1 Introduction to Information Retrieval

UNIT - I

Syllabus

Basic Concepts of IR, Data Retrieval & Information Retrieval, Text mining and IR relation, IR system block diagram, Automatic Text Analysis: Luhn's ideas, Conflation Algorithm, Indexing and Index Term Weighting, Probabilistic Indexing, Automatic Classification. Measures of Association, Different Matching Coefficients, Cluster Hypothesis, Clustering Techniques: Rocchio's Algorithm, Single pass algorithm, Single Link algorithm

1.1 Basic Concepts of IR

University Question

Q. Explain basic concept of Information Retrieval. **SPPU : May 19, 2 Marks**

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

1.2 Data Retrieval and Information Retrieval

University Question

Q. Differentiate between data and information retrieval. **SPPU : Dec. 14, May 19, 6/8 Marks**

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

Table 1.2.1

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

1. Matching

- In data retrieval, we normally search for exact match for e.g. whether a file contains a particular word or n
- In information retrieval, we normally find documents which partially match the request and then select out of them.

2. Inference

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

3. Model

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

4. Classification

- In Data Retrieval we are interested in monothetic classification i.e. one with classes defined by obje possessing attributes, both necessary and sufficient, to belong to a class.
- In Information Retrieval monothetic classification is not required.
- In Information Retrieval polythetic classification gets used.
- In such a classification each individual in a class will possess only a proportion of all attributes possessed by members of that class. Hence no attribute is necessary nor sufficient for membership of a class.

Query language

- The query language which is used in Data Retrieval is mostly artificial; with restricted syntax and vocabula e.g. we can write a query in SQL in its fixed format with fixed words.

- In Information Retrieval, query will have no restriction related to syntax or vocabulary.
- User can provide a query in natural language format. The Information Retrieval system should be able to handle such queries.

6. Query specification

- As Data Retrieval finds exact match and the query follows a restricted format, the query must be complete. User should provide the exact query for his interest.
- In Information Retrieval, user can use natural language to specify the query and hence it is possible that the query may be incomplete e.g. user will not follow the standard grammar of the language. The Information Retrieval system can handle such queries.

7. Items wanted

- In Data Retrieval, user specifies the exact query and hence the list of items will contain those items which exactly match the query.
- In Information Retrieval, the query gets specified in natural language as well as the models which used for finding the items are probabilistic, the system will find items which are relevant to the query.
- User then will decide for best one from the listed output.

8. Error response

- In Data Retrieval, the query must be complete with proper syntax. Hence if there will be any error while specifying the query, meaning will be totally different and we can get wrong items.
- In Information Retrieval, query gets specified in natural language, hence some sort of relaxation can be handled by the system.

1.2.1 Text Mining and IR Relation

- Information Retrieval is related to text, images, audio, video or object oriented type of information. IR deals with the efficient way of storage of the information and various methods of searching the information based on user's interest. Handling textual information is subdomain of IR.
- IR is more to do with search engines where we have large amount of information and based on user's requirement, specific information is extracted from the collection. IR is more hybrid topic which converts, machine learning techniques, Natural Language processing techniques. Now-a-days, more focus of IR is on search engines.

1.3 Information Retrieval System: Block Diagram

University Questions

Q. Draw and explain IR system block diagram. **SPPU : Dec. 16, March 19, 5 Marks**

Q. Draw IR system block diagram. **SPPU : May 19, 3 Marks**

Q. What is a document representative? Explain with a suitable example. **SPPU : March 19, 5 Marks**

- An information retrieval system deals with representation, storage, organization and access of information. Fig. 1.3.1 shows the block diagram of a typical Information Retrieval system.

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

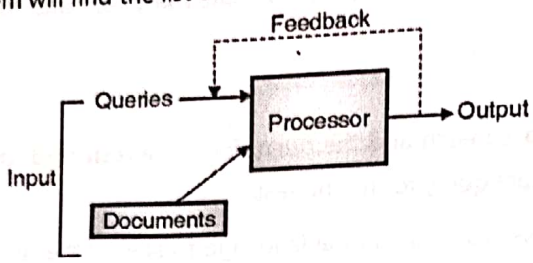


Fig. 1.3.1 : Information Retrieval system block diagram

- The main problem here is to obtain representation of each document and query suitable for computer to use. At the first step the documents are converted into its representation. The documents in its natural language format are one of the representations. But if we store these documents in natural language format, space requirement gets increased as well as time to retrieve the items which are relevant to query is also large.
- Hence most computer based retrieval systems store only representation of the document. A **document representative** could be a list of extracted words considered to be significant. These words are called as **keywords**. Text of a document is lost once it has been converted into document representation.

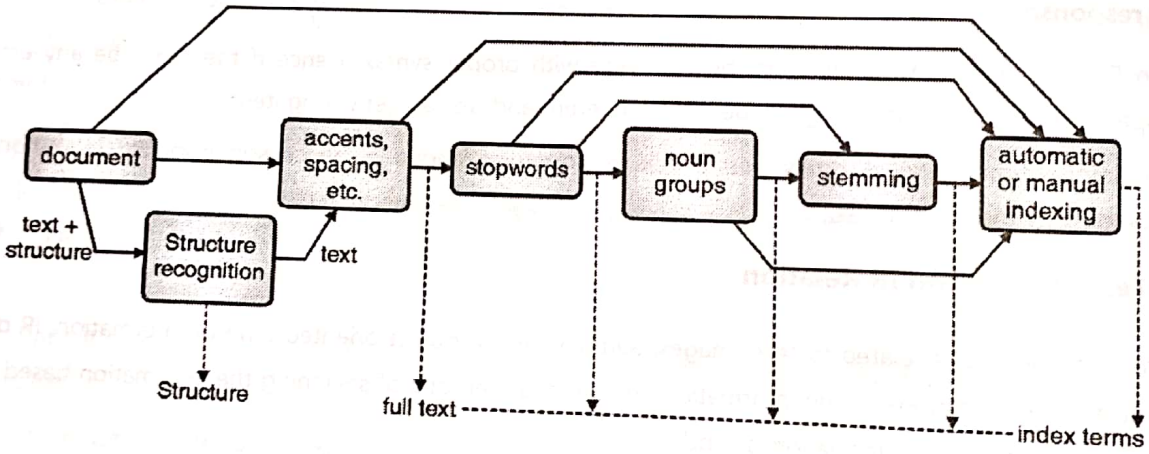


Fig. 1.3.2 : Logical view of the document

- *Fig. 1.3.2 shows the logical view of the document. A full text can be document representative or list of keyword (index terms) can be a document representative or any intermediate status of the document can be documen representative.*
- *As the document gets converted in its internal representation, the queries given by the user are also converted i the same fashion.*
- *The Information Retrieval system will internally find the relevant documents compared with the query and outpu i.e. the list of relevant documents will be provided to the user.*
- *User can stop here or if user wants to refine the query, he can provide the feedback based on which the query get modified.*
- *The same process will be done with modified query and finally the output i.e. list of relevant documents are provided to the user.*

1.4 Automatic Text Analysis

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

1.5 Luhn's Ideas

University Questions

Q. Explain Luhn's idea for understanding the context of the document. **SPPU : Dec. 14, 8 Marks**

Q. Explain Luhn's idea in details. **SPPU : May 16, 5 Marks**

- Document can be represented as a list of words. But here the question aries which words can be picked from the documents to create the document representative. Luhn has given the basic idea for this selection.
- Luhn states that the frequency of word occurrence in an article furnishes a useful measurement of word significance. The relative position within a sentence of words having given values of significance furnishes a useful measurement for determining the significance of sentences.
- The significance factor of a sentence will therefore be based on a combination of these two measurements.
- In short, he states that frequency of the words can be used to extract words and sentences to represent a document. Luhn used the Zipf's law for stating the idea. Zipf's law states that the product of the frequency of use of words and the rank order is approximately constant.
- Luhnhas specified following idea. Let,

  F : Frequency of occurrence of various word types in a given position of text.

  R : The rank order of these words i.e. order of their frequency of occurrence.
- Then plot a graph relating f and r which is like a hyperbolic curve.
- Here we are interested to find the significant words from the document.
- Fig. 1.5.1 shows a plot of the hyperbolic cure relating, f the frequency of occurrence and r, the rank order. Luhn has stated two cut-offs upper cut off and lower cut off.
- Upper cut-off is used for excluding common words. The words whose frequency is greater than the upper cut off are the common words. These words do not contain the semantic of the documents. Hence these words are not considered in the list.

- The words having less frequency as compared to the lower cut-off are the rare words. Hence get discarded.
- Thus, the words whose frequency values are in the range of upper and lower cut-off are considered as significant words. These words become the part of document representative.

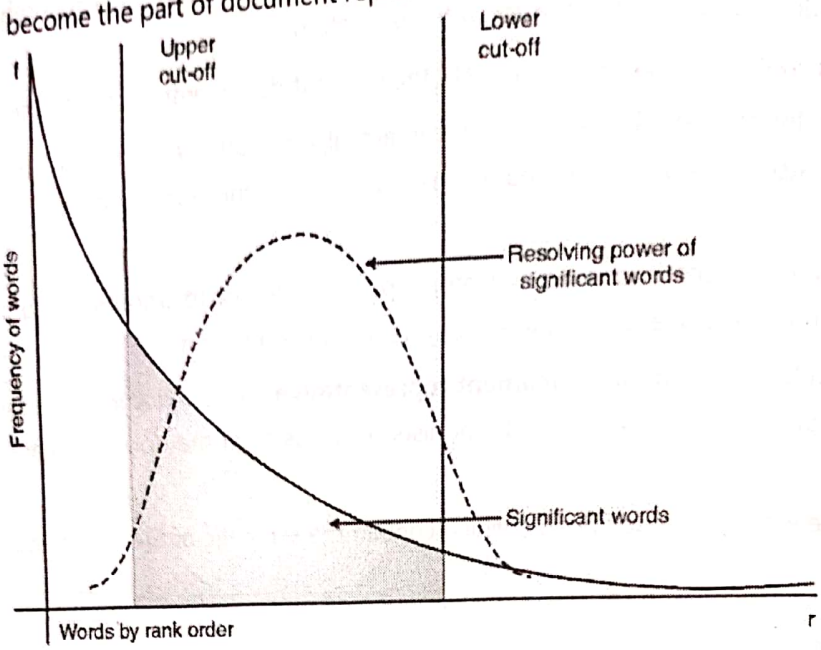


Fig. 1.5.1 : Luhn's Idea

- There is no thumb rule for deciding upper and lower cut-off. They have to be established by trail and error.

