$, so that large data sets can be partitioned. The nonhierarchical methods were used for most of the early work in document clustering when computational resources were limited; see for example work on the SMART project, described by Salton (1971).

Hierarchical methods

Most of the early published work on cluster analysis employed hierarchical methods (Blashfield and Aldenderfer 1978), though this was not so in the IR field. With improvements in computer resources, the easy availability of software packages for cluster analysis, and improved algorithms, the last decade of work on clustering in IR retrieval has concentrated on the hierarchical agglomerative clustering methods (HACM, Willett [1988]).

The cluster structure resulting from a hierarchical agglomerative clustering method is often displayed as a *dendrogram* like that shown in Figure 16.2. The order of pairwise coupling of the objects in the data set is shown, and the value of the similarity function (level) at which each fusion occurred. The dendrogram is a useful representation when considering retrieval from a clustered set of documents, since it indicates the paths that the retrieval process may follow.

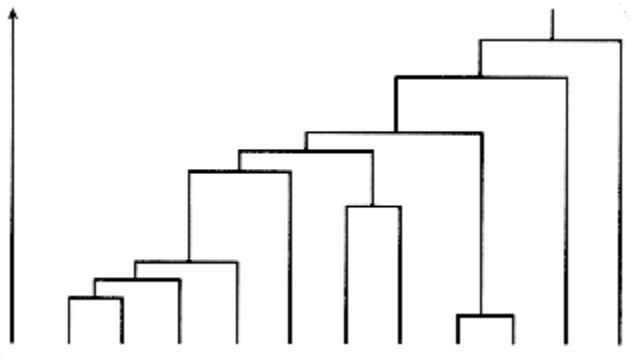


Figure 4.2: Dendrogram of a hierarchical classification

The most commonly used hierarchical agglomerative clustering methods and their characteristics are:

- **single link:** The single link method joins, at each step, the most similar pair of objects that are not yet in the same cluster. It has some attractive theoretical properties (Jardine and Sibson 1971) and can be implemented relatively efficiently, so it has been widely used. However, it has a tendency toward formation of long straggly clusters, or chaining, which makes it suitable for delineating ellipsoidal clusters but unsuitable for isolating spherical or poorly separated clusters.

- **complete link:** The complete link method uses the least similar pair between each of two clusters to determine the intercluster similarity; it is called complete link because all entities in a cluster are linked to one another within some minimum similarity. Small, tightly bound clusters are characteristic of this method.

- **group average link:** As the name implies, the group average link method uses the average values of the pairwise links within a cluster to determine similarity. All objects contribute to intercluster similarity, resulting in a structure intermediate between the loosely bound single link clusters and tightly bound complete link clusters. The group average method has ranked well in evaluative studies of clustering methods (Lorr 1983).

- **Ward's method:** Ward's method is also known as the minimum variance method because it joins at each stage the cluster pair whose merger minimizes the increase in the total within-group error sum of squares, based on the Euclidean distance between centroids. It tends to produce homogeneous clusters and a symmetric hierarchy, and its definition of a cluster center of gravity provides a useful way of representing a cluster. Tests have shown it to be good at recovering cluster structure, though it is sensitive to outliers and poor at recovering elongated clusters (Lorr 1983).

Two other HACM are sometimes used, the *centroid* and *median* methods. In the centroid method, each cluster as it is formed is represented by the coordinates of a group centroid, and at each stage in the clustering the pair of clusters with the most similar mean centroid is merged. The median method is similar but the centroids of the two merging clusters are not weighted proportionally to the size of the clusters. A disadvantage of these two methods is that a newly formed cluster may be more like some point than were its constituent points, resulting in reversals or inversions in the cluster hierarchy.

4.4 ALGORITHMS FOR NONHIERARCHICAL METHODS

4.4.1 Single Pass Methods

The single pass method is particularly simple since it requires that the data set be processed only once. The general algorithm is as follows:

1. Assign the first document D_1 as the representative for C_1**.**

2. For D_i**,** calculate the similarity S with the representative for each existing cluster.

3. If S_{max} is greater than a threshold value S_T, add the item to the corresponding cluster and recalculate the cluster representative; otherwise, use D_i to initiate a new cluster.

4. If an item D_i remains to be clustered, return to step 2.

Though the single pass method has the advantage of simplicity, it is often criticized for its tendency to produce large clusters early in the clustering pass, and because the clusters formed are not independent of the order in which the data set is processed. It is sometimes used to form the groups that are used to initiate reallocation clustering. An example of a single pass algorithm developed for document clustering is the cover coefficient algorithm (Can and Ozkarahan 1984). In this algorithm, a set of documents is selected as cluster seeds, and then each document is assigned to the cluster seed that maximally covers it. For D_i, the cover coefficient is a measure that incorporates the extent to which it is covered by D_j and the uniqueness of D_i, that is, the extent to which it is covered by itself.

4.4.2 Reallocation Methods

The reallocation methods operate by selecting some initial partition of the data set and then moving items from cluster to cluster to obtain an improved partition. Anderberg (1973) discusses some of the criteria that have been suggested to establish an initial partition and to monitor the improvement achieved by reallocation. A general algorithm is:

1. Select M cluster representatives or centroids.

2. For $i = 1$ to N, assign D_i to the most similar centroid.

3. For $j = 1$ to M, recalculate the cluster centroid C_j.

4. Repeat steps 2 and 3 until there is little or no change in cluster membership during a pass through the file.

The single pass and reallocation methods were used in early work in cluster analysis in IR, such as the clustering experiments carried out in the SMART project (Salton 1971). Their time and storage requirements are much lower than those of the HACM and much larger data sets could be processed. With improved processing capability and more efficient hierarchical algorithms, the HACMs are now usually preferred in practice, and the nonhierarchical methods will not be considered further in this chapter.

4.5 ALGORITHMS FOR HIERARCHICAL METHODS

4.5.1 General Algorithm for the HACM

All of the hierarchical agglomerative clustering methods can be described by a general algorithm:

1. Identify the two closest points and combine them in a cluster.

2. Identify and combine the next two closest points (treating existing clusters as points).

3. If more than one cluster remains, return to step 1.

Individual HACM differ in the way in which the most similar pair is defined, and in the means used to represent a cluster. Lance and Williams (1966) proposed a general combinatorial formula, the Lance-Williams dissimilarity update formula, for calculating dissimilarities between new clusters and existing points, based on the dissimilarities prior to formation of the new cluster. If objects C_i and C_j have just been merged to form cluster C_{ij}, the dissimilarity d between the new cluster and any existing cluster C_k is given by:

$$d_{Ci,j}c_k = \alpha_i dc_i c_k + a_j dc_j c_k + \beta dc_i c_j + \gamma | dc_i c_k - dc_j c_k |$$

This formula can be modified to accomodate a variety of HACM by choice of the values of α, β, and γ. The hierarchical clustering methods previously discussed are presented in Table 16.1 in the context of their Lance-Williams parameters and cluster centers.

HACM	Lance-Williams parameters	Cluster centers
Single link	$\alpha_i = \frac{1}{2}$ $\beta = 0$ $\gamma = -\frac{1}{2}$	—
Complete link	$\alpha_i = \frac{1}{2}$ $\beta = 0$ $\gamma = \frac{1}{2}$	—
Group average	$\alpha_i = \frac{m_i}{m_i + m_j}$ $\beta = 0$ $\gamma = 0$	—
Median	$\alpha_i = \frac{1}{2}$ $\beta = -\frac{1}{4}$ $\gamma = 0$	$C_{i,j} = \frac{C_i + C_j}{2}$
Centroid	$\alpha_i = \frac{m_i}{m_i + m_j}$ $\beta = -\frac{m_i m_j}{(m_i + m_j)^2}$ $\gamma = 0$	$C_{i,j} = \frac{m_i C_i + m_j C_j}{m_i + m_j}$
Ward's method	$\alpha_i = \frac{m_i + m_k}{m_i + m_j + m_k}$ $\beta = -\frac{m_k}{m_i + m_j + m_k}$ $\gamma = 0$	$C_{i,j} = \frac{m_i C_i + m_j C_j}{m_i + m_j}$

Table 16.1: Characteristics of HACM

Notes: m_i is the number of items in C_i; the dissimilarity measure used for Ward's method must be the increase in variance (section 4.5.5).

There are three approaches to implementation of the general HACM (Anderberg 1973), each of which has implications for the time and storage requirements for processing. In the *stored matrix* approach, an$N \times N$ matrix containing all pairwise dissimilarity values is stored, and the Lance-Williams update formula makes it possible to recalculate the dissimilarity between cluster centers using only the stored values. The time requirement is $O(N^2)$, rising to $O(N^3)$ if a simple serial scan of the similarity matrix is used; the storage requirement is $O(N^2)$. A *stored data* approach has only an $O(N)$ storage requirement but the need to recalculate the pairwise dissimilarity values for each fusion leads to an $O(N^3)$ time requirement. In the *sorted matrix* approach, the dissimilarity matrix is calculated ($O(N^2)$) and then sorted ($O(N^2 \log N^2)$) prior to the construction of the hierarchy ($O(N^2)$). The data set need not be stored and the similarity matrix is processed serially, which minimizes disk accesses. However, this approach is suitable only for the single link and complete link methods, which do not require the recalculation of similarities during clustering.

In addition to the general algorithm, there are also algorithms specific to individual HACM. For some methods, algorithms have been developed that are the optimal $O(N^2)$ in time

and $O(N)$ in space (Murtagh 1984). These include several algorithms for the single link method, Defay's algorithm for complete link, and a nearest-neighbor algorithm for Ward's method, all of which are discussed below.

16.5.2 Single Link Method

The single link method merges at each stage the closest previously unlinked pair of points in the data set. Since the distance between two clusters is defined as the distance between the closest pair of points each of which is in one of the two clusters, no cluster centroid or representative is required, and there is no need to recalculate the similarity matrix during processing. This makes the method attractive both from a computational and a storage perspective, and it also has desirable mathematical properties (Jardine and Sibson 1971), so that it is one of the most widely used of the HACM.

A number of algorithms for the single link method have been reviewed by Rohlf (1982), including related minimal spanning tree algorithms. The computational requirements range from $O(NlogN)$ to $O(N^5)$. Many of these algorithms are not suitable for information retrieval applications where the data sets have large N and high dimensionality. The single link algorithms discussed below are those that have been found most useful for information retrieval.

Van Rijsbergen algorithm

Van Rijsbergen (1971) developed an algorithm to generate the single link hierarchy that allowed the similarity values to be presented in any order and therefore did not require the storage of the similarity matrix. It is $O(N^2)$ in time and $O(N)$ in storage requirements. It generates the hierarchy in the form of a data structure that both facilitates searching and is easily updated, and was the first to be applied to a relatively large collection of 11,613 documents (Croft 1977). However, most later work with large collections has used either the SLINK or Prim-Dijkstra algorithm, which are quite simple to implement.

SLINK algorithm

The SLINK algorithm (Sibson 1973) is optimally efficient, $O(N^2)$ for computation and $O(N)$ for time, and therefore suitable for large data sets. It is simply a sequence of operations by which a representation of the single link hierarchy can be recursively updated; the dendrogram is built by inserting one point at a time into the representation.

The hierarchy is generated in a form known as the *pointer representation*, which consists of two functions Π and Λ for a data set numbered 1..N, with the following conditions:

- $\Pi(N) = N$
- $\Pi(i) > i$
- $\Lambda(N) = \infty$
- $\Lambda(\Pi(i)) > \Lambda(i)$ for $i < N$

In simple terms, $\Lambda(i)$ is the lowest level (distance or dissimilarity) at which i is no longer the last (i.e., the highest numbered) object in its cluster, and $\Pi(i)$ is the last object in the cluster

it joins at this level; a mathematical definition for these parameters is provided by Sibson (1973).

Fortran code for SLINK is provided in the original paper (Sibson 1973). In the pseudocode below, three arrays of dimension *N* are used: *pi* (to hold the pointer representation), lambda (to hold the distance value associated with each pointer), and distance (to process the current row of the distance matrix); *next* indicates the current pointer for a point being examined.

```
/* initialize pi and lambda for a single point representation */
pi [0] = 0;
lambda[0] = MAXINT;
/*iteratively add the remaining N-1 points to the hierarchy */
for (i = 1; i < N; i++)
{
pi [i] = i;
lambda[i] = MAXINT;
/* calculate and store a row of the distance matrix for i */
for (j = 0; j < i-1; j++) distance[j] =calc_distance(i,j);
for (j = 0; j < i-1; j++)
{
next = pi[j];
if (lambda[j] < distance[j])
distance[next] = min(distance[next],distance[j]);
else
{
distance[next] = min(lambda[j],distance[next]);
pi[j] = i;
lambda[j] = distance[j];
}
}
/* relabel clusters if necessary */
for (j = 0; j <i-1; j++)
{
next = pi  [j];
if (lambda[next] < lambda [j])
pi[j] = i;
}
}
```

For output in the form of a dendrogram, the pointer representation can be converted into the *packed representation*. This can be accomplished in $O(N^2)$ time (with a small coefficient for N^2) and $O(N)$ space. A FORTRAN subroutine to effect the transformation is provided in Sibson's original paper.

Minimal spanning tree algorithms

A minimal spanning tree (MST) is a tree linking *N* objects with *N* - 1 connections so that there are no loops and the sum of the *N* - 1 dissimilarities is minimized. It can be shown that all the information required to generate a single link hierarchy for a set of points is contained in their MST (Gower and Ross 1969). Once an MST has been constructed, the corresponding

single link hierarchy can be generated in $O(N^2)$ operations; or the data structures for the MST can be modified so that the hierarchy can be built simultaneously (Rohlf 1982).

Two fundamental construction principles for MSTs are:

1. Any isolated point can be connected to a nearest neighbor.

2. Any isolated fragment (subset of an MST) can be connected to a nearest neighbor by a shortest available link.

The Prim-Dijkstra algorithm (Dijkstra 1976) consists of a single application of principle 1, followed by N - 1 iterations of principle 2, so that the MST is grown by enlarging a single fragment:

1. Place an arbitrary point in the MST and connect its nearest neighbor to it.

2. Find the point not in the MST closest to any point in the MST and add it to the fragment.

3. If a point remains that is not in the fragment, return to step 2.

The algorithm requires $O(N^2)$ time if, for each point not in the fragment, the identity and distance to its nearest neighbor in the fragment is stored. As each new point is added to the fragment, the distance from that point to each point not in the fragment is calculated, and the NN information is updated if necessary. Since the dissimilarity matrix need not be stored, the storage requirement is $O(N)$.