1.6 Conflation Algorithm

University Questions

Q. How to generate the document representatives using conflation algorithm? **SPPU : Dec. 14, 10 Marks**

Q. Explain steps in conflation algorithm using a suitable example. **SPPU : Dec. 16, March 19, 5 Marks**

Q. List and explain steps of conflation algorithm. **SPPU : May 19, 4 Marks**

Q. You are developing a text processing system for use in an automatic retrieval system. Explain the following parts :

i) Removal of high frequency words

ii) Suffix stripping

iii) Detecting equivalent stems. **SPPU : May 17, 5 Marks**

- Conflation algorithm is the method to find the document representative. It is based on the Luhn's idea.
- The algorithm consists of three parts as shown in Fig. 1.6.1.

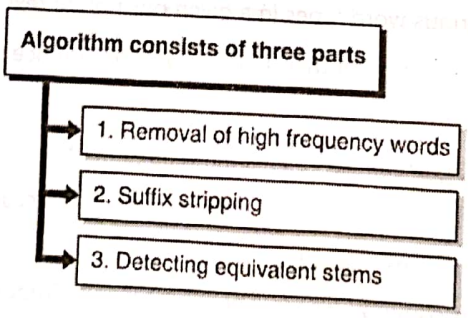


Fig. 1.6.1 : Parts of Conflation algorithm

1. Removal of high frequency words

- The removal of high frequency words i.e. 'stop' words or 'fluff' words is one way of implementing Luhn's upper cut-off. This is done by comparing each word from the document to the list of high frequency words. High frequency words are those words which comes more number of times in the text.
- These words does not contain meaning or semantic of the text. For e.g. is, am, I, are, the etc. are some of words.
- The advantage of this process is not only the non-significant words are removed but also the size of the document can be reduced to 30 to 50%.

2. Suffix stripping

- The second step is the suffix stripping. In this step, each word is handled from the output of first step. If any word is having the suffix, then the suffix gets removed and the word is converted in its original form.
- For e.g. If the word is "Killed" it will be converted into 'Kill' other example are :

	Original word		Word in original form
1.	Processes	→	Process
2.	Repeated	→	Repeat
3.	Kidding	→	Kid
4.	National	→	Nation

- Unfortunately, context free removal leads to a significant error rate. For e.g. we may want UAL removed from FACTUAL but not from EQUAL.
- To avoid erroneously removing suffixes some rules can be followed.For e.g.
  1. The length of remaining stem exceeds a given number, the default is usually 2.
  2. The stem-ending satisfies a certain condition, e.g. does not end with Q.
- For removing the suffixes the rules of grammar of the language can be used. For English language porter's algorithm is one of the algorithm which helps in removal of suffixes.
- The process is called as stemming. An advantage of stemming is it reduces the size of the text. However, too much stemming is not practical and annoys users.

3. Detecting equivalent stems

- After suffix stripping, we will have the list of words. Only one occurrence of the word is kept in the list for e.g. if two words "processing" and "processed" get converted into 'process' only one occurrence of process will be part of the list. Each word is called as 'stem'.
- If two words have the same underlying stem then they refer to the same concept and should be indexed as such. This is obviously an over simplification since words with same stem, such as NEUTRON AND NEUTRALISE, sometimes need to be distinguished.
- The final output from a conflation algorithm is a set of classes, one for each stem detected. As class name is assigned to a document if and only if one of its members occurs as a significant word in the text of the document.

- A document representative then becomes a list of class names. These are referred as the document index **terms or keywords**.
- Queries are also treated in the same way. Thus each query is converted in query representative.

1.7 Indexing and Index Term Weighting

1.7.1 Indexing

- During "Indexing" documents are prepared for use by Information Retrieval system.
- This means preparing the raw document collection into an easily accessible representation of documents. This transformation from a document text into a **representation** of text is known as **indexing** the documents.
- Transforming a document into an indexed form involves the use of :
  - A library or set of regular expressions
  - Parsers
  - A library of stop words (a stop list)
  - Other miscellaneous filters
- Conflation algorithm is used for converting document into its representation i.e. indexing. Each element i.e. word in index language is referred as index term.
- Index language may be pre-coordinate or post-coordinate. In pre-coordinate, the terms are coordinated at the time of indexing. A logical combination of any index terms may be used as a label to identify a class of documents.
- In post-coordinate, the terms are coordinated at the time of searching. In post-coordinate indexing the same class would be identified at search time by combining the classes of documents labelled with the individual index terms.
- The vocabulary of an index language may be controlled or uncontrolled. The former refers to a list of approved index terms that an indexer may use.
- The controls on the language may also include hierarchic relationships between the index terms. Or, one may insist that certain terms can only be used as adjective. There is really no limit to the kind of syntactic controls one may put on a language.
- The index language which comes out of the conflation algorithm is uncontrolled, post-coordinate and derived.
- The vocabulary of index terms at any stage in the evolution of document collection is just the set of all conflation class names.

1.7.2 Index Term Weighting

University Question

Q. Describe index term weighing. **SPPU : May 15, 8 Marks**

- *Weighting* is the final stage in most Information Retrieval indexing applications. For each index term a weight value get assigned which indicates the significance of that index term w.r.to the document.
- The two important factors governing the effectiveness of an index language are **exhaustivity** of indexing and ***specificity*** of index language.

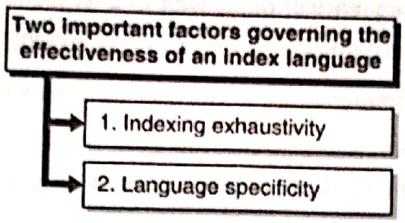


Fig. 1.7.1 : Factors governing effectiveness of index language

1. Indexing exhaustivity

- It is defined as the number of different topics indexed. A high level of exhaustivity of indexing leads to high recall and low precision.
- A low level of exhaustivity leads to low recall and high precision. In short, exhaustivity is the no. of index terms assigned to the document.

2. Language specificity

- It is the ability of the index language to describe topics precisely. It is the level of precision with which a document is actually indexed.
- High specificity leads to high precision and low recall. Specificity means the no. documents to which a given term is assigned in a given collection.
- Refer to the Luhn's idea. He has mentioned the discrimination power of index terms as a function of the rank order of their frequency of occurrence.
- The highest discrimination power being associated with the middle frequencies.
- Considering this idea, each index term is assigned with a weight value which indicates the significance of the index term in the document.
- A frequency count is how many times the index term comes in the document, can be considered as the weight value.
- Different methods are present to find the weight value of index terms. First is to assign the frequency count as the weight value.
- Another way is based on the index term distribution in the entire collection.

  Let, N = Total no. of documents

  n = The document in which an index term accurse;

  Then the weight can be calculated as $-\log\left(\frac{N}{n}\right)+1$

- If we compare two methods. The document frequency weighting places emphasis on content description whereas the second method i.e. weighting by specificity attempts to stress the ability of terms to discriminate one document from another.
- Salton and Yong combined both the methods of weighting considering inter document frequencies and intra document frequencies.

- By considering both, the total frequency of occurrence of a term and its distribution over the documents, that is how many times it occurs in each document, they concluded many things like.
  1. A term with high total frequency of occurrence is not very useful in retrieval irrespective of its distribution.
  2. Middle terms are most useful particularly, if the distribution is skewed.
  3. Rare terms with a skewed distribution are likely to be useful but less so than the middle frequency ones.
  4. Very rare terms are also quite useful but come bottom of the list except for the ones with a high total frequency.
- A **"good"** index term is one which, when assigned as an index term to a collection of documents, renders the documents as dissimilar as possible, where as a **"bad"** index term is one which renders the documents more similar.
- This is quantified through a **term discrimination value** which for a particular term measures the increase or decrease in the average dissimilarity between documents on the removal of that term. Therefore, a good term is one which on removal from the collection of documents, leads to a decrease in average.
- Dissimilarity, whereas a bad term is one which leads on removal to an increase.
- The idea is that a greater separation between documents will enhance retrieval effectiveness but then less separation will depress retrieval effectiveness.

1.8 Probabilistic Indexing

- Probabilistic indexing is based on the probability model for Information Retrieval.
- This model, considers the difference in the distributional behaviour of words as a guide to whether a word should be assigned as an index term.
- The statistical behaviours of 'speciality' words are different from that of 'function' words.
- Function words are closely modelled. By Poisson distribution over all documents where as speciality words did not follow a Poisson distribution.

Let, ω = function word over a set of texts

n = number of occurrences of word ω.

- Then, f (n) the probability that a text will have **n** occurrences of a function word ω is given by,

$$f(n) = \frac{e^{-x} x^n}{n!}$$

- x will vary from word to word and for a given word should be proportional to the length of the text.
- We can interpret x as the mean number of occurrences of the ω in the set of texts.
- 'Speciality words' are content bearing. Whereas function words are not. Word randomly distributed according to a Poisson distribution is not informative about the document in which it occurs.
- A word which does not follow a Poisson distribution is assumed to indicate that it conveys information as to what document is about.

- For e.g. WAR is speciality word. It can come in relative documents. Whereas 'FOR' is a function word, which can be randomly distributed.
- This model also assumes that a document can be about a word of **some degree**. A document collection can be broken up into subsets. Each subset being made up of documents that are about a given word to the **same degree**.
- Content-bearing word is a word that distinguishes more than one class of documents w.r.to the extent to which the topic referred to by the word is treated in the documents in each class.
- These are candidates for index terms. These content bearing words can be mechanically detected by measuring the extent to which their distributions deviate from that expected under a Poisson process.
- In this model the status of one of these content words within a subset of documents of the same 'aboutness' is one of non-content-bearing, this is, within the given subset it does not discriminate between further subsets.
- The assumptions based on which a word can be considered as index term for the document are :
  - The probability that a document will be found relevant to request for information on a subject is a function of the relative extent to which the topic is treated in the document.
  - The no. of tokens in a document is a function of the extent to which the subject referred to by the word is treated in the document.
- The indexing rule based on these assumptions indexes a document with word ω if probability exceeds some cost function.
- If there are only two subsets differing in the extent to which they are about a word ω then the distribution of ω can be described by a mixture of two Poisson distributions.

$$\text{Thus, } f(n) = \frac{P_1 e^{-x_1} x_1^n}{n!} + \frac{(1-P)\ e^{-x_2} x_2^n}{n!}$$

- Here, P_1 is the probability of a random document belonging to one of the subsets and x_1 and x_2 are the mean occurrences in the two classes.
- This model is called as **2-Poisson model**. It describes the statistical behaviour of a content-bearing word over two classes which are 'about' that word to different extents, these classes are not necessarily the relevant and non-relevant documents although by assumption (1) we can calculate the probability of reference for any document from one of these classes.
- It is the ratio $\dfrac{P_1 e^{-x} x_1^k}{P_1 e^{-x_1} x_1^k + (1-P_1)\ e^{-x_2} x_2^k}$
- That is used to make the decision whether to assign an index term ω that occurs k times in a document.
- This ratio is in fact the probability that the particular document belongs to the class which treats ω to an average extent of $\mathbf{x_1}$ given that it contains exactly **k** occurrences of ω.

1.9 Automatic Classification

- Classification is the process to categories the given document in different groups. Here, we make the subsets of given objects.
- There are two main areas of application of classification methods in Information Retrieval as shown in Fig. 1.9.1.

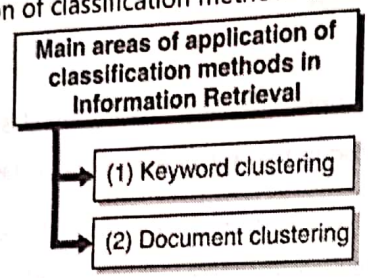


Fig. 1.9.1 : Areas of application ofclassification method in IR

1. Keyword clustering

- Many automatic retrieval systems rely on thesauri to modify queries and document representatives to improve the chance of retrieving relevant documents. In practice many of thesauri are constructed manually.
- They have mainly been constructed in two ways :
  1. Words which are deemed to be about the same topic are linked.
  2. Words which are deemed to be about related things are linked.

2. Document clustering

- Document clustering is to group the documents in the collection. The purpose will be to group the documents in such a way that retrieval will be faster or alternatively it may be to construct a thesaurus automatically.
- Whatever the purpose the 'goodness' of the classification can finally only be measured by its performance during retrieval.
- Considering the collection, the given documents will be divided in different groups (or subsets). Each group will be considered as a single cluster.
- A document can become part of a particular cluster if it is logically related to other members of the cluster. Thus a single cluster will contain all those documents which are semantically related to each other.
- Purpose of clustering is to increase the speed of searching. In practice it is not possible to match each analysed document with each analysed search request because the time consumed by such operation would be excessive.
- Using clustering process will have following steps :
  - For the given collection, using some algorithms, documents will be converted into different groups. Each group or cluster will contain the semantically related documents.
  - For each cluster, one cluster representative will be decided. The cluster representative may be such a document which is semantically near to all other documents of that cluster.
  - When user will fire a query, it will be firstly matched with the cluster representative. If the query has relation with cluster representative, then it indicates that the documents which are member of the particular cluster may be part of the answer set with respect to the query.
  - It there is no match, the cluster will not handled further.
  - Once there is match with query and cluster representative, then each document from the cluster are checked for match with query. The documents which are logically near to query become part of answer set.

- Thus, using clustering, the searching is done at two different levels.
  1. **Level 1 :** Comparing query and cluster representative.
  2. **Level 2 :** Comparing query and actual document.
- In clustering, documents are grouped because they are in some sense related to each other; but more basically, they are grouped because they are likely to be wanted together, and logical relationship is the means of measuring this likelihood.
- The classification of documents can be done manually or via the intermediate calculation of a measure of closeness between documents. The first approach has proved theoretically to be intractable.

1.10 Measures of Association

University Question

Q. List with definition different measures of association. **SPPU : May 19, 4 Marks**

- To distribute the objects in different groups, the relationship between each pair of document is considered.
- The relationship will indicate that whether a particular document is semantically near to the second document as compared with other documents are not.
- The relationship between the documents can be mentioned using three different methods as shown in Fig. 1.10.1.

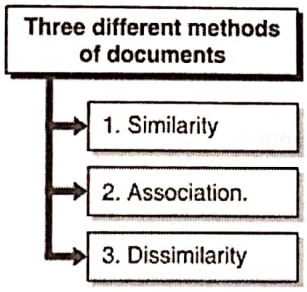


Fig. 1.10.1 : Methods of documents

1. Similarity

Similarity value indicates how much two documents or objects are near to each other.

2. Association

Association is same as similarity, but the difference is objects which are considered for comparison are the objects characterized by discrete-state attributes.

3. Dissimilarity

- Dissimilarity values show that how much far the objects are. Thus, the similarity value indicates the likeliness of two objects. If someone wants to find out the group of documents which are similar to each other, the similarity value can be considered.
- For the information retrieval system, we are interested in finding the subsets of the given documents. Documents in the collection are described using the list of index terms.
- Here for information retrieval systems, each pair of document is considered. Two documents will be similar to each other if they have more number of common index terms.

- If two documents are having less number of common index terms then obviously they will be semantically fa from each other. Hence we will not include such documents in the same group.
- To define the relation between the objects various measures of association are defined. The value of th measure will be more if two objects have more number of similar attributes.
- It follows that a cluster method depending only on the rank-ordering of the association values would giv identical clustering for all these measures.
- There are five measures of association methods. As we are using these measures for information retriev system, we should consider the representation of the document inside the system.
- Here the assumption is each document is represented by a list of keywords. A query is also represented by list of keywords.
- Each list of keyword is considered as one set.
- Thus the terms which are assumed here are :
  1. X : The list of index terms related to a document e.g. Document 1.
  2. Y : The list of index terms related to a document e.g. Document 2.
  3. | X | : The number of index terms present in the Document 1.
  4. | Y | : The number of index terms present in the Document 2
  5. ∩ : Intersection of two sets.

For example :

X : The list of index terms related to Document 1.

1.	Bat	2.	Ball
3.	Stump	4.	Pen
5.	Pencil	6.	Night
7.	Dog	8.	Cat
9.	Coat	10.	Fur

Y : The list of index terms related to Document 2.

1.	Pencil	2.	Paper
3.	Rubber	4.	Cat
5.	Mouse	6.	Book
7.	Eye	8.	Nose
9.	Heart	10.	Dark

- Here I am considering only the nouns. In real scenario the actual index terms may have original representatio of nouns, verbs, etc.

Thus :

$$|X| = 10$$

$$|Y| = 10$$

- *Now we will define different measure of association methods.*

1.11 Different Matching Coefficients

University Questions

Q. Describe different matching coefficient. SPPU : May 15, 8 Marks

Q. Write a short note on matching coefficients. SPPU : Dec. 16, 4 Marks

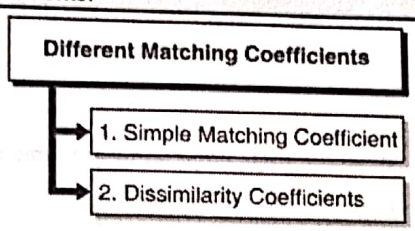


Fig. 1.11.1 : Different matching coefficients

1.11.1 Simple Matching Coefficient

- It is the number of shared index terms.
- Thus, we can calculate simple matching coefficient as

$$|X \cap Y|$$

- This method does not consider the size of X and Y
- In our example, the common terms are :
  1. Pencil
  2. Cat
- Thus, value of simple matching coefficient is 2.

1. Dice's coefficient

$$\frac{|X \cap Y|}{|X| + |Y|}$$

It is the shared number index terms divided by addition of sizes of both sets X and Y.

2. Jaccard's coefficient

$$\frac{|X \cap Y|}{|X \cup Y|}$$

It is calculated as shared index terms divided by union of set X and set Y.

3. Cosine coefficient

$$\frac{|X \cap Y|}{|X|^{1/2} \times |Y|^{1/2}}$$

Cosine coefficient can be calculated as number of common index terms divided by square root of size of X set plus square root of size of Y set.

4. Overlap coefficient

$$\frac{|X \cap Y|}{\min(|X|, |Y|)}$$

- Overlap coefficient can be calculated as number of common index terms divided by the size of set either Y having comparatively minimum entries.
- The Dice's coefficient, Jaccard's coefficient, cosine coefficient and overlap coefficients are normalized ver of simple matching coefficient. The values of these coefficients range from 0 to 1.
- It is necessary that the values of coefficients must be normalized. Following example presents importan normalized values.

Let,

1. $S1(X, Y) = |X \cap Y| \Rightarrow$ Simple matching coefficient which is not normalized.
2. $S2(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|} \Rightarrow$ normalized coefficient

Case 1

Let,

1. $|X| = 1$
2. $|Y| = 1$ and
3. $|X \cap Y| = 1$

Then, $S1(X, Y) = 1$

$$S2(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2.1}{1 + 2} = 1$$

Case 2

Let,

1. $|X| = 10$
2. $|Y| = 10$
3. $|X \cap Y| = 1$

Then, $S1(X, Y) = 1$

$$S2(X, Y) = \frac{2.|X \cap Y|}{|X| + |Y|}$$

$$= \frac{2.1}{10 + 10} = \frac{2}{20} = \mathbf{\frac{1}{10}}$$

- In first case, both the coefficients have same value i.e. 1, which indicates that there is exact match. But in sec case, even though there is a single common term present in both the set X and Y. Coefficient S1 has value 1 a which doesn't reflect any difference between case 1 and case 2 whereas value of S2 coefficient is 1/10, which g comparatively real scenario.

Ex. 1.11.1 : Let,

Document 1 = {CPU, keyboard, RAM, VGA, SMPS, USB, CD-ROM, Printer}

Document 2 = {CPU, VGA, Simulator, OS, Video, USB, Printer, Scanner, Compiler}

Find the similarity between two documents using different matching coefficients. **SPPU - May 15, 8 Ma**

Soln. :

Let,

X = Document 1

= {CPU, keyboard, RAM, VGA, SMPS, USB, CD-ROM, Printer} and

Y = Document 2

= {CPU, VGA, Simulator, OS, Video, USB, Printer, Scanner, Compiler}

Set (X ∩ Y) = {CPU, VGA, USB, Printer} and

(X union Y) = {CPU, keyboard, RAM, VGA, SMPS, USB, CD-ROM, Printer, Simulator, OS, Video, Scanner, Compiler}

Hence,

$|X| = 8$ and $|Y| = 9$

$|X \cap Y| = 4$

$|X \text{ union } Y| = 13$

Following are the similarity coefficients:

(i) Simple matching coefficient = $|X \cap Y| = 4$

(ii) Dice's coefficient = $\frac{|X \cap Y|}{|X| + |Y|} = 4/(8+9) = 4/17 = 0.23529412$

(iii) Jaccard's coefficient = $\frac{|X \cap Y|}{|X \cup Y|} = 4/13 = 0.30769$

(iv) Cosine coefficient = $\frac{|X \cap Y|}{|X|^{1/2} \times |Y|^{1/2}} = 4/(2.82 \times 3) = 0.472$

(v) Overlap coefficient = $\frac{|X \cap Y|}{\min(|X|, |Y|)} = 4/8 = 0.5$

1.11.2 Dissimilarity Coefficients

University Question

Q. Explain the properties of dissimilarity coefficients used in information retrieval. **SPPU : May 15, 8 Marks**

- Coefficients which are explained in previous section are based on similarity values.
- There are some coefficients defined, which are based on dissimilarity values between the documents.