A FORTRAN version of the Prim-Dijkstra algorithm is provided by Whitney (1972). The algorithm here uses arrays *npoint* and *ndistance* to hold information on the nearest in-tree neighbor for each point, and *notintree* is a list of the *nt* unconnected points. *Lastpoint* is the latest point added to the tree.

```
/*  initialize lists */
for (i = 0; i < n; i++)
{
ndistance[i] = MAXINT;
notintree[i] = i;
}
/* arbitrarily place the Nth point in the MST */
lastpoint = n;
nt = n-1;
/*  grow the tree an object at a time */
for ( i = 0; i < n-1;i++)
{
/*consider the lastpoint in the tree for the NN list */
for (j = 0; j < nt; j++)
{
D = calculate_distance(lastpoint, notintree[j]);
if  (D < ndistance[j]
{
npoint[j] =lastpoint;
ndistance[j] = D;
}
```

```
}
/* find the unconnected point closest to a point in the     */
/* tree                                                     */
nj = index_of_min(ndistance);
/* add this point to the MST; store this point and their    */
/* clustering level                                         */
lastpoint = notintree[nj];
store_in_MST ( lastpoint, npoint[nj], ndistance[nj]);
/* remove lastpoint from notintree list;  */
/* close up npoint and ndistance lists    */
notintree[nj] = nt;
npoint[nj] = npoint[nt];
ndistance[nj] =ndistance[nt];
nt = nt - 1;
}
}
```

4.5.3 Complete Link Method

The small, tightly bound clusters typical of the complete link method have performed well in comparative studies of document retrieval (Voorhees 1986a). Unfortunately, it is difficult to apply to large data sets since there does not seem to be an algorithm more effective than the stored data or stored matrix approach to the general HACM algorithm.

Defays' CLINK algorithm

The best-known algorithm for implementing the complete link method is the CLINK algorithm developed by Defays (1977). It is presented in a form analogous to the SLINK algorithm, uses the same three arrays (*pi*, *lambda*, and *distance*), and like SLINK, produces output in the form of the pointer representation. Defays presents a CLINK subroutine which allows his algorithm to be incorporated into Sibson's original FORTRAN program for SLINK. CLINK is efficient, requiring $O(N^2)$ time, $O(N)$ space, but it does not seem to generate an exact hierarchy and has given unsatisfactory results in some information retrieval experiments (El-Hamdouchi and Willett 1989).

Voorhees algorithm

The Voorhees algorithm (1986b) for the complete link method has been used to cluster relatively large document collections with better retrieval results than the CLINK algorithm (El-Hamdouchi and Willett 1989). It is a variation on the sorted matrix approach, and is based on the fact that if the similarities between all pairs of documents are processed in descending order, two clusters of size m_i and m_j can be merged as soon as the $m_i \times m_i$th similarity of documents in the respective clusters is reached. This requires a sorted list of document-document similarities, and a means of counting the number of similarities seen between any two active clusters. The large number of zero-valued similarities in a typical document collection make it more efficient than its worst case $O(N^3)$ time, $O(N^2)$ storage requirement would suggest; however it is still very demanding of resources, and El-Hamdouchi and Willett found it impractical to apply it to the largest of the collections they studied.

4.5.4 Group Average Link Method

Because the similarity between two clusters is determined by the average value of all the pairwise links between points for which each is in one of the two clusters, no general $O(N^2)$ time, $O(N)$ space algorithm is known. The general HACM algorithm can be used, but with $O(N^3)$ time for the stored data approach and $O(N^2)$ storage for the stored matrix approach, implementation may be impractical for a large collection. However, a more efficient special case algorithm is available.

Voorhees algorithm

Voorhees (1986b) has pointed out that the group average link hierarchy can be constructed in $O(N^2)$ time, $O(N)$ space if the similarity between documents chosen is the inner product of two vectors using appropriately weighted vectors. In this case, the similarity between a cluster centroid and any document is equal to the mean similarity between the document and all the documents in the cluster. Since the centroid of the cluster is the mean of all the document vectors, the centroids can be used to compute the similarities between the clusters while requiring only $O(N)$ space. Voorhees was able to cluster a document collection of 12,684 items using this algorithm, for which she provides pseudocode.

Using Voorhees' weighting scheme and intercentroid similarity, El-Hamdouchi (1987) was able to implement the group average link method using the reciprocal nearest neighbor algorithm described below for Ward's method.

4.5.5 Ward's Method

Ward's method (Ward 1963; Ward and Hook 1963) follows the general algorithm for the HACM, where the object/cluster pair joined at each stage is the one whose merger minimizes the increase in the total within-group squared deviation about the means, or variance. When two points D_i and D_j are clustered, the increase in variance I_{ij} is given by:

$$I_{ij} = \frac{m_i m_j}{m_i + m_j} d_{ij}^2$$

where m_i is the number of objects in D_i and d^2_{ij} is the squared Euclidean distance, given by:

$$d_{ij}^2 = \sum_{k=1}^{L} (x_{ik} - x_{jk})^2$$

where D_i is represented by a vector $(x_i1, x_i2, \ldots, x_{iL})$ in L-dimensional space. The cluster center for a pair of points D_i and D_j is given by:

$$\frac{m_i D_i + m_j D_j}{m_i + m_j}$$

Reciprocal nearest neighbor algorithm

The mathematical properties of Ward's method make it a suitable candidate for a reciprocal nearest neighbor (RNN) algorithm (Murtaugh 1983, 1985). For any point or cluster, there exists a chain of nearest neighbors (NNs) so that

NN(i) = j; NN(j) = k; ...; NN(p) = q; NN(q) = p

The chain must end in some pair of objects that are RNNs, since the interobject distances are monotonically decreasing along the chain (ties must be arbitrarily resolved).

An efficient clustering algorithm can be based on the process of following a chain of nearest neighbors:

1. Select an arbitrary point.

2. Follow the NN chain from this point till an RNN pair is found.

3. Merge these two points and replace them with a single point.

4. If there is a point in the NN chain preceding the merged points, return to step 2; otherwise return to step 1. Stop when only one point remains.

This algorithm requires $O(N^2)$ computation but only $O(N)$ storage. It carries out agglomerations in restricted spatial regions, rather than in strict order of increasing dissimilarity, but still results (for Ward's method) in a hierarchy that is unique and exact. This is designated the Single Cluster Algorithm since it carries out one agglomeration per iteration; a Multiple Cluster Algorithm, suitable for parallel processing, has also been proposed (Murtagh 1985).

4.6 EVALUATION AND VALIDATION

As Dubes and Jain (1980, p. 179) point out:

The thoughtful user of a clustering method or algorithm must answer two questions: (i) Which clustering method is appropriate for a particular data set? (ii) How does one determine whether the results of a clustering method truly characterize the data?

These are important questions because any clustering method will produce a set of clusters, and results are sought which reflect some "natural" grouping rather than one that arises as an artifact of the method. The answer to the first question can be found in evaluative studies of clustering methods, and to the second question, in validation techniques for clustering solutions.

4.6.1 Evaluation

Many evaluative studies have attempted to determine the "best" clustering method (Lorr 1983) by applying a range of clustering methods to test data sets and comparing the quality of the results, for example by using artificially created data structures, or comparing the cluster results to a classification established by experts in the field. Even under laboratory conditions it is difficult to evaluate clustering methods, since each method has different properties and strengths. The results of these studies do not suggest a single best method,

though Ward's method, and in more recent studies, the group average method, have performed well. It is usually advisable to apply more than one clustering method and use some validation method to check the reliability of the resulting cluster structure.

For retrieval purposes, the "best" method for clustering a document collection is that which provides the most effective retrieval in response to a query. Several evaluative studies have taken this approach, using standard test collections of documents and queries. Most of the early work used approximate clustering methods, or the least demanding of the HACM, the single link method, a restriction imposed by limited processing resources. However, two recent studies are noteworthy for their use of relatively large document collections for the evaluation of a variety of HACM (El-Hamdouchi and Willett 1989; Voorhees 1986a). Voorhees compared single link, complete link, and group average methods, using document collections of up to 12,684 items, while El-Hamdouchi and Willett compared these three methods plus Ward's method on document collections of up to 27,361 items. Voorhees found complete link most effective for larger collections, with complete and group average link comparable for smaller collections; single link hierarchies provided the worst retrieval performance. El-Hamdouchi and Willett found group average most suitable for document clustering. Complete link was not as effective as in the Voorhees study, though this may be attributed to use of Defays' CLINK algorithm. As noted in section 16.8.1, there are several ways in which retrieval from a clustered document collection can be performed, making comparisons difficult when using retrieval as an evaluative tool for clustering methods.

4.6.2 Validation

Cluster validity procedures are used to verify whether the data structure produced by the clustering method can be used to provide statistical evidence of the phenomenon under study. Dubes and Jain (1980) survey the approaches that have been used, categorizing them on their ability to answer four questions: is the data matrix random? how well does a hierarchy fit a proximity matrix? is a partition valid? and which individual clusters appearing in a hierarchy are valid? Willett (1988) has reviewed the application of validation methods to clustering of document collections, primarily the application of the random graph hypothesis and the use of distortion measures.

An approach that is carried out prior to clustering is also potentially useful. Tests for *clustering tendency* attempt to determine whether worthwhile retrieval performance would be achieved by clustering a data set, before investing the computational resources which clustering the data set would entail. El-Hamdouchi and Willett (1987) describe three such tests. The *overlap test* is applied to a set of documents for which query-relevance judgments are available. All the relevant-relevant (RR) and relevant-nonrelevant (RNR) interdocument similarities are calculated for a given query, and the overlap (the fraction of the RR and RNR distributions that is common to both) is calculated. Collections with a low overlap value are expected to be better suited to clustering than those with high overlap values. Voorhees' *nearest neighbor test* considers, for each relevant document for a query, how many of its *n* nearest neighbors are also relevant; by averaging over all relevant documents for all queries in a test colleciton, a single indicator for a collection can be obtained. The *density test* is defined as the total number of postings in the document collection divided by the product of the number of documents and the number of terms that have been used for the indexing of those documents. It is particularly useful because it does not require any predetermined query-relevance data or any calculation of interdocument similarities. Of the three tests, the density test provided the best indication of actual retrieval performance from a clustered data set.

The goal of a hierarchical clustering process may be to partition the data set into some unknown number of clusters *M* (which may be visualized as drawing a horizontal line across the dendrogram at some clustering level). This requires the application of a *stopping rule*, a statistical test to predict the clustering level that will determine *M*. Milligan and Cooper (1985) evaluated and ranked 30 such rules, one or more of which is usually present in a software package for cluster analysis (though not necessarily those ranked highest by Milligan and Cooper).

4.7 UPDATING THE CLUSTER STRUCTURE

In many information retrieval environments, the collection is dynamic, with new items being added on a regular basis, and, less frequently, old items withdrawn. Since clustering a large data set is resource-intensive, some mechanism for updating the cluster structure without the need to recluster the entire collection is desirable. Relatively little work has been done on methods for cluster maintenance (Can and Ozkarahan 1989), particularly for the hierarchical methods. In certain cases, update of the cluster structure is implicit in the clustering algorithm. This is true of both the van Rijsbergen and SLINK algorithms for the single link method, and the CLINK algorithm for the complete link method, all of which operate by iteratively inserting a document into an existing hierarchy.

Where the application uses a partitioned data set (from a nonhierarchical or hierarchical method), new items may simply be added to the most similar partition until the cluster structure becomes distorted and it is necessary to regenerate it. For a few methods, cluster update has been specifically incorporated. Crouch's reallocation algorithm includes a mechanism for cluster maintenance (Crouch 1975). Can and Ozkarahan (1989) review the approaches that have been taken for cluster maintenance and propose a strategy for dynamic cluster maintenance based on their cover coefficient concept.

4.8 DOCUMENT RETRIEVAL FROM A CLUSTERED DATA SET

Document clustering has been studied because of its potential for improving the efficiency of retrieval, for improving the effectiveness of retrieval, and because it provides an alternative to Boolean or best match retrieval. Initially the emphasis was on efficiency: document collections were partitioned, using nonhierarchical methods, and queries were matched against cluster centroids, which reduced the number of query-document comparisons that were necessary in a serial search. Studies of retrieval from partitioned document collections showed that though retrieval efficiency was achieved, there was a decrease in retrieval effectiveness (Salton 1971). Subsequent study has concentrated on the effectiveness of retrieval from hierarchically clustered document collections, based on the *cluster hypothesis*, which states that associations between documents convey information about the relevance of documents to requests (van Rijsbergen 1979).

4.8.1 Approaches to Retrieval

There are several ways in which a query can be matched against the documents in a hierarchy (Willett 1988). A *top-down search* involves entering the tree at the root and matching the query against the cluster at each node, moving down the tree following the path of greater similarity. The search is terminated according to some criterion, for instance when the cluster size drops below the number of documents desired, or when the query-cluster similarity begins to decrease. A single cluster is retrieved when the search is terminated. Since it is difficult to adequately represent the clusters in the very large top-level clusters, a useful modification is to eliminate the top-level clusters by applying a threshold clustering level to

the hierarchy to obtain a partition, and using the best of these mid-level clusters as the starting point for the top-down search. The top-down strategy has been shown to work well with the complete link method (Voorhees 1986a).

A *bottom-up search* begins with some document or cluster at the base of the tree and moves up until the retrieval criterion is satisfied; the beginning document may be an item known to be relevant prior to the search, or it can be obtained by a best match search of documents or lowest-level clusters. Comparative studies suggest that the bottom-up search gives the best results (apart from the complete link method), particularly when the search is limited to the bottom-level clusters (Willett 1988). Output may be based on retrieval of a single cluster, or the top-ranking clusters may be retrieved to produce a predetermined number of either documents or clusters; in the latter case, the documents retrieved may themselves be ranked against the query.