Properties of dissimilarity coefficients

- Any dissimilarity function can be transformed into similarity function by a simple transformation of the form :

$$S = (1 + d)^{-1}$$

- But the reverse is not always true.
- If P is the set of objects to be clustered. A pair wise dissimilarity coefficient D is a function from P ×P to the non-negative real numbers.

- D (i.e. dissimilarity coefficient) satisfies following conditions :

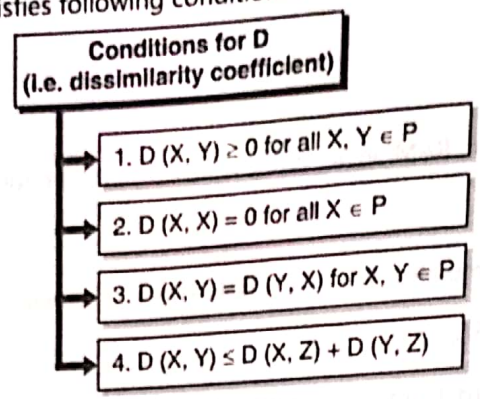


Fig. 1.11.2 : Condition for D

1. **D (X, Y) ≥ 0 for all X, Y∈ P**

   It says that the dissimilarity coefficient should be non-negative.

2. **D (X, X) = 0 for all X ∈ P**

   If we find the dissimilarity value by comparing same document by itself, the dissimilarity coefficient should have value 0 because there is exact match

3. **D (X, Y) = D (Y, X) for X, Y ∈ P**

   Dissimilarity value should not depend on the sequence in which we handle the documents. Thus the dissimilarity coefficient must be same between 2 documents, without considering the order of handling.

4. **D (X, Y) ≤ D (X, Z) + D (Y, Z)**

   This is based on theorem from Euclidian geometry which states that the sum of length of two sides of triangle is always greater than the length of the third side.

Dissimilarity coefficients example

- Examples of dissimilarity coefficient which satisfies above conditions are :

1. $\frac{|X \Delta Y|}{|X| + |Y|}$

Where, $(X \Delta Y) = (X \cup Y) - (X \cap Y)$

It is the symmetric difference of set X and Y.

This coefficient is simply related to Dice's coefficient by,

$$1 - \frac{2|X \cap Y|}{|X| + |Y|} = \frac{|X \Delta Y|}{|X| + |Y|}$$

The same coefficient can be represented in another form.

If each document is represented as the binary string where each entry represents an absence or presence of i^{th} keyword, which is indicated by zero or one in i^{th} position. In this case, the above dissimilarity coefficient can be represented like :

$$\frac{\sum x_i (1 - y_i) + \sum y_i (1 - x_i)}{\sum x_i + \sum y_i}$$

Where, the summation is over the total number of different keywords in the document collection.

2. Salton considered document representatives as the binary rectors embedded in an n-dimensional Euclidian space.

Where, n = Total number of index terms.

Thus, $$\frac{|X \cap Y|}{|X|^{1/2} \times |Y|^{1/2}}$$

Can then be interpreted as the cosine of the angular separation of the two binary rectors X and Y,

$$\therefore \cos\theta = \frac{(X, Y)}{\|X\| \cdot \|Y\|}$$

Where, (X, Y) = inner product

$\|\cdot\|$ = length of rector

If the space is Euclidian then for $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_n)$

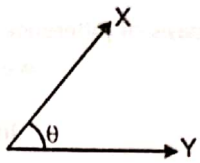


Fig. 1.11.3

We get, $$\frac{\sum_{i=1}^{n} x_i y_i}{\left(\sum_{i=1}^{n} x_i^2\right)^{1/2}\left(\sum_{i=1}^{n} y_i^2\right)^{1/2}}$$

3. Expected mutual information measure

Measure of association can be defined based on probabilistic model. It can be measures by the association between two objects by the extent to which their distribution deviate from stochastic independence. For two distinct probability distribution $P(x_i)$ and $P(x_j)$, the expected mutual information measure can be defined as follows :

$$I(x_i, x_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i) \cdot P(x_j)}$$

Properties of the function

i. When x_i and x_j are independent,

$$P(x_i)\, P(x_j) = P(x_i\, x_j)$$

So, $$I(x_i x_j) = 0$$

ii. $I(x_i x_j) < I(x_j x_i)$ which shows it is symmetric.

iii. It is invariant under one-to-one transformation of coordinates.

iv. $I(x_i, x_j)$ is often interpreted as a measure of the statistical information contained in x_i about x_j.

v. When we apply this function to measure the association between two index terms, say i and j, then x_i and x_j are binary variables. Thus $P(x_i = 1)$ will be the probability of occurrence of the term i and similarly $P(x_i = 0)$ will be the probability of its non-occurrence. The extent to which two index terms i and j are associated is then measured by $I(x_i, x_j)$. It measures the extent to which their distributions deviate from stochastic independence.

4. **Information Radius :** (dissimilarity between two classes of objects)

Dissimilarity between two classes of objects can be defined with a function. For example, we can discriminate two classes on the basis of their probability distribution over a simple two point space {1, 0}.

Let,

$P_1(1), P_1(0)$: Probability distribution associated with class I.

$P_2(1), P_2(0)$: Probability distribution associated with class II.

On the basis of difference between them, we measure the dissimilarity between class I and II by information radius.

Thus,

$$\text{Information radius} = uP_1(1) \log \frac{P_1(1)}{uP_1(1) + vP_2(1)} + vP_2(1) \log \frac{P_2(1)}{uP_1(1) + vP_2(1)}$$
$$+ uP_1(0) \log \frac{P_1(0)}{uP_1(0) + vP_2(0)} + vP_2(0) \log \frac{P_2(0)}{uP_1(0) + vP_2(0)}$$

Here, u and v are positive weights adding to units.

Properties

- Under some interpretation, the expected mutual information measure is a special case of the information radius.

For e.g.

- Let, $P_1(\cdot)$ and $P_2(\cdot)$ are two conditional distributions $P(\cdot/w_1)$ and $P(\cdot/w_2)$.

$$u = P(\cdot/w_1)$$
$$v = P(\cdot/w_2)$$

- Then for information radius we get,

$$P(x) = P(x/w_1) P(w_1) + P(x/w_2) P(w_2) ; x = 0, 1$$
$$P(x/w_i) = P(x/w_i) P(x), i = 1, 2$$

- We came to the expected mutual information measure $I(x_1 w_i)$.

1.12 Cluster Hypothesis

- Closely related documents tend to be relevant to the same requests.
- A basic assumption in retrieval systems is that documents relevant to a request are separated from those which are non-relevant.
- The relevant documents are more like one another than like non-relevant documents.
- This can be tested as follows : Compute the association between all pairs of documents :
  (a) Both of which are relevant to a request.
  (b) One of which is relevant and the other is non-relevant.
- Based on a set of requests, the relative distribution of relevant-relevant (R – R) and relevant - non-relevant (R - R) association of a collection can be defined.
- Plotting the relative frequency against strength of association for two hypothetical collections X and Y, we might get distribution as shown in Fig. 1.12.1.

- In Fig. 1.12.1, R-R is the distribution of relevant associate. R-N-R is the distribution of relevant non-relevant association.
- From graph we can conclude that :
  (1) Separation for collection X is good while for Y is poor.
  (2) Strength of association between relevant documents is greater for X than for Y.
- A linear search ignores the relationship exists between documents. Hence, structuring a collection in such a way that relevant documents will be part of one class will speed up the process of retrieval of the documents.

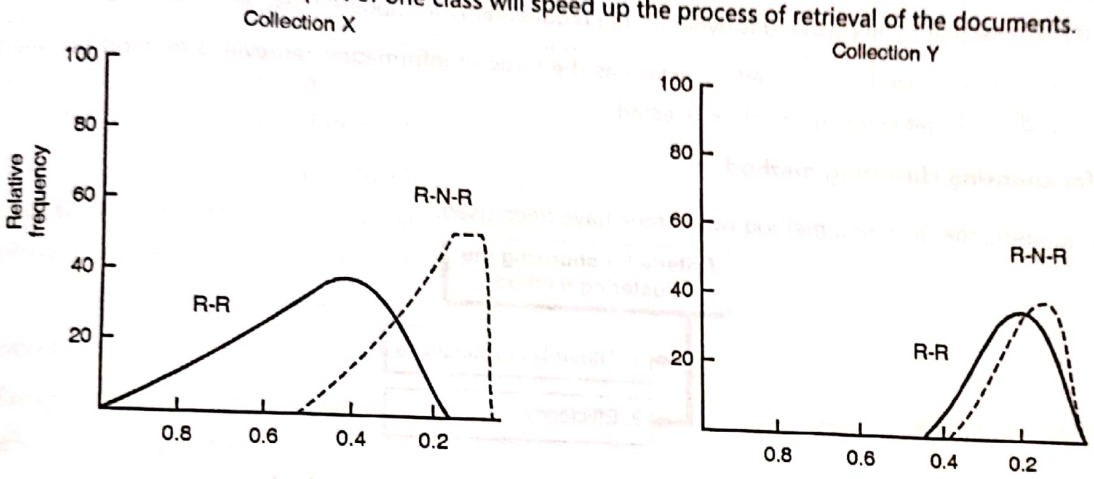


Fig. 1.12.1

- The searching will be more effective, since classes will contain only relevant documents and no non-relevant documents.
- Cluster hypothesis is based on the document descriptions. Hence the objects should be described in such a way that we can increase the distance between the two distributions R-R and R-N-R.
- We want to make more likely clear that we will retrieve relevant documents and less likely that we will retrieve non-relevant.
- Thus, cluster hypothesis is a convenient way of expressing the aim of such operations as document clustering. It does not say anything about how the separation is to be exploited.

1.12.1 Clustering in Information Retrieval

- 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.
- Cluster analysis can be performed on documents in several ways :
  (i) Documents may be clustered on the basis of the terms that they contain. The aim of this approach is to provide more efficient and more effective retrieval.
  (ii) Documents may be clustered based on co-occurring citations in order to provide insights into the nature of the literature of a field.
  (iii) Terms may be clustered on the basis of documents in which they co-occur. It is useful in construction of a thesaurus or in the enhancement of queries.

- Although cluster analysis can be easily implemented with available software packages, it may have some problems like :
  - 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 cluster or cluster hierarchies, which can be expensive in terms of computational resources.
  - Assessing the validity of the result obtained.
  - If the collection to be clustered is dynamic, the requirements for update must be considered.
  - If the aim is to use the clustered collection as the basis of information retrieval, a method for searching clusters or cluster hierarchy must be selected.

Criteria for choosing clustering method

While choosing the clustering method two criteria have been used.

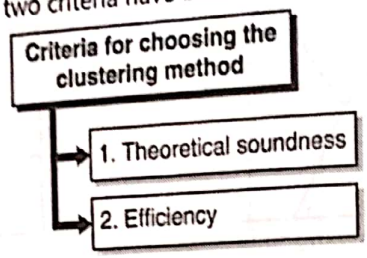


Fig. 1.12.2 : Criteria for choosing clustering method

1) Theoretical soundness

The clustering method should satisfy some constraints like :

(a) The method produces a clustering which is unlikely to be altered drastically when further objects are incorporated i.e. **it is stable under growth.**

(b) The method is stable in the sense that small errors in the description of the objects lead to small changes in clustering.

(c) The method is independent of the initial ordering of the objects.

2) Efficiency

The method should be efficient in terms of speed requirement and storage requirement.

1.13 Clustering Algorithm

- Clustering methods are usually categorized according to the type of cluster they produce. Thus, the clustering methods can be categorized as :
  - i. Hierarchical methods
  - ii. Non-hierarchical methods
- Hierarchical methods produce the output as ordered list of clusters. Whereas non-hierarchical methods produce unordered lists.

- Other categorization of the methods are
  i. The methods producing exclusive clusters
  ii. The methods producing overlapping clusters
- Here are some definitions related to clustering methods. While discussing about different algorithms we will use these terms.

1.13.1 Definitions

1. **Cluster :** A cluster is an ordered list of objects, which have some common characteristics.
2. **Distance between two clusters :** The distance between two clusters involves some or all elements of the two clusters. The clustering method determines how the distance should be computed.
3. **Similarity :** A similarity measure SIMILAR (d_1, d_j) can be used to represent the similarity between the documents. Typically similarity is normalized value which ranges from 0 to 1-Similarity generates values of 0 for documents exhibiting no agreement among the assigned index terms and 1 when perfect agreement is detected. Intermediate values are obtained for cases of partial agreement.
4. **Threshold :** The lowest possible input value of similarity required to join two objects in one cluster.
5. **Similarity matrix :** Similarity between objects calculated by the function SIMILAR (d_i,d_j) represented in the form of a matrix is called a similarity matrix.
6. **Dissimilarity coefficient :** The dissimilarity of two clusters is defined to be the distance between them. The smaller the value of dissimilarity coefficient, the more similar two clusters are.
7. **Cluster representative (seed) :** The representative of the cluster. Every incoming object's similarity is compared with cluster representative. A clustering method can have predetermined parameters like :
   1. The number of clusters desired.
   2. A minimum and maximum size for each cluster.
   3. A threshold value on the matching function, below which an object will not be included in cluster.
   4. The control of overlap between clusters.
   5. An arbitrarity chosen objective function which is optimized.

Now, let us discuss different clustering algorithms.

1.14 Rocchio's Algorithm

- Rochhio developed a clustering algorithm in 1966. It was developed on SMART project.
- Several **parameters** which are defined as the **input** for this algorithm are as follows :
  - Minimum and maximum documents per cluster.
  - Lower bound on the correlation between an item and a cluster below which an item will not be placed in cluster. This is a threshold that would be used in the final clean up phase of unclustered items.
  - Similarity coefficient : On new line the algorithm operates in three stages.

Stage 1 : Algorithm selects (by some criterion) a number of objects as cluster centres. The remaining objects are assigned to the centres or rag-bag cluster. Rag-bag is a temporary cluster used to are misfit objects.

On the basis of the initial assignments, the cluster representatives are computed and all objects are more assigned to clusters. The assignment rules are euplicitly defined in terms of thresholds on a matching function.

The final cluster may overlap (i.e an object may be assigned to more than one cluster).

⇩

Stage 2 : It is an iterative step. Here, the input parameters can be adjusted so that the resulting classification meets prior specification of such things as cluster size, etc. more nearly.

⇩

Stage 3 : This is the 'tidying up' stage. After the stage 2, the objects which are unassigned objects (i.e. part of rag-bag cluster) are forcibly assigned and overlap between clusters is reduced.

1.15 Single-Pass Algorithm

Process

Single-pass algorithm process as follows :

- The object descriptions are processed serially.
- The first object becomes the cluster representative of the first cluster.
- Each subsequent object is matched against all cluster representatives existing at its processing time.
- A given object is assigned to one cluster (or more if overlap is allowed) according to some condition on matching function.
- When an object is assigned to a cluster the representative for that cluster is recomputed.
- It an object fails a certain test it becomes the cluster representative of a new cluster.

Examples

Similarity matrix : Object = {1, 2, 3, 4, 5, 6}

1						
2	0.6					
3	0.6	0.8				
4	0.9	0.9	0.7			
5	0.9	0.6	0.6	0.9		
6	0.5	0.5	0.9	0.5	0.5	
	1	2	3	4	5	6

Threshold : 0.89

Case 1 : Clustering method : exclusive

Process object from 1 to 6.

1. **Object 1**

First object i.e. object 1 becomes part of cluster as well as cluster representative.

$$\therefore C_1 = \{1\}$$

2. **Object 2**

a. The object 2 is compared with object 1 (as it is cluster representative of C1) to check whether it can become part of first cluster.

Compare, similarity (1, 2) and threshold value.

$$\text{As } 0.6 < 0.89$$

Object 2 can't become part of cluster

b. Hence a new cluster is created whose cluster representative will be object 2.

$$\therefore C_1 = \{1\}.$$

$$C_2 = \{2\}$$

3. **Object 3**

a. Similarly (1, 3) < threshold

$$0.6 < 0.89$$

Hence object 3 can't become part of cluster 1.

b. Similarly (2, 3) < threshold

$$0.8 < 0.89$$

Hence object 3 can't become part of cluster 2.

c. Create new cluster whose cluster representative is object 3.

$$\therefore C_1 = \{1\}$$

$$C_2 = \{2\}$$

$$C_3 = \{3\}$$

4. **Object 4**

$$\text{Similarly } (1, 4) > \text{threshold}$$

$$0.9 > 0.89$$

∴Object 4 becomes part of cluster 1.

$$\therefore C_1 = \{1, 4\}$$

$$C_2 = \{2\}$$

$$C_3 = \{3\}$$

As the new element is added is cluster, cluster representative is again calculated. In this example there is no change, in cluster representative.

5. Object 5

Check similarity (1, 5) > threshold

0.9 > 0.89

$\therefore C_1 = \{1, 4, 5\}$

$C_2 = \{2\}$

$C_3 = \{3\}$

As new element is added in cluster 1, cluster representative is calculated again. Based on the similarity va
4, 5 objects. All objects are equidistant hence no change in cluster representative.

6. Object 6

a. Check similarity (1, 6) < threshold

$0.5 < 0.89$

b. Check similarity (2, 6) < threshold

$0.5 < 0.89$

c. Similarity (3, 6) > threshold

$0.9 > 0.89$

Hence, object 6 becomes part of cluster 3.

$\therefore C_1 = \{1, 4, 5\}$

$C_2 = \{2\}$

$C_3 = \{3, 6\}$

Here again for C_3, object 3 and 6 are equidistant hence no change in cluster representative.

Thus, the output of single pass algorithm is as follows :

$C_1 = \{1, 4, 5\}$

$C_2 = \{2\}$

$C_3 = \{3, 6\}$

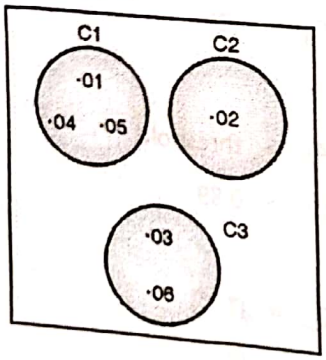


Fig. 1.15.1

When the clustering method is exclusive, once an object become part of a single cluster, handling of that
over.

Case 2

When clustering method is overlapping each object is compared with cluster representatives of each cluster.

⇒ Process objects from 1 to 6.

1. Object 1

As there is no cluster present, create new cluster whose cluster representative is object 1 itself.

$$\therefore C_1 = \{1\}$$

2. Object 2

a. Compare similarity (1, 2) < threshold

b. ∴ Create new cluster.

$$\therefore C_1 = \{1\}$$
$$C_2 = \{2\}$$

3. Object 3

a. Compare similarity (1, 3) < threshold

$$0.6 < 0.89$$

b. Compare similarity (2, 3) < threshold

$$0.8 < 0.89$$

c. Create new cluster.

$$\therefore C_1 = \{1\}$$
$$C_2 = \{2\}$$
$$C_3 = \{3\}$$

4. Object 4

a. Compare similarity (1, 4) > threshold

$$0.9 > 0.89$$

Hence object 4 can become part of cluster 1. As the method is overlapping, go on checking the object 4's similarity value with all cluster representative.

b. Compare similarity (2, 4) > threshold

$$0.9 > 0.89$$

Object 4 can become part of cluster C_2.

c. Compare similarity (3, 4) < threshold

$$0.7 < 0.89$$
$$\therefore C_1 = \{1, 4\}$$
$$C_2 = \{2, 4\}$$
$$C_3 = \{3\}$$

5. **Object 5**

a. compare similarity (1, 5) > threshold

$$0.9 > 0.89$$

b. compare similarity (2, 5) < threshold

$$0.6 < 0.89$$

c. Compare similarity (3, 5) < threshold

$$0.6 < 0.89$$

$$\therefore C_1 = \{1, 4, 5\}$$

$$C_2 = \{2, 4\}$$

$$C_3 = \{3\}$$

6. **Object 6**

a. Compare similarity (1, 6)

$$0.5 < 0.89$$

b. Compare similarity (2, 6) < threshold

$$0.5 < 0.89$$

c. Compare similarity (3, 6) < threshold

$$0.9 > 0.89$$

$$\therefore C_1 = \{1, 4, 5\}$$

$$C_2 = \{2, 4\}$$

$$C_3 = \{3, 6\}$$

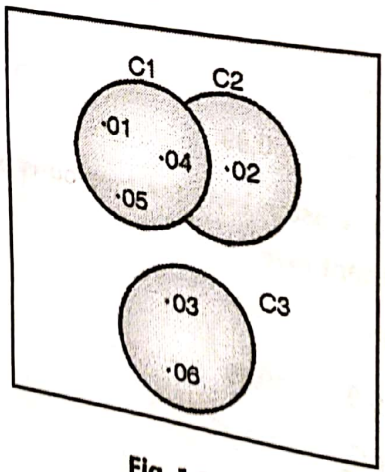


Fig. 1.15.2

Advantage of Single Pass Algorithm :

Simple for implementation.