A simple retrieval mechanism is based on *nearest neighbor clusters*, that is, retrieving a document and that document most similar to it. Griffiths et al. (1984) determined that for a variety of test collections, search performance comparable to or better than that obtainable from nonclustered collections could be obtained using this method.

4.8.2 Cluster Representatives

A centroid or cluster representative is a record that is used to represent the characteristics of the documents in a cluster. It is required in retrieval so that the degree of similarity between a query and a cluster can be determined; it is also needed in the nonhierarchical methods where document-cluster similarity must be determined in order to add documents to the most similar cluster. Ranks or frequencies have been used to weight terms in the representative; usually a threshold is applied to eliminate less significant terms and shorten the cluster representative. A binary representative may also be used, for example including a term if it occurs in more than $log_2 m$ documents in the cluster (Jardine and van Rijsbergen 1971).

UNIT-5

- **Search Statements and Binding**
- **Similarity Measures and Ranking**
- **Relevance Feedback**
- **Selective Dissemination of Information Search**
- **Weighted Searches of Boolean Systems**
- **Searching the INTERNET and Hypertext**

- **Systemic approach**

– User outside the system

– Static/fixed information need

– Retrieval effectiveness measured

– Batch retrieval **simulations**

- **User-centered approach**

– User part of the system, interacting with other components, trying to resolve an anomalous state of knowledge

– Task-oriented evaluation

Search Statements and Binding Search statements are the statements of an information need generated by users to specify the concepts they are trying to locate in items. As discussed in Chapter 2, the search statement use traditional Boolean logic and/or Natural Language. In generation of the search statement, the user may have the ability to weight (assign an importance) to different concepts in the statement. At this point the binding is to the vocabulary and past experiences of the user.

Binding in this sense is when a more abstract form is redefined into a more specific form. The search statement is the user's attempt to specify the conditions needed to subset logically the total item space to that cluster of items that contains the information needed by the user.

The next level of binding comes when the search statement is parsed for use by a specific search system. The search system translates the query to its own meta language.

This process is similar to the indexing of item processes described in Chapter 5. For example, statistical systems determine the processing tokens of interest and the weights assigned to each processing token based upon frequency of occurrence from the search statement.

Natural language systems determine the syntactical and discourse semantics using algorithms similar to those used in indexing. Concept systems map the search statement to the set of concepts used to index items.

The final level of binding comes as the search is applied to a specific database. This binding is based upon the statistics of the processing tokens in the database and the semantics used in the database.

This is especially true in statistical and concept indexing systems. Some of the statistics used in weighting are based upon the current contents of the database. Some examples are Document Frequency and Total Frequency for a specific term.

Frequently in a concept indexing system, the concepts that are used as the basis for indexing are determined by applying a statistical algorithm against a representative sample of the database versus being generic across all databases (see Chapter 5).

Natural Language indexing techniques tend to use the most corpora-independent algorithms. Figure 7.1 illustrates the three potential different levels of binding. Parenthesis are used in the second binding step to indicate expansion by a thesaurus.

The length of search statements directly affect the ability of Information Retrieval Systems to find relevant items. The longer the search query, the easier it is for the system to find items. Profiles used as search statements for Selective Dissemination of Information systems are usually very long, typically 75 to 100 terms.

In large systems used by research specialists and analysts, the typical adhoc search statement is approximately 7 terms. In a paper to be published in SIGIR-97, Fox et al. at Virginia Tech have noted that the typical search statement on the Internet is one or two words. These extremely short search statements for **Search Statements**

INPUT	Binding
"Find me information on the impact of the oil spills in Alaska on the price of oil"	User search statement using vocabulary of user
impact, oil (petroleum), spills (accidents), Alaska, price (cost, value)	Statistical system binding extracts processing tokens
impact (.308), oil (.606), petroleum (.65), spills (.12), accidents (.23), Alaska (.45), price (.16), cost (.25), value (.10)	Weights assigned to search terms based upon inverse document frequency algorithm and database

Figure 7.1 Examples of Query Binding

- The length of search statements directly affect the ability of Information Retrieval Systems to find relevant items.
- The longer the search query, the easier it is for the system to find items.
- Selective Dissemination of Information systems are usually very long, typically 75 to 100 terms.

Similarity Measures and Ranking

- Searching in general is concerned with calculating the similarity between a user's search statement and the items in the database.
- Restricting the similarity measure to passages gains significant precision with minimal impact on recall.
- Once items are identified as possibly relevant to the user's query, it is best to present the most likely relevant items first- Ranking is a scalar number that represents how similar an item is to the query.
- Once items are identified as possibly relevant to the user's query, it is best to present the most likely relevant items first. This process is called "ranking."
- Usually the output of the use of a similarity measure in the search process is a scalar number that represents how similar an item is to the query.

Similarity measure

A variety of different similarity measures can be used to calculate the similarity between the item and the search statement. A characteristic of a similarity formula is that the results of the formula increase as the items become more similar. The value is zero if the items are totally dissimilar. An example of a simple "sum of the products" similarity measure from the examples in Chapter 6 to determine the similarity between documents for clustering purposes is:

SIM(Itemi, Itemj) = Z(Termix) (Termja)

This formula uses the summation of the product of the various terms of two items when treating the index as a vector. If Itemj is replaced with Queryj then the same formula generates the similarity between every Item and Queryj. The problemwith this simple measure is in the normalization needed to account for variances in the length of items. Additional normalization is also used to have the final results come between zero and +1 (some formulas use the range -1 to +1).

One of the originators of the theory behind statistical indexing and similarity functions was Robertson and Spark Jones (Robertson-76). Their model suggests that knowledge of terms in relevant items retrieved from a query should adjust the weights of those terms in the weighting process. They used the number of relevant documents versus the number of non-relevant documents in the database and the number of relevant documents having a specific query term versus the number of non-relevant documents having that term to devise Ibur formulas for weighting. This assumption of the availability of relevance information in the weighting process was later relaxed by Croft and Harper (Croft-79). Croft expanded this original concept, taking into account the frequency of occurrence of terms within an item producing the following similarity formula (Croft-83):

Similarity formula by Salton in SMART system

- To determine the "weight" an item has with respect to the search statement, the Cosine formula is used to calculate the distance between the vector for the item and the vector for the query:

$$SIM(DOC_i, QUERY_j) = \frac{\sum_{k=1}^{n} (DOC_{i,k} * QTERM_{j,k})}{\sqrt{\sum_{k=1}^{n} (DOC_{i,k})^2 * \sum_{k=1}^{n} (QTERM_{j,k})^2}}$$

The Jaccard formula is:

- The denominator becomes depend uponthe no of terms in common.
- Common elements common increase, the similarity value quickly decreases, in the range -1 and +1.

$$SIM(DOC_i, QUERY_j) = \frac{\sum_{k=1}^{n} (DOC_{i,k} * QTERM_{j,k})}{\sum_{k=1}^{n} DOC_{i,k} + \sum_{k=1}^{n} QTERM_{j,k} - \sum_{k=1}^{n} (DOC_{i,k} * QTERM_{j,k})}$$

The Dice:

- measure simplifies the denominator from the Jaccard measure and introduces a factor of 2 in the numerator.
- The normalization in the Dice formula is also invariant to the number of terms in common.

$$SIM(DOC_i, QUERY_j) = \frac{2 * \sum_{k=1}^{n} (DOC_{i,k} * QTERM_{j,k})}{\sum_{k=1}^{n} DOC_{i,k} + \sum_{k=1}^{n} QTERM_{j,k}}$$

- Use of a similarity algorithm returns the complete data base as search results.
- Many of the items have a similarity close or equal to zero.
- Thresholds(default is the similarity > zero) are usually associated with the search process.
- The threshold defines the items in the resultant Hit file from the query.

- Thresholds are either a value that the similarity measure must equal or exceed or a number that limits the number of items in the Hit file.

Normalizing denominator results vary with commonality of terms

QUERY = (2, 2, 0, 0, 4)
DOC1 = (0, 2, 6, 4, 0)
DOC2 = (2, 6, 0, 0, 4)

	Cosine	Jaccard	Dice
DOC1	36.66	16	20
DOC2	36.66	–12	20

Figure 7.2 Normalizing Factors for Similarity Measures

If threshold = 4 only Doc1 is selected
If threshold = 4 all selected

Vector:	American, geography, lake, Mexico, painter, oil, reserve, subject
DOC1	geography *of* Mexico *suggests* oil reserves *are available* vector (0, 1, 0, 2, 0, 3, 1, 0)
DOC2	American geography *has* lakes *available everywhere* vector (1, 3, 2, 0, 0, 0, 0, 0)
DOC3	painters *suggest* Mexico lakes *as* subjects vector (0, 0, 1, 3, 3, 0, 0, 2)
QUERY	oil reserves *in* Mexico vector (0, 0, 0, 1, 0, 1, 1, 0)

SIM(Q, DOC1) = 6, SIM (Q, DOC2) = 0, SIM(Q, DOC3) = 3

Figure 7.3 Query Threshold Process

The items are stored in clusters.(TOP-DOWN) A, B, C, I – centroids amd K to N, P, Q, R-ITEMS

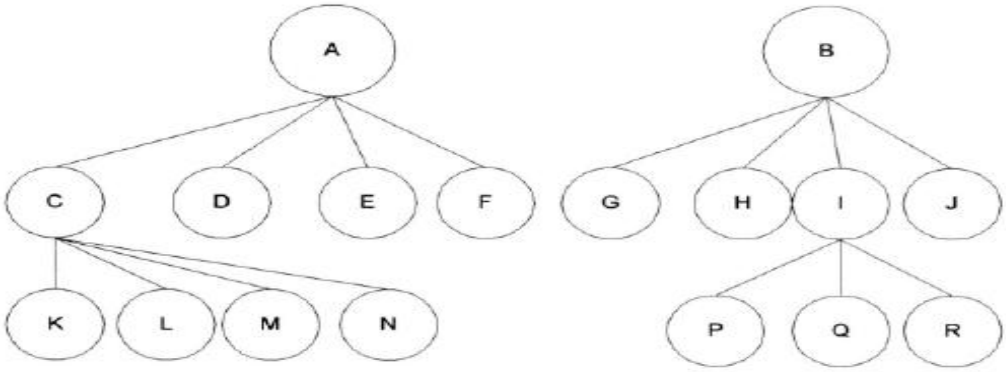


Figure 7.4 Item Cluster Hierarchy

- The query is compared to the centroids "A" and "B." If the results of the similarity measure are above the threshold, the query is then applied to the nodes' children.
- The filled circle represents the query and the filled boxes represent the centroids for the three clusters represented by the ovals.

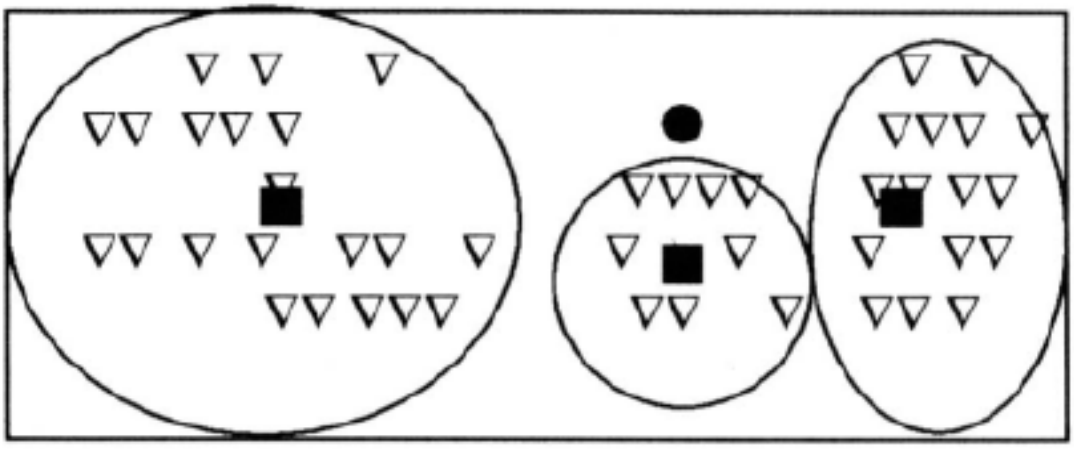


Figure 7.5 Centroid Comparisons

Relevance Feedback

- Thesuari and semantic networks provide utility in generally expanding a user's search statement to include potential related search terms.
- But this still does not correlate to the vocabulary used by the authors that contributes to a particular database.
- There is also a significant risk that the thesaurus does not include the latest jargon being used, acronyms or proper nouns.
- In an interactive system, users can manually modify an inefficient query or have the system automatically expand the query via a thesaurus.

- Relevant items (or portions of relevant items) are used to reweight the existing query terms and possibly expand the user's search statement with new terms.
- The relevance feedback concept was that the new query should be based on the old query modified to increase the weight of terms in relevant items and decrease the weight of terms that are in non-relevant items.

  The formula used is:

$$Q_n = Q_0 + \frac{1}{r}\sum_{i=1}^{r} DR_i - \frac{1}{nr}\sum_{j=1}^{nr} DNR_j$$

where

Q_n	= the revised vector for the new query
Q_0	= the original query
r	= number of relevant items
DR_i	= the vectors for the relevant items
nr	= number of non-relevant items
DNR_j	= the vectors for the non-relevant items.