Disadvantage of Single Pass Algorithm

The output depends on the sequence in which objects are handled

Ex.1.15.1 : Clusters the documents using single pass clustering algorithm for the following example. Threshold value is 10.

	Terms in document				
	T1	T2	T3	T4	T5
Doc 1	1	2	0	0	1
Doc 2	3	1	2	3	0
Doc 3	3	0	0	0	1
Doc 4	2	1	0	3	0
Doc 5	2	2	1	5	1

SPPU - May 16, 10 Marks

Soln. :

Step 1 : Start with Doc.1. As initially no cluster is present, Document 1introduces Cluster 1 i.e. C1.Hence,

$$C1 = \{Doc\ 1\}$$

Centroid of this cluster is<1, 2, 0, 0, 1>

C1= { Doc 1} ; Centroid of C1 : <1, 2, 0, 0, 1 >

Step 2 : Now we need to make decision for Doc.2 Either it can become part offirst cluster or it can introduce a new cluster.

For making the decision, we need to find similarity between centroid of first cluster and Doc. 2. Hence, we will use dot product for simplicity.

Centroid of C1 :<1, 2, 0, 0, 1 >

Doc. 2 :< 3, 1, 2, 3, 0>

$$SIM\ (Doc\ 2, C1) = 1*3 + 2*1 + 0*2 + 0*3 + 1*0 = 5$$

Now compare threshold value and

$$SIM(Doc\ 2, C1) = 10 > 5$$

Hence Doc 2 can't become part of first cluster. Hence, new cluster will be introduced.

C1= { Doc 1} ; Centroid of C1 : <1, 2, 0, 0, 1 >

C2 = {Doc 2} ; Centroid of C2 : < 3,1,2,3,0 >

Step 3 : Make decision for Doc.3

Hence,

Doc. 3 :< 3, 0, 0, 0, 1>

Centroid of C1 :< 1, 2, 0, 0, 1 >

$$SIM(Doc.3\ , C1) = 3*1 + 0*2 + 0*0 + 0*0 + 1*1 = 6$$

$$10 > 6,$$ hence Doc.3 can't be part of C1.

Now, check whether Doc.3 can become part of C2.Hence,

Doc. 3 :< 3, 0, 0, 0, 1>

Centroid of C2 :< 3, 1, 2, 3, 0>

SIM(Doc.3 , C2) = 3*3 + 0*1 + 0*2 + 0*3 + 1*0 = 9

Threhold 10> 9, hence, Doc 3 cant become part of cluster 2 also. Hene, introduce a new Cluster.

C3 = { Doc. 3)

C1= { Doc 1} ; Centroid of C1 : <1, 2, 0, 0, 1 >

C2 = {Doc 2} ; Centroid of C2 : < 3, 1, 2, 3, 0 >

C3= { Doc. 3}; Centroid of C3 :< 3, 0, 0, 0, 1>

Step 4 :

Now. Make decision for Doc. 4:< 2, 1, 0, 3, 0>

SIM(Doc.4 , C1) = 1*2 + 2*1 + 0*0 = 0*3 + 1*0 = 4

Threshold 10 >4 ; Doc.4 cannot become part of C1.

SIM(Doc.4, C2) = 3*2 + 1*1 + 2*0 + 3*3 + 0*0 = 16

Threshold 10 < 16, Hence Doc.4 will become part of C2.

C2 = {Doc.2, Doc. 4 }

As new document is included in cluster, Centroid of cluster is recalculated which is average of Doc.2 and Doc. 4

Doc. 2 :< 3, 1, 2, 3,0 >

Doc. 4:< 2, 1, 0, 3, 0 >

Centroid of C2 = <5/2,2/2, 2/2, 6/2, 0/2> = <2.5, 1, 1, 3, 0>

Thus Clusters available are:

C1= { Doc 1} ; Centroid of C1 : <1, 2, 0, 0, 1 >

C2 = {Doc 2, Doc. 4} ; Centroid of C2 : < 2.5, 1, 1, 3, 0>

C3= { Doc. 3}; Centroid of C3 :< 3, 0, 0, 0, 1>

Step 5 : Finally we need to find where Doc. 5 will fit,

Doc. 5:<2, 2, 1, 5, 1>

SIM(Doc.5, C1) = 1*2 + 2*2 + 0*1 + 0*5 + 1*1 = 7

Threshold 10 > 7, Hence Doc.5 cannot be part of C1

SIM(Doc. 5, C2) = 2.5*2 +1*2 + 1*1 + 3*5 + 0*1

= 5 +2 + 1+ 15 + 0 = 23

Threshold 10 < 23, Hence Doc.5 becomes part of C2.

Thus,

C2 = {Doc 2, Doc. 4, Doc. 5}

Centroid of C2 will be recalculated which is average of Doc.2, Doc.4 and Doc. 5. Hence

Doc. 2 :< 3, 1, 2, 3,0 >

Doc. 4:< 2, 1, 0, 3, 0 >

Doc. 5: <2, 2, 1, 5, 1>

Centroid of C2 = <7/3, 4/3, 3/3, 11/3, 1/3>

= <2.33, 1.33, 1, 3.66, 0.33>

Thus finally, we have 3 Clusters.

C1 = { Doc 1} ; Centroid of C1 : <1, 2, 0, 0, 1 >

C2 = {Doc 2, Doc. 4, Doc. 5};

Centroid of C2 :<2.33, 1.33, 1, 3.66, 0.33 >

C3 = { Doc. 3}; Centroid of C3 :< 3, 0, 0, 0, 1>

1.16 Single Link Algorithm

University Question

Q. Show how single link clusters may be derived from the dissimilarity coefficient by thresholding it.

SPPU : May 17, March 19, 5 Marks

- The single link method is the best known of hierarchical methods. It operates by joining at each step, the two most similar objects, which are not yet in the same cluster. The name **single link** refers to the joining of pairs of clusters by the single shortest link between them.
- The dissimilarity coefficient is the basic input to a single-link clustering algorithm. Single-link produces the output which is a hierarchy with associated numerical levels called a **dendogram.**
- The hierarchy is represented by a free structure. The dendogram and its respective tree is as shown in Fig. 1.16.1.

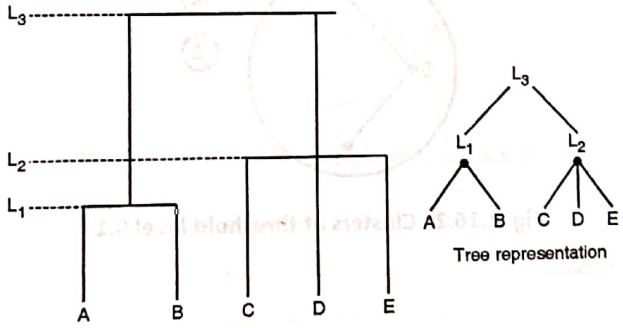


Fig. 1.16.1 : Dendogram

Here,

{A, B, C, D, E} are the objects clusters are :

At level 1 : {A, B}, {C}, {D}, {E}

At level 2 : {A, B} {C, D, E}

At level 3 : {A, B, C, D, E}

At each level of hierarchy a set of classes can be identified. As we move up in hierarchy, the classes at lower level are nested in the classes at higher levels.

Example : Objects = {1, 2, 3, 4, 5}

Dissimilarity matrix

2	0.4			
3	0.4	0.2		
4	0.3	0.3	0.3	
5	0.1	0.4	0.4	0.1
	1	2	3	4

Binary matrices generated at various levels.

a. Threshold = 0.1

Binary matrix

2	0			
3	0	0		
4	0	0	0	
5	1	0	0	1
	1	2	3	4

Graph and cluster

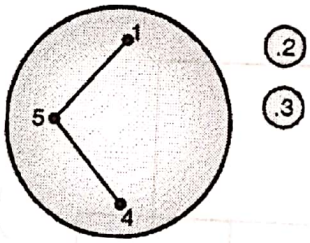


Fig. 1.16.2 : Clusters at threshold level 0.1

b. Threshold = 0.2

Binary matrix

2	0			
3	0	1		
4	0	0	0	
5	1	0	0	1
	1	2	3	4

Graph and cluster

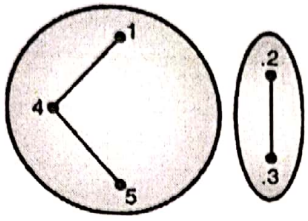


Fig. 1.16.3 : Clusters at threshold valve 0.2

c. Threshold = 0.3

Binary matrix

2	0			
3	0	1		
4	1	1	1	
5	1	0	0	1
	1	2	3	4

Graph and cluster

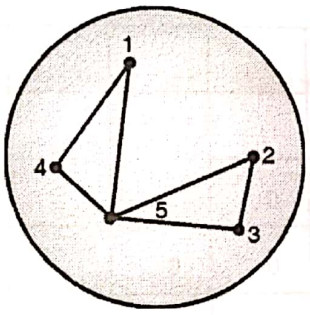


Fig. 1.16.4 : Clusters at threshold value 0.3

- As all objects are grouped under single cluster, we stop.
- The graph at each level is given by a set of vertices corresponding to the objects to be clustered, and any two vertices are linked if their dissimilarity at most equal to the value of level L.
- Thus, the single link can be defined in term of graph as at any level a single-link cluster is precisely the set of vertices of a connected component of the graph of that level. Note that whereas the graphs at any two district values taken by the dissimilarity coefficient will be different. This is not necessary in the case of clusters at those levels.
- It may possible that by increasing the level the links introduced between vertices do not change the total number of connected vertices in a component.
- E.g. clusters at level 0.3 and 0.4 are same. The hierarchy is created by varying the level from the lowest possible value, increasing it through successive values of the DC until all objects are contained in one cluster.