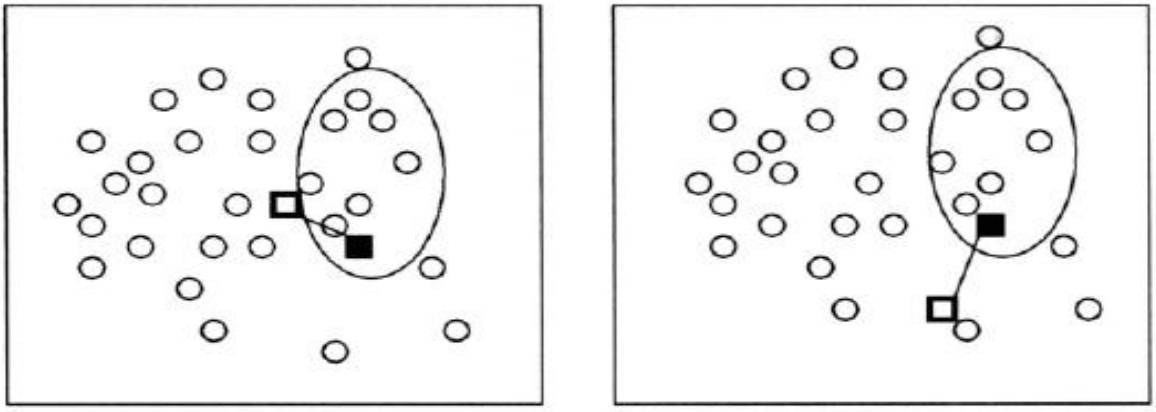


Figure 7.6 Impact of Relevance Feedback

	Term 1	Term 2	Term 3	Term 4	Term 5
Q_o	3	0	0	2	0
$DOC1_r$	2	4	0	0	2
$DOC2_r$	1	3	0	0	0
$DOC3_{nr}$	0	0	4	3	3
Q_n	3¾	1¾	0	1¼	0

Figure 7.7 Query Modification via Relevance Feedback

	DOC1	DOC2	DOC3
Q_o	6	3	6
Q_n	14½	9.0	3.75

$$Q_n = (3, 0, 0, 2, 0) + ¼ (2+1, 4+3, 0+0, 0+0, 2+0) - ¼ (0, 0, 4, 3, 2)$$
$$= (3¾, 1¾, 0\ \{-1\}, 1¼, 0)$$

Selective Dissemination of Information Search

- Selective Dissemination of Information, frequently called dissemination systems, are becoming more prevalent with the growth of the Internet.
- A dissemination system is sometimes labeled a "**push**" system while a search system is called a "**pull**" system.
- In a dissemination system, the user defines a profile and as new info is added to the system it is automatically compared to the user's profile.
- If it is considered a match, it is asynchronously sent to the user's "**mail**" file.
- The differences between the two functions lie in the dynamic nature of the profiling process, the size and diversity of the search statements and number of simultaneous searches per item.
- In the search system, an existing database exists.
- As such, corpora statistics exist on term frequency within and between terms.
- These can be used for weighting factors in the indexing process and the similarity comparison (e.g., inverse document frequency algorithms).
- Profiles are relatively static search statements that cover a diversity of topics.
- Rather than specifying a particular information need, they usually generalize all of the potential information needs of a user.

- One of the first commercial search techniques for dissemination was the Logicon Message Dissemination System (LMDS). The system originated from a system created by Chase, Rosen and Wallace (CRW Inc.). It was designed for speed to support the search of thousands of profiles with items arriving every 20 seconds.
- It demonstrated one approach to the problem where the profiles were treated as the static database and the new item acted like the query.
- It uses the terms in the item to search the profile structure to identify those profiles whose logic could be satisfied by the item.
- The system uses a least frequently occurring trigraph (three character) algorithm that quickly identifies which profiles are not satisfied by the item.
- The potential profiles are analyzed in detail to confirm if the item is a hit.
- Another example of a dissemination approach is the Personal Library Software (PLS) system. It uses the approach of accumulating newly received items into the database and periodically running user's profiles against the database.
- This makes maximum use of the retrospective search software but loses near real time delivery of items.
- Another approach to dissemination uses a statistical classification technique and explicit error minimization to determine the decision criteria for selecting items for a particular profile.
- Schutze et al. used two approaches to reduce the dimensionality: selecting a set of existing features to use or creating a new much smaller set of features that the original features are mapped into.
- A x^2 measure was used to determine the
- most important features. The test was applied to a table that contained the number
- of relevant(N_r)and non-relevant($N^{\eta}{}_r$) items in which a term occurs plus the number of relevant and non-relevant items in which the term does not occur respectively).
- The formula used was:

$$\chi^2 = \frac{N(N_r N_{nr-} - N_{r-} N_{nr})^2}{(N_r + N_{r-})(N_{nr} + N_{nr-})(N_r + N_{nr})(N_{r-} + N_{nr-})}$$

Weighted Searches of Boolean Systems

- The two major approaches to generating queries are Boolean and natural language.
- Natural language queries are easily represented within statistical models and are usable by the similarity measures discussed.

Introduction to Information Visualization

Cognition and Perception

- The user-machine interface has primarily focused on a paradigm of a typewriter.
- As computers displays became ubiquitous, man-machine interfaces focused on treating the display as an extension of paper with the focus on consistency of operations.
- The advent of WIMP interfaces and the evolution of the interface focused on how to represent to the user what is taking place in the computer environment.
- Extending the HCI to improve the information flow, thus reducing wasted user overhead in locating needed information.
- Although the major focus is on enhanced visualization of information, other senses are also being looked at for future interfaces.
- The audio sense has always been part of simple alerts in computers.
- The sounds are now being replaced by speech in both input and output interfaces.
- The tactile (touch) sense is being addressed in the experiments using Virtual Realty (VR).
- A significant portion of the brain is devoted to vision and supports the maximum information transfer function from the environment to a human being.
- Visualization is the transformation of information into a visual form which enables the user to observe and understand the information.
- The Gestalt psychologists postulate that the mind follows a set of rules to combine the input stimuli to a mental representation that differs from the sum of the individual inputs (Rock-90):
- **Proximity** - nearby figures are grouped together
- **Similarity** - similar figures are grouped together
- **Continuity** - figures are interpreted as smooth continuous patterns rather than discontinuous concatenations of shapes (e.g., a circle with its diameter drawn is perceived as two continuous shapes, a circle and a line, versus two half circles concatenated together)
- **Closure** - gaps within a figure are filled in to create a whole (e.g., using dashed lines to represent a square does not prevent understanding it as a square)
- **Connectedness** - uniform and linked spots, lines or areas are perceived as a single unit.
- Shifting the information processing load from slower cognitive processes to faster perceptual systems significantly improves the information-carrying interfaces between humans and computers (Card-96).

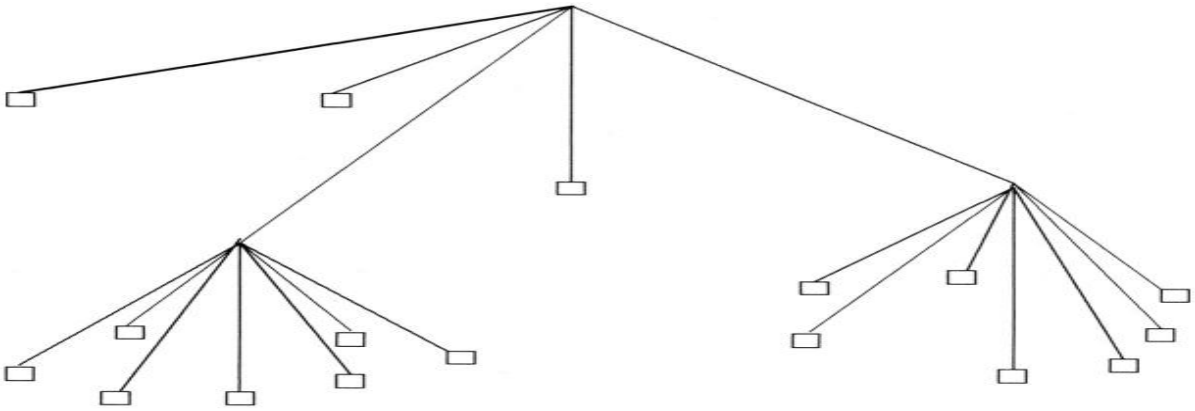
Aspects of the Visualization Process

- One of the first-level cognitive processes is pre attention, that is, taking the significant visual information from the photoreceptors and forming primitives.

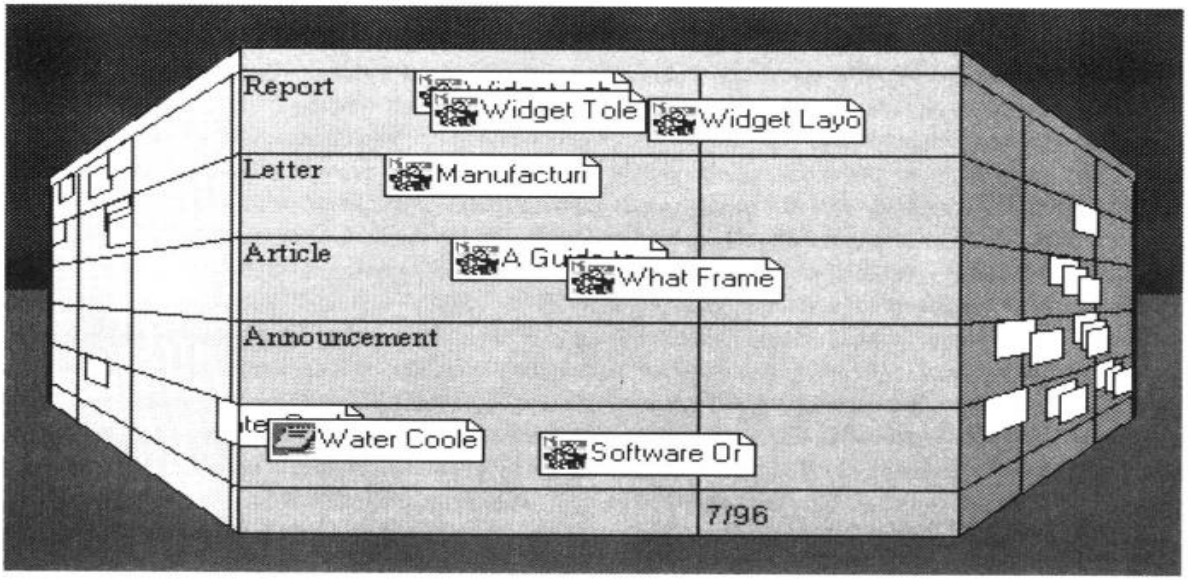
- An example of using the conscious processing capabilities of the brain is the detection of the different shaped objects and the border between them shown between the left side and middle of the Figure.
- The reader can likely detect the differences in the time it takes to visualize the two different boundaries.
- This suggests that if information semantics are placed in orientations, the mind's clustering aggregate function enables detection of groupings easier than using different objects.

Information Visualization Technologies

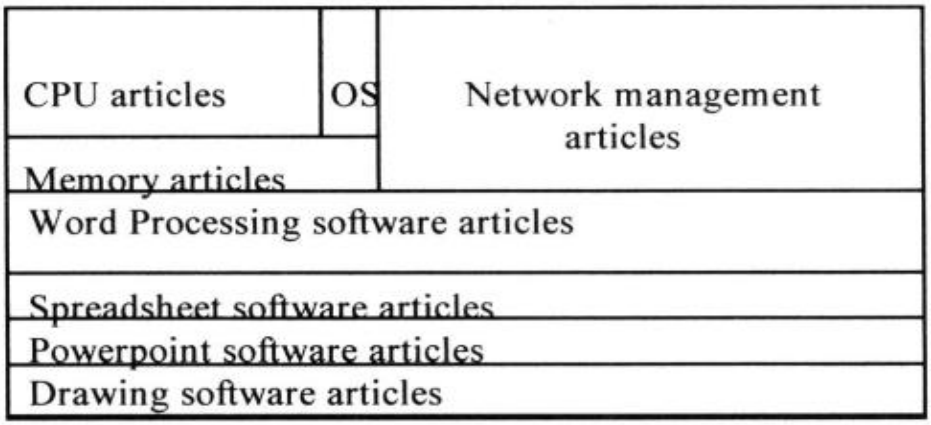
- The ones focused on Information Retrieval Systems are investigating how best to display the results of searches, structured data from DBMSs and the results of link analysis correlating data.
- The goals for displaying the result from searches fall into two major classes: document clustering and search statement analysis.
- Visualization tools in this area attempt to display the clusters, with an indication of their size and topic, as a basis for users to navigate to items of interest.
- Displaying the total set of terms, including additional terms from relevance feedback or thesaurus expansion, along with documents retrieved and indicate the importance of the term to the retrieval and ranking process.
- One way of organizing information is **hierarchical**.
- A **tree structure** is useful in representing information that ranges over time (e.g., genealogical lineage), constituents of a larger unit (e.g., organization structures, mechanical device definitions) and aggregates from the higher to lower level (e.g., hierarchical clustering of documents).
- A two-dimensional representation becomes difficult for a user to understand as the hierarchy becomes large.
- One of the earliest experiments in information visualization was the Information Visualizer developed by XEROX PARC.
- It incorporates various visualization formats such as DataMap, InfoGrid, ConeTree, and the Perspective wall.
- The **Cone-Tree** is a 3-Dimensional representation of data, where one node of the tree is represented at the apex and ail the information subordinate to it is arranged in a circular structure at its base.
- Any child node may also be the parent of another cone.
- Selecting a particular node, rotates it to the front of the display.



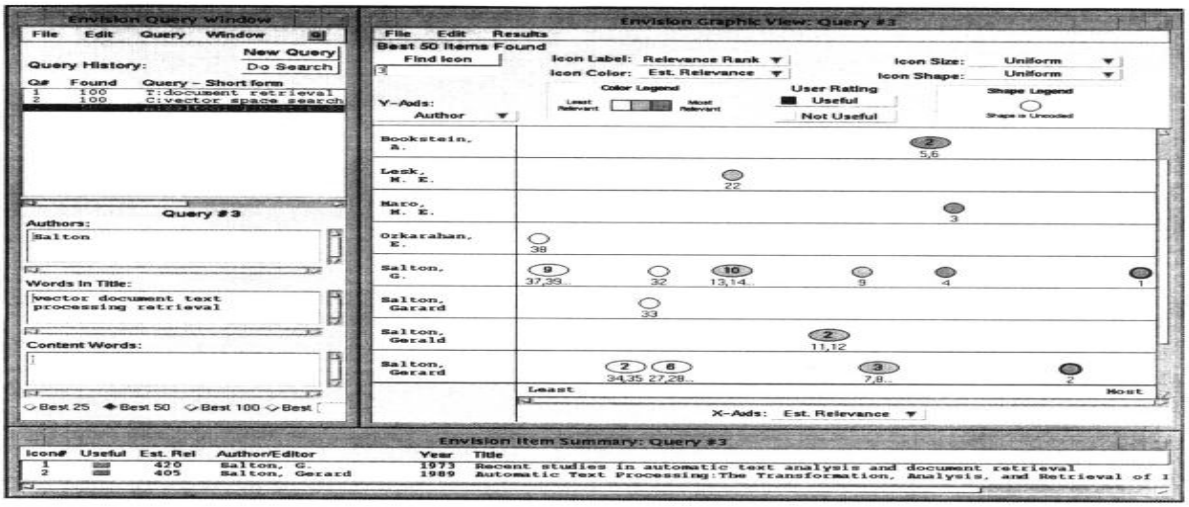
- The **perspective wall** divides the information into three visual areas with the area being focused on in the front and other areas out of focus to each.
- This allows the user to keep all of the information in perspective while focusing on a particular area.



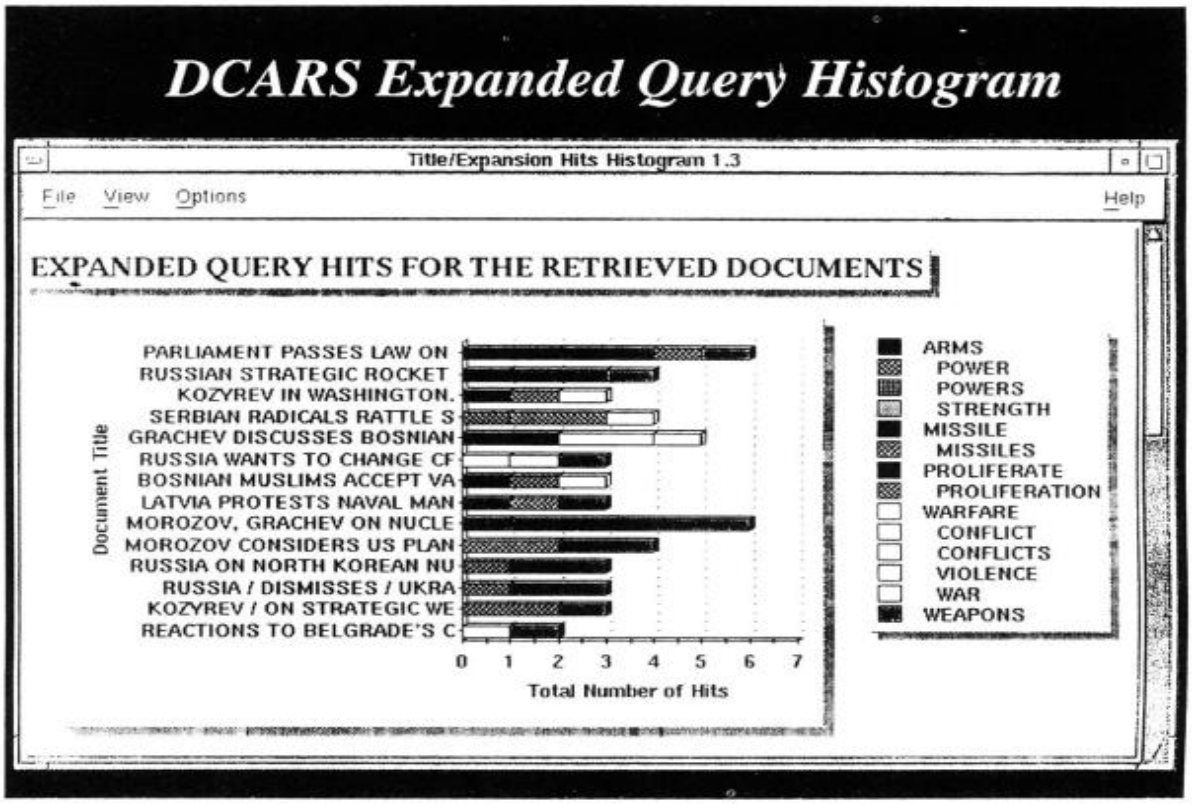
- **Tree maps** (Johnson-91).
- This technique makes maximum use of the display screen space by using rectangular boxes that are recursively subdivided based upon parent-child relationships between the data.
- The CPU, OS, Memory, and Network management articles are all related to a general category of computer operating systems versus computer applications which are shown in the rest of the figure.



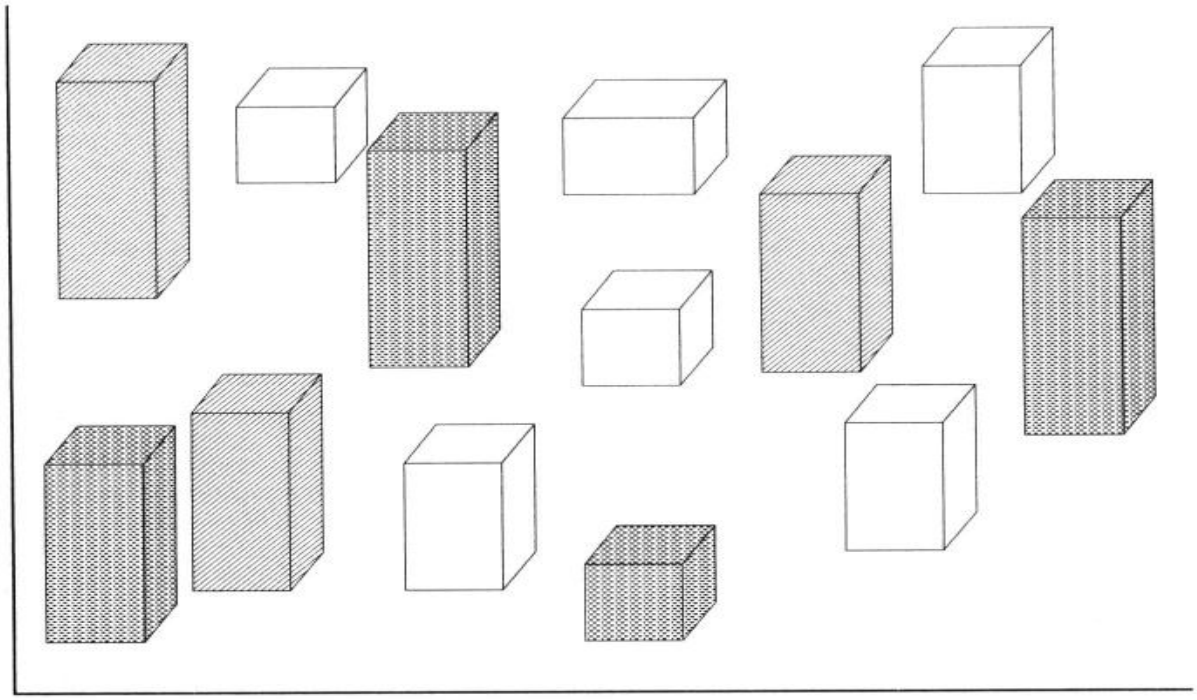
- The **Envision system** not only displays the relevance rank and estimated relevance of each item found by a query, but also simultaneously presents other query information.
- The design is intentionally graphical and simple using two dimensional visualization.
- This allows a larger variety of user computer platforms to have access to their system (Nowell-96).
- Envision's three interactive windows to display search results: Query window, Graphic View window, and Item Summary window.



- Document Content Analysis and Retrieval System (DCARS) being developed by Calspan Advanced Technology Center.
- Their system is designed to augment the Retrieval Ware search product.
- They display the query results as a histogram with the items as rows and each term's contribution to the selection indicated by the width of a tile bar on the row.
- DCARS provides a friendly user interface that indicates why a particular item was found, but it is much harder to use the information in determining how to modify search statements to improve them.



- Another representation that is widely used for both hierarchical and network related information is the "cityscape" which uses the metaphor of movement within a city.
- In lieu of using hills, as in the terrain approach, skyscrapers represent the theme (concept) areaas



UNIT-6

TEXT SEARCH ALGORITHM

Searching in Compressed Text (Overview)

- What is Text Compression
- Definition The Shannon Bound Huffman Codes The Kolmogorov Measure Searching in Non-adaptive Codes KMP in Huffman Codes
- Searching in Adaptive Codes
- The Lempel-Ziv Codes Pattern Matching in Z-Compressed Files Adapting Compression for Searching

THE TECHNIQUES ARE

Three classical text retrieval techniques have been defined for organizing items in a textual database, for rapidly identifying the relevant items

The techniques are

- Full text scanning(streaming)
- Word inversion
- Multi attribute retrieval
- Text streaming architecture

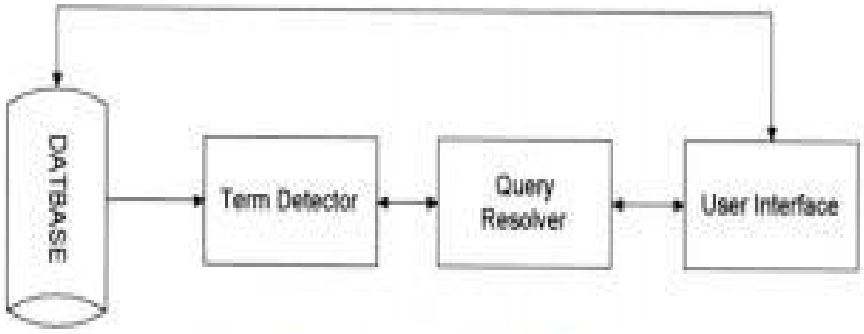


Figure 9.1 Text Streaming Architecture

Software Text Search Algorithms

- In software streaming techniques, the item to be searched is read into memory and then the algorithm is applied.
- There are four major algorithms associated with software text search:
- Brute force approach
- Knuth-Morris-Pratt

- Boyer-Moore
- Shift-OR algorithm
- Rabin -karp

Brute Force :

This approach is the simplest string matching algorithm. The idea is to try and match the search string against the input text. If as soon as a mis-match is detected in the comparison process , shift the input text one position and start the comparison process over.

KNUTH-MORRIS PRATT

Even in the worst case it does not depend upon the length of the text pattern being searched for. The basic concept behind the algorithm is that whenever a mismatch is detected , the previous matched characters define the number of characters that can be skipped in the input stream prior to process again . Starting the comparison

Position :1 2 3 4 5 6 7 8

Input stream a b d a d e f g

Search pattern a b d f

Worked example of the search algorithm[edit]

To illustrate the algorithm's details, consider a (relatively artificial) run of the algorithm, where W = "ABCDABD" and S = "ABC ABCDAB ABCDABCDABDE". At any given time, the algorithm is in a state determined by two integers:

- m, denoting the position within S where the prospective match for W begins,
- i, denoting the index of the currently considered character in W.

In each step the algorithm compares S[m+i] with W[i] and advances i if they are equal. This is depicted, at the start of the run, like

```
          1       2
m: 012345678901234567890 12
S: ABCABCDAB ABCDABCDABDE
W: ABCDABD
i: 0123456
```

The algorithm compares successive characters of W to "parallel" characters of S, moving from one to the next by incrementing i if they match. However, in the fourth step S[3] = ' ' does not match W[3] = 'D'. Rather than beginning to search again at S[1], we note that no 'A' occurs between positions 1 and 2 in W; hence, having checked all those characters previously (and knowing they matched the corresponding characters in S), there

is no chance of finding the beginning of a match. Therefore, the algorithm sets `m = 3` and `i = 0`.

```
          1       2
m: 012345678901234567890 12
S: ABCABCDAB ABCDABCDABDE
W:    ABCDABD
i:    0123456
```

This match fails at the initial character, so the algorithm sets `m = 4` and `i = 0`

```
          1       2
m: 012345678901234567890 12
S: ABC ABCDABABCDABCDABDE
W:     ABCDABD
i:     0123456
```

Here `i` increments through a nearly complete match `"ABCDAB"` until `i = 6` giving a mismatch at `W[6]` and `S[10]`. However, just prior to the end of the current partial match, there was that substring `"AB"` that could be the beginning of a new match, so the algorithm must take this into consideration. As these characters match the two characters prior to the current position, those characters need not be checked again; the algorithm sets `m = 8` (the start of the initial prefix) and `i = 2` (signaling the first two characters match) and continues matching. Thus the algorithm not only omits previously matched characters of `S` (the `"BCD"`), but also previously matched characters of `W` (the prefix `"AB"`).

```
          1       2
m: 012345678901234567890 12
S: ABC ABCDABABCDABCDABDE
W:       ABCDABD
i:       0123456
```

This search fails immediately, however, as `W` does not contain another `"A"`, so as in the first trial, the algorithm returns to the beginning of `W` and begins searching at the mismatched character position of `S`: `m = 10`, reset `i = 0`.

```
          1       2
m: 012345678901234567890 12
S: ABC ABCDABABCDABCDABDE
W:         ABCDABD
i:         0123456
```

The match at `m=10` fails immediately, so the algorithm next tries `m = 11` and `i = 0`.

```
          1       2
m: 012345678901234567890 12
S: ABC ABCDAB ABCDABCDABDE
W:          ABCDABD
i:          0123456
```

Once again, the algorithm matches "ABCDAB", but the next character, 'C', does not match the final character 'D' of the word W. Reasoning as before, the algorithm sets m = 15, to start at the two-character string "AB" leading up to the current position, set i = 2, and continue matching from the current position.

```
          1       2
m: 012345678901234567890 12
S: ABC ABCDAB ABCDABCDABDE
W:              ABCDABD
i:              0123456
```

This time the match is complete, and the first character of the match is S[15]

Description of pseudocode for the search algorithm[edit]

The above example contains all the elements of the algorithm. For the moment, we assume the existence of a "partial match" table T, described below, which indicates where we need to look for the start of a new match in the event that the current one ends in a mismatch. The entries of T are constructed so that if we have a match starting at S[m] that fails when comparing S[m + i] to W[i], then the next possible match will start at index m + i - T[i] in S (that is, T[i] is the amount of "backtracking" we need to do after a mismatch). This has two implications: first, T[0] = -1, which indicates that if W[0] is a mismatch, we cannot backtrack and must simply check the next character; and second, although the next possible match will *begin* at index m + i - T[i], as in the example above, we need not actually check any of the T[i] characters after that, so that we continue searching from W[T[i]]. The following is a sample pseudocode implementation of the KMP search algorithm.

```
algorithm kmp_search:
input:
    an array of characters, S (the text to be searched)
    an array of characters, W (the word sought)
output:
    an integer (the zero-based position in S at which W is found)

define variables:
    an integer, m ← 0 (the beginning of the current match in S)
    an integer, i ← 0 (the position of the current character in W)
```

```
        an array of integers, T (the table, computed elsewhere)

while m + i < length(S) do
if W[i] = S[m + i] then
if i = length(W) - 1 then
return m
let i ← i + 1
else
if T[i] > -1 then
let m ← m + i - T[i], i ← T[i]
else
let i ← 0, m ← m + 1

    (if we reach here, we have searched all of S unsuccessfully)
return the length of S
```

Efficiency of the search algorithm[edit]

Assuming the prior existence of the table T, the search portion of the Knuth–Morris–Pratt algorithm has complexity $O(n)$, where n is the length of S and the O is big-O notation. Except for the fixed overhead incurred in entering and exiting the function, all the computations are performed in the **while** loop. To bound the number of iterations of this loop; observe that T is constructed so that if a match which had begun at S[m] fails while comparing S[m + i] to W[i], then the next possible match must begin at S[m + (i - T[i])]. In particular, the next possible match must occur at a higher index than m, so that T[i] < i.