- For an object are contained in one cluster. For an object to belong to a cluster, it needs to be linked to only ... other member of cluster.

Ex. 1.16.1 : Dissimilarity matrix is given as follows.

1						
2	0.6					
3	0.6	0.8				
4	0.9	0.9	0.7			
5	0.9	0.6	0.6	0.9		
6	0.5	0.5	0.9	0.5	0.5	
	1	2	3	4	5	6

Threshold 0.4, 0.6, 0.8, 0.9.

Apply single link algorithm and calculate cluster for above 6 objects. **SPPU : May 19, 5 Ma...**

Soln. :

a. At level 1 : Threshold = 0.4

Binary matrix

1						
2	0					
3	0	0				
4	0	0	0			
5	0	0	0	0		
6	1	1	0	0	0	
	1	2	3	4	5	6

Graph and cluster

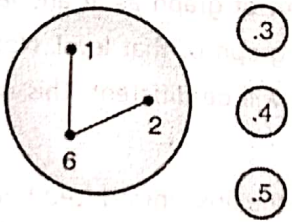


Fig. P.1.16.1 : Clusters at threshold valve 0.4

b. At level 2 : Threshold = 0.6

Binary matrix

1						
2	1					
3	1	0				
4	0	0	0			
5	0	1	1	0		
6	1	1	0	1	1	
	1	2	3	4	5	6

Graph and cluster

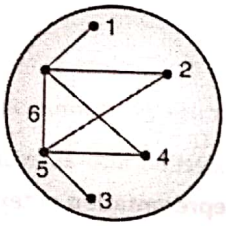


Fig. P.1.16.1(a) : Clusters at threshold valve 0.6

As all objects are grouped in single cluster, we stop.

Review Questions

Q. 1 Differentiate between Data Retrieval and Information Retrieval.

Q. 2 Explain Luhn's Idea.

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

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

Q. 5 Explain index term weighting (exhaustively of indexing and the specificity of the index language).

Q. 6 Explain conflation algorithm in detail.

2 Indexing and Searching Techniques

UNIT - II

Syllabus

Indexing : Inverted file, Suffix trees & suffix arrays, Signature Files, Scatter storage or hash addressing.
Searching Techniques : Boolean Search, sequential search, Serial search, cluster-based retrieval, Query languages, Types of queries, Patterns matching, structural queries.
IR Models : Basic concepts, Boolean Model, Vector Model, Probabilistic Model

2.1 Indexing and Index Term Weighting

2.1.1 Indexing

- During "Indexing" documents are prepared for use by Information Retrieval system.
- This means preparing the raw document collection into an easily accessible representation of documents. Th transformation from a document text into a **representation** of text is known as **indexing** the documents.
- Transforming a document into an indexed form involves the use of :
  - A library or set of regular expressions
  - Parsers
  - A library of stop words (a stop list)
  - Other miscellaneous filters
- Conflation algorithm is used for converting document into its representation i.e. indexing. Each element i.e. word index language is referred as index term.
- Index language may be pre-coordinate or post-coordinate. In pre-coordinate, the terms are coordinated at t time of indexing. A logical combination of any index terms may be used as a label to identify a class of document
- In post-coordinate, the terms are coordinated at the time of searching. In post-coordinate indexing the same cla would be identified at search time by combining the classes of documents labelled with the individual index term
- The vocabulary of an index language may be controlled or uncontrolled. The former refers to a list of approve index terms that an indexer may use.
- The controls on the language may also include hierarchic relationships between the index terms. Or, one may ins that certain terms can only be used as adjective. There is really no limit to the kind of syntactic controls one m put on a language.
- The index language which comes out of the conflation algorithm is uncontrolled, post-coordinate and derived.
- The vocabulary of index terms at any stage in the evolution of document collection is just the set of all conflati class names.

2.1.2 Index Term Weighting

University Question

Q. Describe index term weighing.

SPPU : May 15, 8 Marks

- Weighting is the final stage in most Information Retrieval indexing applications. For each index term a weight value get assigned which indicates the significance of that index term w.r.to the document.
- The two important factors governing the effectiveness of an index language are **exhaustivity** of indexing and **specificity** of index language.

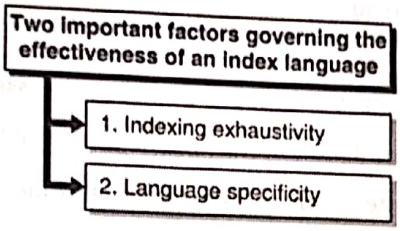


Fig. 2.1.1 : Factors governing effectiveness of index language

1. Indexing exhaustivity

- It is defined as the number of different topics indexed. A high level of exhaustivity of indexing leads to high recall and low precision.
- A low level of exhaustivity leads to low recall and high precision. In short, exhaustivity is the no. of index terms assigned to the document.

2. Language specificity

- It is the ability of the index language to describe topics precisely. It is the level of precision with which a document is actually indexed.
- High specificity leads to high precision and low recall. Specificity means the no. documents to which a given term is assigned in a given collection.
- Refer to the Luhn's idea. He has mentioned the discrimination power of index terms as a function of the rank order of their frequency of occurrence.
- The highest discrimination power being associated with the middle frequencies.
- Considering this idea, each index term is assigned with a weight value which indicates the significance of the index term in the document.
- A frequency count is how many times the index term comes in the document, can be considered as the weight value.
- Different methods are present to find the weight value of index terms. First is to assign the frequency count as the weight value.
- Another way is based on the index term distribution in the entire collection.

Let, N = Total no. of documents

n = The document in which an index term accurse;

Then the weight can be calculated as – $\log\left(\frac{N}{n}\right) + 1$

- If we compare two methods. The document frequency weighting places emphasis on content description whereas the second method i.e. weighting by specificity attempts to stress the ability of terms to discriminate one document from another.
- Salton and Yong combined both the methods of weighting considering **inter** document frequencies and intra document frequencies.
- By considering both, the total frequency of occurrence of a term and its distribution over the documents, that is how many times it occurs in each document, they concluded many things like.
  1. A term with high total frequency of occurrence is not very useful in retrieval irrespective of its distribution.
  2. Middle terms are most useful particularly, if the distribution is skewed.
  3. Rare terms with a skewed distribution are likely to be useful but less so than the middle frequency ones.
  4. Very rare terms are also quite useful but come bottom of the list except for the ones with a high total frequency.
- A **"good"** index term is one which, when assigned as an index term to a collection of documents, renders the documents as dissimilar as possible, where as a **"bad"** index term is one which renders the documents more similar.
- This is quantified through a **term discrimination value** which for a particular term measures the increase or decrease in the average dissimilarity between documents on the removal of that term. Therefore, a good term is one which on removal from the collection of documents, leads to a decrease in average.
- Dissimilarity, whereas a bad term is one which leads on removal to an increase.
- The idea is that a greater separation between documents will enhance retrieval effectiveness but then less separation will depress retrieval effectiveness.

2.2 Inverted Files

University Questions

- **Q.** Explain inverted index file. How it can be used in Information Retrieval. **SPPU : May 12, 8 Marks**
- **Q.** Explain the concept of Inverted index file. How it can be used in Information Retrieval. **SPPU : May 13, 8 Marks**
- **Q.** What is inverted file ? How it can be used in Information Retrieval? **SPPU : Dec. 13, 8 Marks**
- **Q.** Explain inverted file index concept with example? **SPPU : Dec. 14, 8 Marks**
- **Q.** Draw the generalized structure of an inverted file? Explain the algorithm for builtding an inverted file of a given document. **SPPU : May 15, 10 Marks**

2.2.1 Introduction

- In order to speed up the searching task, a word-oriented mechanism for indexing a text collection is required and that is called as inverted file or inverted index.
- The inverted file structure consists of two elements :
  1. Vocabulary
  2. Occurrences

- The vocabulary is the set of all different words in the text.
- For each word a list of all the text positions where the word appears is stored. And the set of all those lists is called the occurrences.
- Example of inverted file is as shown in Fig. 2.2.1.
- In the Fig. 2.2.1, a sample index and an inverted index built on it is shown. The words are converted to lower case and some are not indexed. It also shows the occurrences point to character positions in the text.
- The positions can refer to words or characters.
  - Word positions (i.e. position I refers to the i^{th} word) is phrase and proximity queries.
  - Character positions (i.e. positions i is the r^{th} character) facilitate direct access to the matching text positions.
- The space required for the vocabulary is small and it grows as O (n^{β}) where β is constant in between O and L dependent on the text.

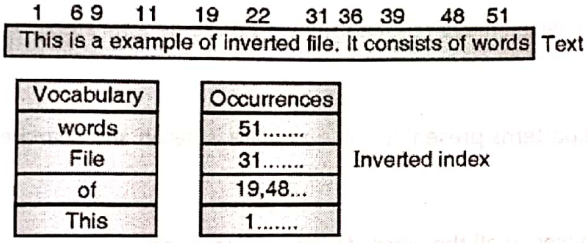


Fig. 2.2.1 : Sample text and inverted index

- The space required for the occurrences is more because each word appearing in the text is referenced once in that structure so extra space is O (n).
- To reduce space requirements, a technique called block addressing is used.
- In block addressing.
  1. The text is divided into blocks.
  2. The occurrences point to the blocks where the word appears.
  3. The classical indices which point to the exact occurrences are called 'full inverted indices'.
- By using block addressing
  1. Pointers are smaller because there are fewer blocks than positions.
  2. All the occurrences of a word inside a single block are collapsed to one reference.
- This is shown in Fig. 2.2.2. In Fig. 2.2.2, the sample text is split into four blocks and an inverted index using block addressing built on it. The occurrences denote block numbers.