This fact implies that the loop can execute at most $2n$ times, since at each iteration it executes one of the two branches in the loop. The first branch invariably increases i and does not change m, so that the index m + i of the currently scrutinized character of S is increased. The second branch adds i - T[i] to m, and as we have seen, this is always a positive number. Thus the location m of the beginning of the current potential match is increased. At the same time, the second branch leaves m + i unchanged, for m gets i - T[i] added to it, and immediately after T[i] gets assigned as the new value of i, hence new_m + new_i = old_m + old_i - T[old_i] + T[old_i] = old_m + old_i. Now, the loop ends if m + i = n; therefore, each branch of the loop can be reached at most n times, since they respectively increase either m + i or m, and m ≤ m + i: if m = n, then certainly m + i ≥ n, so that since it increases by unit increments at most, we must have had m + i = n at some point in the past, and therefore either way we would be done.

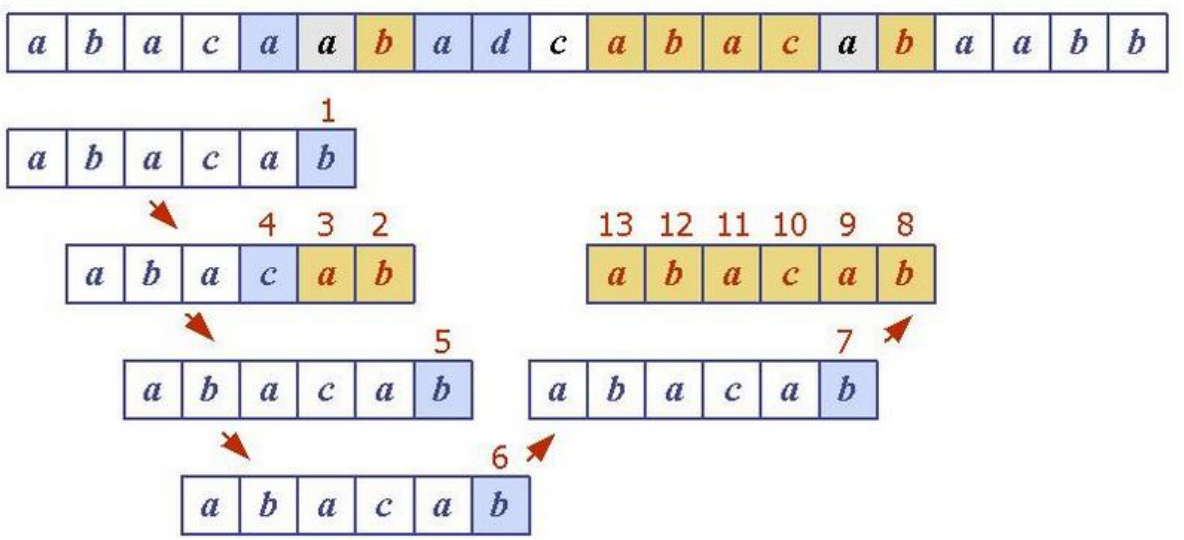
Thus the loop executes at most $2n$ times, showing that the time complexity of the search algorithm is $O(n)$.

Here is another way to think about the runtime: Let us say we begin to match W and S at position i and p. If W exists as a substring of S at p, then W[0..m] = S[p..p+m]. Upon success, that is, the word and the text matched at the positions (W[i] = S[p+i]), we

increase i by 1. Upon failure, that is, the word and the text does not match at the positions ($W[i] \neq S[p+i]$), the text pointer is kept still, while the word pointer is rolled back a certain amount ($i = T[i]$, where T is the jump table), and we attempt to match $W[T[i]]$ with $S[p+i]$. The maximum number of roll-back of i is bounded by i, that is to say, for any failure, we can only roll back as much as we have progressed up to the failure. Then it is clear the runtime is $2n$.

BOYER MOORE ALGORITHM

- string algorithm is significantly enhanced as the comparison Process started at the end of the search pattern processing right to left versus the start of the search pattern.
- The advantage is that large jumps are mismatched character in the input stream the search pattern which occurs frequently.



Examples:

1) Input:

txt[] = "THIS IS A TEST TEXT"

pat[] = "TEST"

Output:

Pattern found at index 10

2) Input:

txt[] = "AABAACAADAABAAABAA"

```
pat[] = "AABA"

Output:

Pattern found at index 0
Pattern found at index 9
Pattern found at index 13
```

Pattern searching is an important problem in computer science. When we do search for a string in notepad/word file or browser or database, pattern searching algorithms are used to show the search results.

We have discussed the following algorithms in the previous posts:

Naive Algorithm
KMP Algorithm
Rabin Karp Algorithm
Finite Automata based Algorithm

In this post, we will discuss Boyer Moore pattern searching algorithm. Like KMP and Finite Automata algorithms, Boyer Moore algorithm also preprocesses the pattern.
Boyer Moore is a combination of following two approaches.
1) Bad Character Heuristic
2) Good Suffix Heuristic
Both of the above heuristics can also be used independently to search a pattern in a text. Let us first understand how two independent approaches work together in the Boyer Moore algorithm. If we take a look at the Naive algorithm, it slides the pattern over the text one by one. KMP algorithm does preprocessing over the pattern so that the pattern can be shifted by more than one. The Boyer Moore algorithm does preprocessing for the same reason. It preporcesses the pattern and creates different arrays for both heuristics. At every step, it slides the pattern by max of the slides suggested by the two heuristics. So it uses best of the two heuristics at every step. Unlike the previous pattern searching algorithms, Boyer Moore algorithm starts matching from the last character of the pattern.
In this post, we will discuss bad character heuristic, and discuss Good Suffix heuristic in the next post.

The idea of bad character heuristic is simple. The character of the text which doesn't match with the current character of pattern is called the Bad Character. Whenever a character doesn't match, we slide the pattern in such a way that aligns the bad character with the last occurrence of it in pattern. We preprocess the pattern and store the last occurrence of every possible character in an array of size equal to alphabet size. If the character is not present at all, then it may result in a shift by m (length of pattern). Therefore, the bad character heuristic takes O(n/m) time in the best case.

```
/* Program for Bad Character Heuristic of Boyer Moore String Matching Algorithm */

# include <limits.h>
```

```
# include <string.h>
# include <stdio.h>

# define NO_OF_CHARS 256

// A utility function to get maximum of two integers
intmax (inta, intb) { return(a > b)? a: b; }

// The preprocessing function for Boyer Moore's bad character heuristic
voidbadCharHeuristic( char*str, intsize, intbadchar[NO_OF_CHARS])
{
   inti;

   // Initialize all occurrences as -1
   for(i = 0; i < NO_OF_CHARS; i++)
       badchar[i] = -1;

   // Fill the actual value of last occurrence of a character
   for(i = 0; i < size; i++)
       badchar[(int) str[i]] = i;
}

/* A pattern searching function that uses Bad Character Heuristic of
  Boyer Moore Algorithm */
voidsearch( char*txt,  char*pat)
{
   intm = strlen(pat);
   intn = strlen(txt);

   intbadchar[NO_OF_CHARS];
```

```
/* Fill the bad character array by calling the preprocessing
   function badCharHeuristic() for given pattern */
badCharHeuristic(pat, m, badchar);

ints = 0;  // s is shift of the pattern with respect to text
while(s <= (n - m))
{
   intj = m-1;

   /* Keep reducing index j of pattern while characters of
      pattern and text are matching at this shift s */
   while(j >= 0 && pat[j] == txt[s+j])
      j--;

   /* If the pattern is present at current shift, then index j
      will become -1 after the above loop */
   if(j < 0)
   {
      printf("\n pattern occurs at shift = %d", s);

      /* Shift the pattern so that the next character in text
         aligns with the last occurrence of it in pattern.
         The condition s+m < n is necessary for the case when
         pattern occurs at the end of text */
      s += (s+m < n)? m-badchar[txt[s+m]] : 1;

   }

   else
      /* Shift the pattern so that the bad character in text
         aligns with the last occurrence of it in pattern. The
```

```
            max function is used to make sure that we get a positive
            shift. We may get a negative shift if the last occurrence
            of bad character in pattern is on the right side of the
            current character. */
          s += max(1, j - badchar[txt[s+j]]);
    }
}

/* Driver program to test above funtion */
intmain()
{
    chartxt[] = "ABAAABCD";
    charpat[] = "ABC";
    search(txt, pat);
    return0;
}
```

Run on IDE

Output:

```
pattern occurs at shift = 4
```

HARDWARE TEXT SEARCH ALGORITHMS

- Specialized hardware machine to perform the searches and pass the results to the main computer which supported the user interface and retrieval of hits.
- Since the searcher is hardware based, scalability is achieved By increasing the number of hardware search devices.
- The only limit on speed is the time it takes to flow the text off of secondary storage By having one search machine per disk, the maximum time it takes to search a database of any size will be the time to search one disk

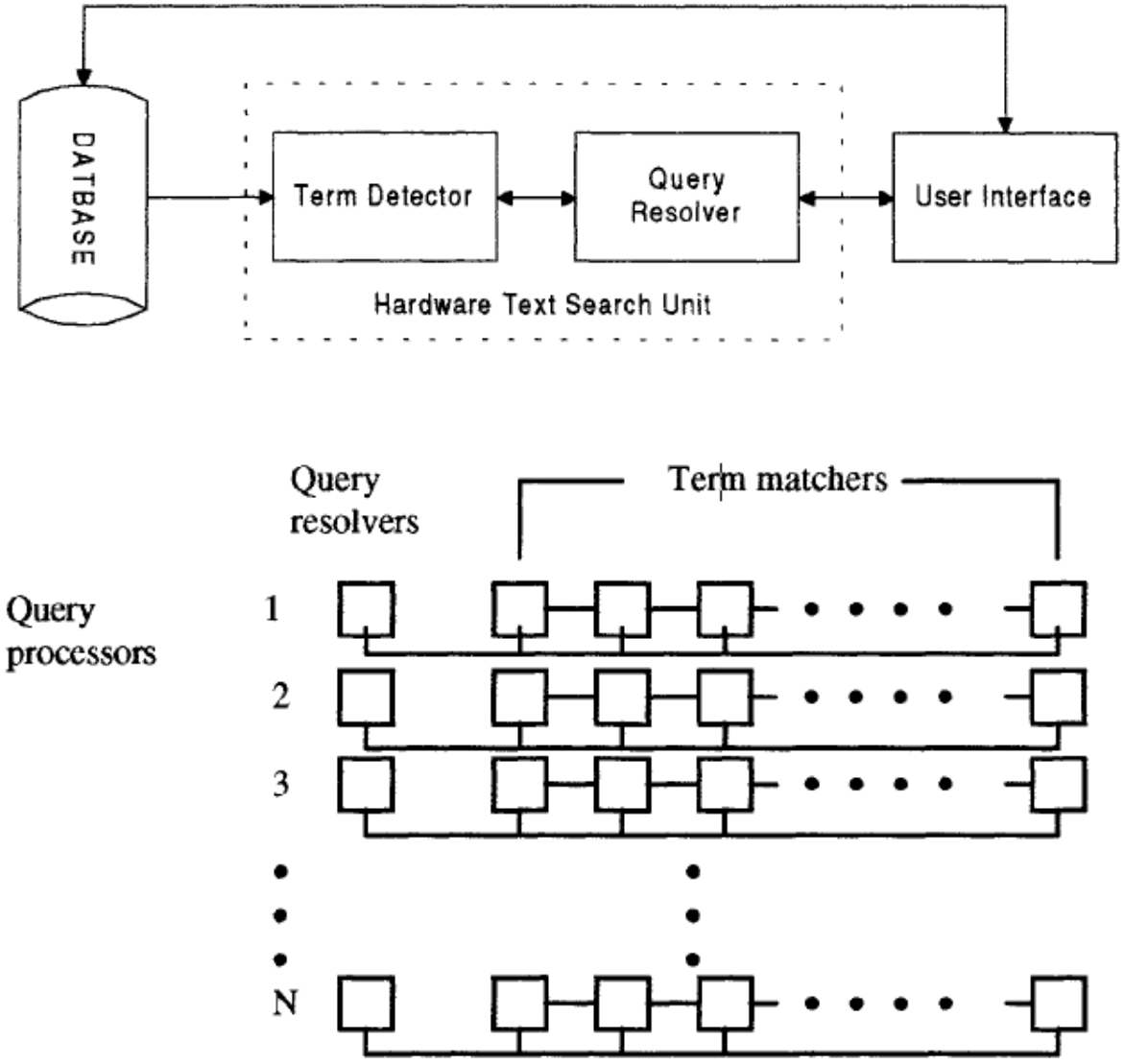


Figure 9.10 GESCAN Text Array Processor

The Fast Data Finder (FDF) is the most recent specialized hardware text search unit still in use in many organizations. It was developed to search text and has been used to search English and foreign languages. The early Fast Data Finders consisted of an array of programmable text processing cells connected in series forming a pipeline hardware search processor (Mettler-93). The cells are implemented using a VSLI chip. In the TREC tests each chip contained 24
processor cells with a typical system containing 3600 cells (the FDF-3 has a rack mount configuration with 10,800 cells). Each cell is a comparator for a single character, limiting the total number of characters in a query to the number of cells. The cells are interconnected with an 8-bit data path and approximately 20-bit control path. The text to be searched passes through each cell in a pipeline fashion until the complete database has been searched. As data are analyzed at each cell, the 20 control lines states are modified depending upon their current state and the results from the comparator. An example of a Fast Data Finder system is shown in Figure 9.11.

A cell is composed of both a register cell (Rs) and a comparator (Cs). The input from the Document database is controlled and buffered by the microprocessor/memory and feed

through the comparators. The search characters are stored in the registers. The connection between the registers reflects the control lines that are also passing state information.

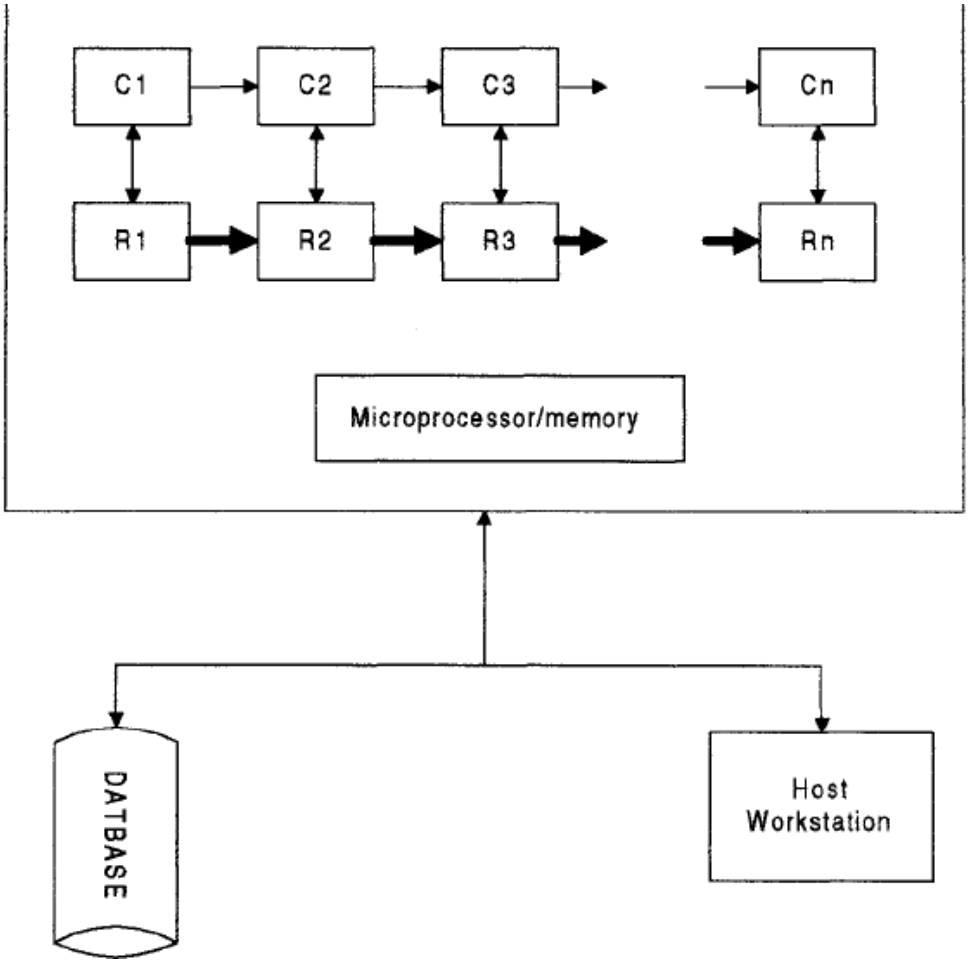


Fig 9.1Fast data finder architecture

INFORMATION SYSTEM EVALUATION

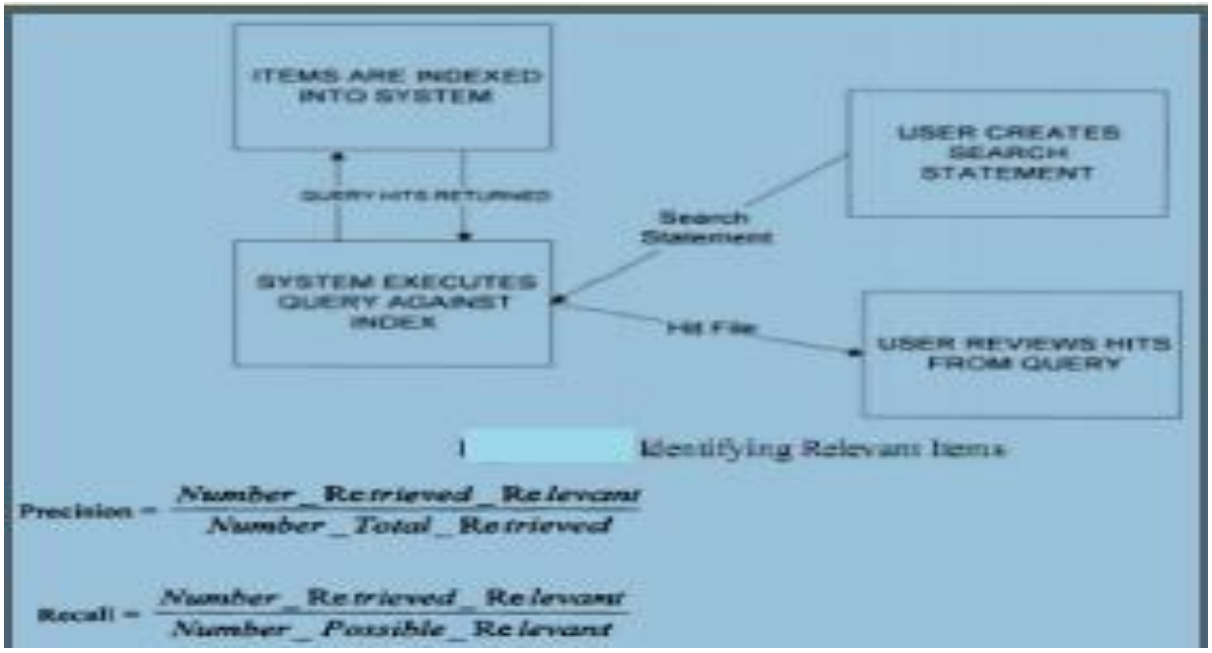
- In recent years the evaluation of IRS and techniques for indexing, sorting, searching and retrieving information have become increasingly important.
- This growth in interest is due to two major reasons:
  1.The growing number of retrieval systems being used
  2.Additional focus on evaluation methods themselves

- There are many reasons to evaluate the effectiveness of an IRS
  1.To aid in the selection of a system to procure
  2.To monitor and evaluate system effectiveness
  3.To evaluate query generation process for improvements
  4.To determine the effects of changes made to an existing information system
- From a human judgment standpoint, relevancy can be considered:
  1.Subjective: depends upon a specific user's judgment

2.Situational : relates to a user's requirements
3.Cognitive : depends on human perception and behaviour
4.Temporal : changes over time
5.Measurable : observable at points in time

- Ingwersen categorizes the information view into four types of "aboutness":
  1.AuthorAboutness:determined by the author's language as matched by the system in natural language retrieval
  2. Indexer Aboutness : determined by the indexer's transformation of the author's natural language into a controlled vocabulary
  3.Request Aboutness : determined by the user's or intermediary's processing of a search statement into a query
  4.UserAboutness : determined by the indexer's attempt to represent the document according to presupposition about what the user will want to know

- Measures used in system evaluation
  To define the measures that can be used in evaluating IRS, it is useful to define the major functions associated with identifying relevant items in an information system.



UNIT-7

- The most important characteristic of a multimedia information system is the variety of data it must be able to support.
- Multimedia systems must have the capability to store, retrieve, transport and present data with very heterogeneous characteristics such as text, images (both still and moving), graphs and sound.Multimedia Information Retrieval Systems specifically for handling multimedia data.
- Multimedia IR systems must be able to deal with different kinds of media and with semi-structured data.Multimedia IR systems must be able to deal with different kinds of media and with semi-structured data
- The main goal of a multimedia IR system is to efficiently perform retrieval, based on user requests, exploiting not only data attributes, as in traditional DBMSs, but also the content of multimedia objects.
- Data retrieval relies on the following basic steps:
    1.Query specification
    2.Query processing and optimization
    3.Query answer
    4.Query iteration

  Multimedia IR systems should therefore combine both the DBMS and the IR technology, to integrate the data modelling capabilities of DBMSs with the advanced and similarity based query capabilities of IR systems

Data Modelling

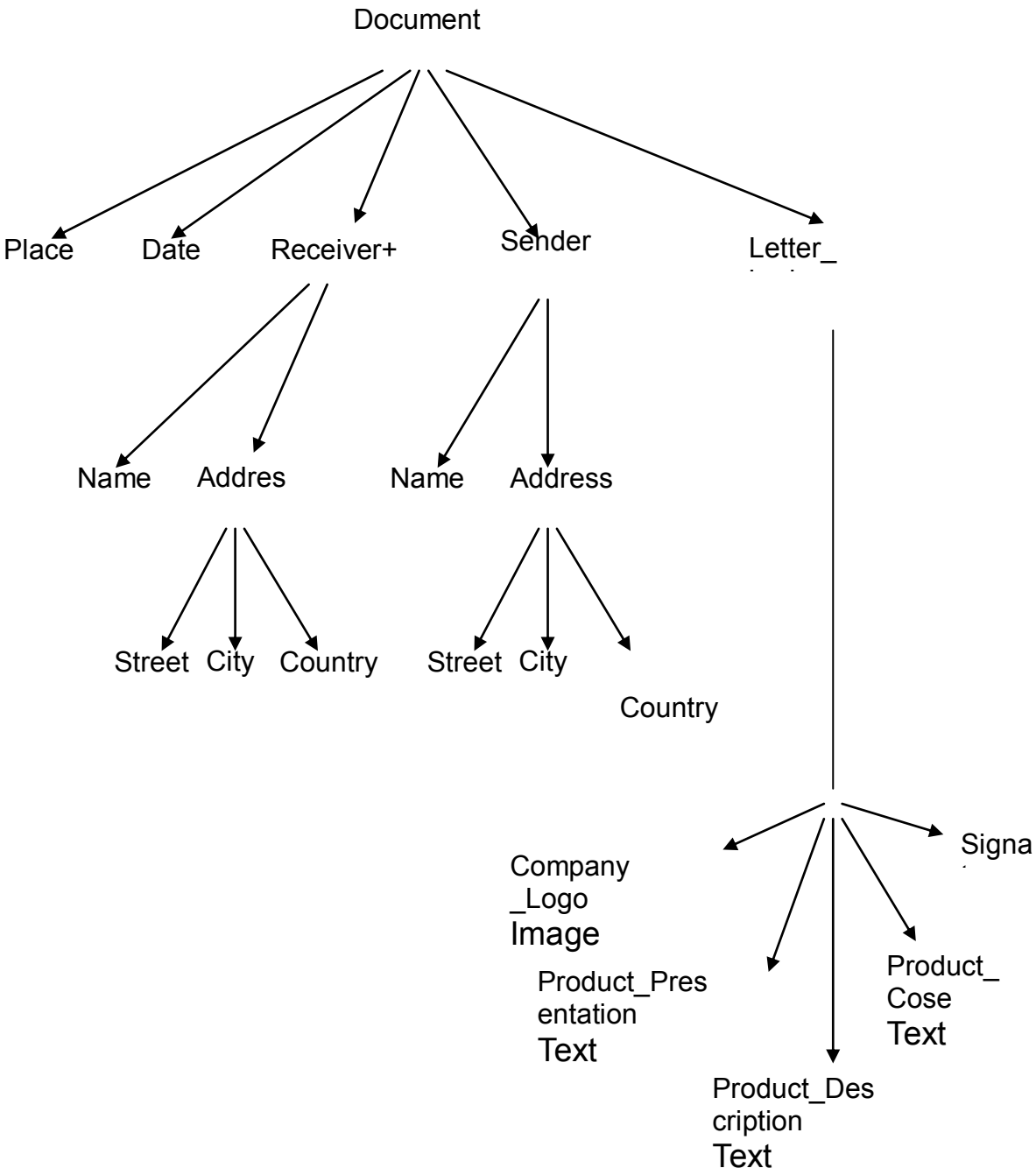
- The complex nature of multimedia data may benefit from the use of DBMS functions for data representation and querying. Multimedia data is inherently different from conventional data. The main difference is that information about the content of multimedia data are usually not encoded into attributes provided by the data schema (structured). Rather, text, image, video and audio data are typically unstructured.
- Addressing data modelling issues in the framework of multimedia IR systems entails two main tasks.
- A data model should be defined by which the user can specify the data to be stored into the system.
- The system should provide a model for the internal representation of multimedia data.

  MULTOS data model : 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.
- MULTOS is based on a client/ server architecture. Three different types of document servers are supported:
    1.Current servers
    2.Dynamic servers
    3.Archive servers
- These three servers differ in storage capacity and document retrieval speed
- Two types of index are constructed:
    1. Object Index
    2.Cluster Index

Do

Conceptual structure of the type **Generic_Letter**

Complete conceptual structure of the type **Business_Product_Letter**



Query languages

Relational/object-oriented database system

- Exact match of the values of attributes

Multimedia IR system

- Similarity-based approach
- Considers the structure and the content of the objects
- Content-based query

- Retrieve multimedia objects depending on their globe content

In designing a multimedia query language, three main aspects require attention

- How the user enters their request to the system
- Which conditions on multimedia objects can be specified in the user request
- How uncertainty, proximity, and weights impact the design of the query language

Request specification

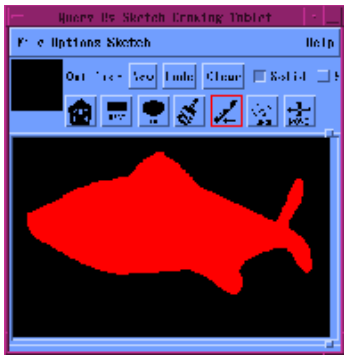
Interfaces

- Browsing and navigation
- Specifying the conditions the objects of interest must satisfy, by means of queries

Queries can be specified in two different ways

- Using a specific query language
- Query by example
- Using actual data (object example)

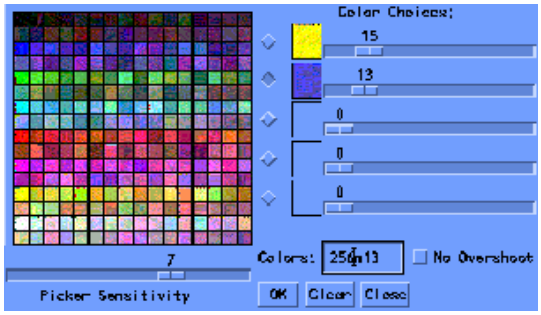
Search Using Sketch in QBIC



- **Sketch entry**

Results of search

Search by Color in QBIC



- **Color selection**
- **15% yellow and 13% blue**

- **Results of search**

Conditions on multimedia data

1. Attribute predicates

- Concern the attributes for which an exact value is supplied for each object
- Exact-match retrieval

2. Structural predicates

- Concern the structure of multimedia objects
- Can be answered by metadata and information about the database schema
- "Find all multimedia objects containing at least one image and a video clip"

3. Semantic predicates

- Concern the semantic content of the required data, depending on the features that have been extracted and stored for each multimedia object
- "Find all the red houses"
- Exact match cannot be applied

Indexing and Searching

→Given two objects O1and O2the distance (=dissimilarity) of the two objects is denoted by D(O1, O2).

→Similarity queries can be classified into two categories: Whole match : given a collection of N objects O1, O2, . .. , ON and a query object Q, we want to find those data objects that are within distance €from Q.

Sub-pattern match: Given N data objects O1, O2, . . . , ON, a

query (sub-) object Q and a tolerance €, we want to identify the parts of the data objects that match the query

Ideal method which required by all categories of queries should ful fill the following requirements:

- o It should be fast.
- o It should be "correct"
- o It must have a small space overhead It should be dynamic

Algorithm-1 Search:

- Map the query object Q into a point F(Q) in feature space
- Using a spatial access method, retrieve all points within the desired tolerance €from F(Q).
- Retrieve the corresponding objects, compute their actual distance from Q and discard the false alarms.
- Lemma(Lower Bounding): to guarantee no false dismissals for whole match queries, the feature extraction function F() should satisfy the following formula:

$$\mathcal{D}_{feature}(\mathcal{F}(O_1), \mathcal{F}(O_2)) \leq \mathcal{D}(O_1, O_2)$$

Algorithm-2 (GEMINI) Generic Multimedia object Indexing approach:

- Determine the distance function D() between two objects o Find one or more numerical feature-extraction functions, to provide a 'quick-and-dirty' test
- Prove that the distance in feature space lower-bounds the actual distance D(), to guarantee correctness. Use a SAM, to store and retrieve the f-D feature vectors.
- Feature extracting question: If we are allowed to
- use only one numerical feature to describe each data object, what should this feature be?

UNIT-8

- **Introduction**
- **Document Models, Representations, and Access**
- **Prototypes, Projects, and Interfaces**
- **Standards**
- **Trends and Research Issues**
- **Environment in the Library**
- **Online Public Access Catalogues (OPACs)**
- **Document Databases**
- **IR in Organizations**
- **Conclusion**

Digital library (DL) can be defined by:

The combination of a collection of digital objects (repository);descriptions of those objects (metadata); a set of users (patrons or target audience or users);systems that offer a variety of services such as capture, indexing, cataloging, search, browsing, retrieval, delivery, archiving, and preservation.

Information retrieval (IR) is essential for the sucess of DLs, so they can achieve high levels of effectiveness while at the same time affording ease of use to a diverse community.
A significant portion of the research and development efforts related to DLs has been in the IR area.This presentation reviews some of these efforts, organizes them into a simple framework, and highlights needs for the future.

Document Models, Representations, and Access
Without documents there would be no IR or DLs. Hence, it is apropriate to consider definitions of 'document', and to develop suitable formalizations, as well as to articulate research concerns.
Because DLs can be constructed for a particular institution or nation, it is likely that the expansion of DLs will increase acess to documents in a variety of languages.
There are issues of character encoding:Unicode provides a single 16-bit encoding scheme suitable for all natural languages;Downloading fonts from a special server or gateway is a less costly implementation.The next crucial problem is searching multilingual collections:

The simplest approach is to locate words or phrases in dictionaries and to use the translated terms to search in collections in other languages.It is likely that research in this area will continue to be of great importance to both the IR and DL communities.

Multimedia documents' streams usually must be synchronized in some way, and so it is promising that a new standard for handling this over the Web has been adopted.

IR has been applied to various types of multimedia content:
Columbia University: a large image collection from the Web can be searched on content using visual queries;

- IBM: *Query By Image Content (*QBIC) system for images and video was developed.
- Better handling of multimedia is at the heart of future research on many types of documents in DLs.Very powerful representation, description, query and retrieval systems may be required to properly handle the complexity of multimedia collections.

Document Models, Representations, and Access - Structured Documents –

- Structured documents are streams with one or more structures imposed.
- Metadata can be represented as an SGML document and SGML content can be included in the base document and /or be kept separately.
- Structure is often important:
- in documents, when one wants to add value or make texts 'smart' (SGML);
- in retrieval (OpenText);
- at the level above documents, which makes searching necessary and possible.

Document Models, Representations, and Access - Distributed Collections –

- Most DLs are spread across computers, that is spanning physical and/or logical spaces.

- Dealing with collections of information that are distributed in nature is one of the commom requirements of DL technology.

Document Models, Representations, and Access - Distributed Collections –

- There are two approaches to tackle this problem:
- The first one is to develop a description language for each DL and to build federated search systems that can interpret it;
- The second one is to make each DL support a powerful protocol aimed at effective retrieval.
- The first course is supported by BioKleili system and the second one by Computer Interchange of Museum Information (CIMI).

Document Models, Representations, and Access - Distributed Collections –

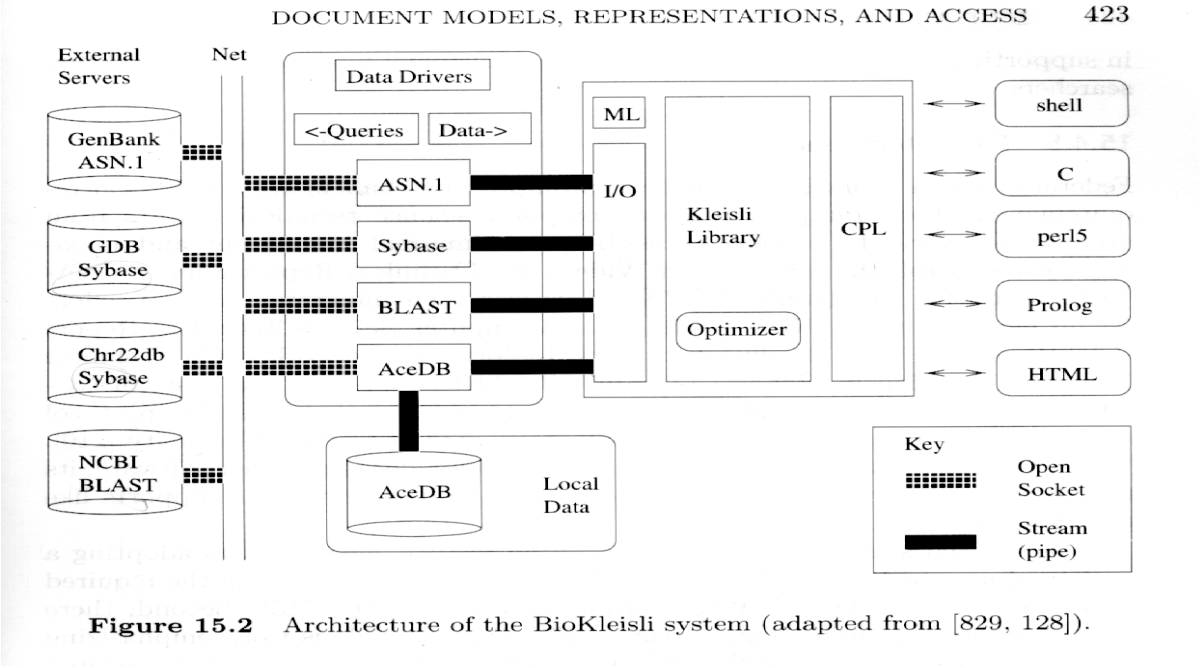


Figure 15.2 Architecture of the BioKleisli system (adapted from [829, 128]).

Document Models, Representations, and Access- Federated Search –

- Federated search is the support for finding items that are scattered among a distributed collection of information sources or services, typically involving sending queries to a number of servers and then merging the results to present in an integrated, consistent, coordinated format.

Document Models, Representations, and Access- Acess –

- DLs must manage intellectual properties. These services must support agreed-upon principles, copyright practices, contracts and other agreements and laws.
- A key to the implementation of policies for access management is having trusted systems.
- Stronger mechanisms are crucial in DLs to:
- protect intellectual property rights;
- control the types of access afforded to different user groups.

Prototypes, Projects and Interfaces- Internacional Range of Efforts -

- Since each nation wishes to share highlights of its history, culture, and accomplishments with the rest of the world, developing a DL can be very helpful.
- Examples :
- European ERCIM program and UK initiatives;
- New Zealand, Australia and Singapore;

- IBM Digital Library;
- Networked Digital Library of Theses and Dissertations (NDLTD).

Prototypes, Projects, and Interfaces- Usability –

- Key to the success of the DLs is having usable systems. This is a serious challenge!
- Simple library catalog systems were observe in 1986 to be difficult to use.
- A 1997 study at Virginia Tech of four digital library : ACM; NCSTRL; NDLTD; IEEE-CS;
- The participants were 48 Virginia Tech students. 38 graduate students, 8 undergraduate, 2 other graduate studies;
- The study concluded that many systems have serious usability problems :
- Pre-test found that very few users had worked with a DL;
- Post-test showed that user expectations and priorities changed over the short test period;
- Features derived from user feedback and avaluators observations : clear overview; search criteria for simple search; search criteria for advanced search; fast searching and retrieval; eaxamples searches; ability to download a fraction of the article; save queries for future refinement.

Standards

- Since there are many DL projects worldwide, involving diverse research, development, and commercial approaches, it is imperative that standards are employed so as to make interoperability and data exchange possible.
- At the heart of supporting federated DLs is agreement on protocols for computer-computer communication.

Standards- Protocols and Federation –

- The standard Z39.50 was designed to search remote bibliographic collections;
- One example of widespread utilization was the WAIS system (based on Z39.50), very popular before WWW emerged;
- The application of Z39.50 to DLs was demonstrated in the CIMI project;
- Dienst is another standard which has been considered in regard to NDLTD.

Trends and Issues

There are the problems of handling worldwide DLs, in the context of varying collection principles, enormous difference in response time between local and remote servers, and the needs of users for different views.Of central concern is covering the range from personal to global DLs, the so-called ‘scaling’ problem.At the same time, the problem of interoperability must be faced.

Environment in the Library

Web provides ubiquitous informational repository but chaotic and unstructured many of the sources may be questionable regarding accuracy, reliability, completeness results for query "information retrieval" on Google wikipedia article on IR 1979 book on IR by van RIjsbergen

Many searches do not admit a Web solution diagnosis or treatment of a disease surgery procedure case law and precedents patent rights Libraries provide IR services for collections, books, journals, digital materials all systematically acquired and organized There will always be closed repositories in the corporate world with vast repositories of data

OPACs Catalogue access point for materials held by library Integrated Library System (ILS) system that manages all library catalogues and collections OPAC early on, libraries used card catalogues later, followed on microfilms and microfiche forms in the 1970's, online catalogues were implemented initially, they had very limited functionality in the 1980's, true online public access catalogues (OPACs) today, OPACs are the main component of an ILS Library Systems, Modern Information Retrieval, Addison Wesley, 2010 – p. 7 OPACs OPACs used standardized record formats (MARC) minimal subject information (title, a few headings) Three generations first generation: known-item finding tools through search by author, title, control number second generation: increased search technology with access by subject headings, keywords, boolean queries problems included failed searches, navigational confusion enhancements represented large investments for a library third generation: focus on open systems architectures, improved GUI, support for Z39.50 and Dublin Core, hypertext links, java programming, ranked results sets problems include slow pace of development failure to match trends in the Web Library Systems, Modern Information Retrieval, Addison Wesley, 2010 – p. 8 OPACs and Bibliographic Records Libraries use standardized systems for cataloguing and classifying materials Anglo-American

Cataloguing Rules: to describe materials Library of Congress or Dewey Decimal Classification: to assign subject codes Library of Congress Subject Headings: to assign subject descriptors For sharing information, they rely on centralized bibliographic utilities to lower the cost per unit to catalogue materials broaden access through shared databases facilitate the sharing of materials Library Systems, Modern Information Retrieval, Addison Wesley, 2010 – p. 9 Centralized Bibliographic Utilities Broaden access through shared databases Facilitate the sharing of materials Online Computer Library Center (OCLC) used by 69,000 libraries in 112 countries and territories union catalogue of collections over 10,000 libraries WorldCat 125 million bibliographic records 1.3 billion library holdings opened to the public in 2006 as worldcat.org Library Systems, Modern Information Retrieval, Addison Wesley, 2010 – p. 10 MARC

MARC is the Machine Readable Cataloguing Record underlines cooperation among libraries provides support to distinct online catalogues data format that implements national and international standards Information Interchange Format (ANSI Z39.2) .