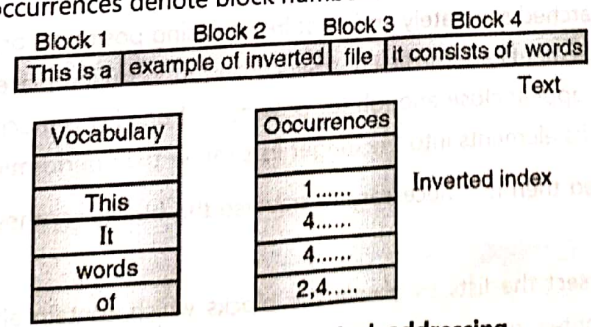


Fig. 2.2.2 : By using block addressing

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

2.2.2 Searching

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

1. Vocabulary search

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

2. Retrieval of occurrences

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

3. Manipulation of occurrences

- In this the occurrences are processed to solve phrases. proximity or Boolean operations.
- If block addressing is used then we can search directly the text to find information mission from t occurrences.
- Searching on an inverted index always starts in the vocabulary. Because it is good to have it in separate file that separate file can fits in main memory even for large collections.
- To speed up the search. Single-word queries can be searched using any suitable data structure like hashin Tries or B-trees.
- But storing the words in lexicographical order is cheaper in space and very competitive in performance as t word can be binary searched at O (log n) cost.
- Prefix and range queries can also be solved with binary search, tries or B-tree but not with hashing.
- If the query is formed by single words, then the process ends by delivering the list of occurrences.
- It is more difficult to solve context queries with inverted indices.
- Each element must be searched separately and a list in increasing positional order must be generated for eac one. Then the lists of all elements are traversed in synchronization to find places where all the words appear i sequence for a phrase or appear close enough for proximity. If one list is much shorter than the others then i is better to binary search its elements into the longer lists rather than performing a linear merge.
- If block addressing is used then it is necessary to traverse the blocks for these queries because the positio information is needed.

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

2.2.3 Construction

- An inverted index on a text of a characters can be built in O(n) time.
- All the vocabulary is kept in a trie data structure and for each word a list of its occurrences are stored.
- Each word of the text is read and searched in the trie. If it is not found then it is added to the trie with an empty list of occurrences and the new position is added to the end of its list of occurrences.
- This is shown in Fig. 2.2.3.

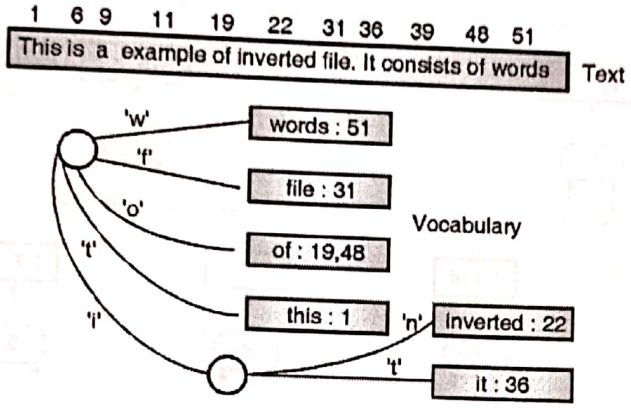


Fig. 2.2.3 : Building an inverted index

- Once the text is exhausted, the trie is written to disk together with the list of occurrences.
- The index is splitted into two files.
  1. In first file the lists of occurrences are stored contiguously and the file is called as 'posting file'.
  2. In the second file the vocabulary is stored in lexicographical order and for each word a pointer to its list in the first file is included.
- This allows the vocabulary to be kept in memory at search time in many cases and also the number of occurrences of a word can be immediately known from the vocabulary with less or no space overhead.
- Using this scheme, the overall process take O(n) time in worst case. In the trieO(1) operations are performed per text character and the positions can be inserted at the end of the lists of occurrences in O(1) time.
- But the above discussed scheme is not suitable for large texts where the index does not fit in the main memory and the paging mechanism used reduce the performance of the algorithm.
- So alternative to above scheme is discussed below.
- When no memory is available then the partial index I_i obtained is written to disk and erased from main memory before continuing with the test of the text.
- So, finally we will have a number of partial indices I_i exist on disk. These indices are then merged in hierarchical manner.
- Indices I_1 and I_2 are merged to obtain I_{1-2}, I_3 and I_4 are merged to obtain I_{3-4} and so on. So in this way the resulting partial indices are approximately twice the size.
- When all the indices at one level are merged then merging proceeds for the next level. Joining the index I_{1-2} with the index I_{3-4} to form I_{1-4}. This is continued until there is only one index comprising the whole text.

- This is shown in Fig. 2.2.4.
- In the Fig. 2.2.4 rectangles are used to represent partial indices, rounded rectangles are used to represent merg operations. The numbers inside the margin operations show a possible merging order.

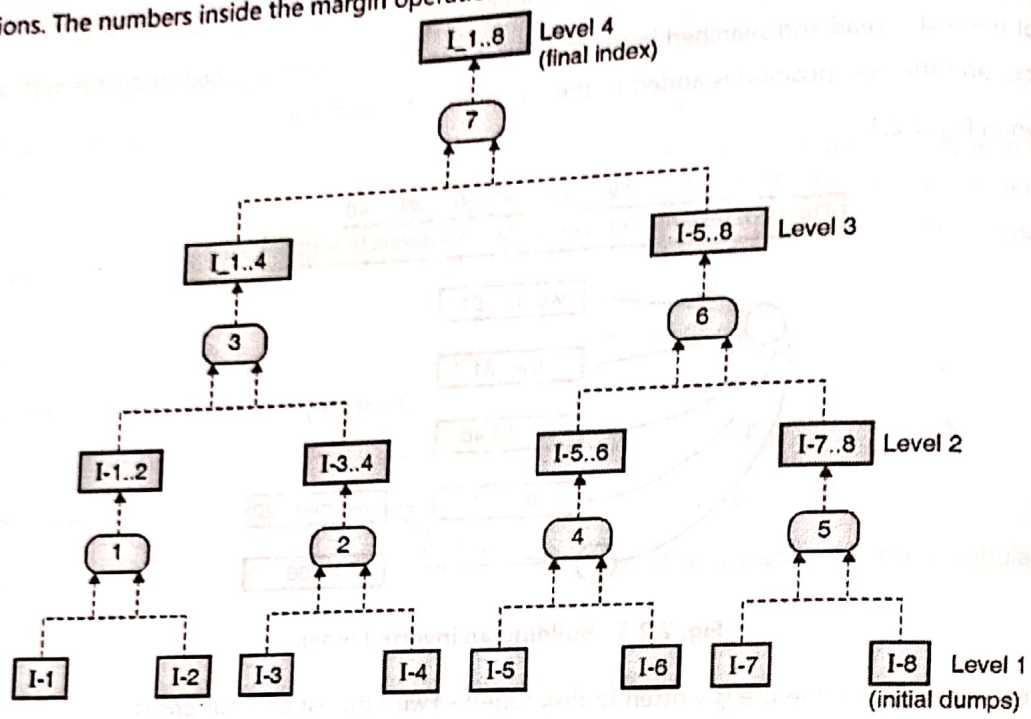


Fig. 2.2.4 : Merging the partial indices in a binary fashion

- Merging two indices consists of merging the sorted vocabularies and whenever the same word appears in bo indices then both lists of occurrences are merged.
- This approach is very fast and its complexity is O (n_1 + n_2) where, n_1 and n_2 are the sizes of the indices.
- The total time required to generate the number of partial indices O (n / M) is O(n). Each level of merging perfor a linear process over the whole index with the cost of O(n). To merge the O(n/M) partial indices $\log_2$ (n/M) mergi levels are necessary. So total cost of the algorithm is O (n log (n/M)).
- More than two indices can be merged at once, this will not change the complexity but it improves the efficien because less merging levels exist.
- Also the memory buffers for each partial index to merge will be smaller and more disk seeks will be performed.
- In practice it is a good idea to merge 20 partial indices at once.
- We can perform in-place merging to reduce build-time space requirements. When two or more indices a merged. Write the result in the same disk blocks of the original indices instead of on a new file.
- It is also possible to perform the hierarchical merging as soon as the files are generated. This reduces the spac requirements because the vocabularies are merged and redundant words are eliminated.
- If block addressing is used, this algorithm changes very little. So with block addressing maintenance is cheap.
- It new text of size n' is added to the database, then building inverted index for the new text and merging both th indices takes O (n + n' log (n' | M)). Deleting text can be done by an O(n) pass over the index eliminating th occurrences that point inside eliminated text areas.

2.3 Suffix Trees and Suffix Arrays

University Question

Q. Explain suffix array with the help of diagram. **SPPU : May 14, 4 Marks**

2.3.1 Introduction

- Inverted indices see the text as sequence of words and this restricts the kinds of queries than can be answered. Similarly other queries like phrases are expensive to solve. And the concept of words does not exist in some applications like genetic databases.
- Suffix arrays are space efficient implementation of suffix trees. Suffix arrays allow and answer more complex queries very efficiently.
- This structure can be used to index only words as the inverted index as well as to index any text character.
- Main drawback of this structure is construction process is very costly, the text must be readily available at query time and the results are not delivered in text position order.
- This data structure is suitable for a wider spectrum of applications such as generic databases. But for word-based applications, inverted files perform better.
- This index considers the text as one long string. Each position in the text is considered as a text suffix.
- Text suffix is a string that goes from that text position to the end of the text.
- Each suffix is uniquely identified by its position.
- There is no need to index all text positions so it is possible to index only word beginnings to have functionality similar to inverted indices.
- Index points are selected from the text, which points to the beginning of the text positions which will be retrievable and those elements which are not index points are not retrievable. Fig. 2.3.1 illustrates this.

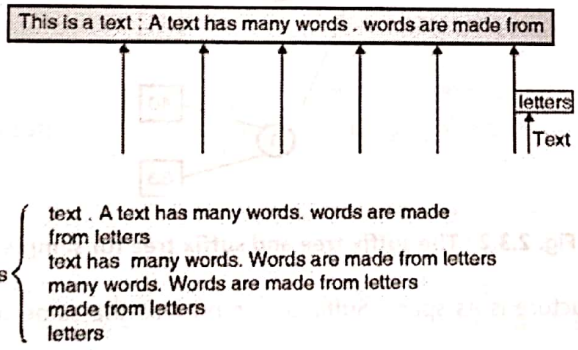


Fig. 2.3.1 : Sample text with index points and suffixes corresponding to index points

2.3.2 Structure of Suffix Tree

University Question

Q. Explain working of suffix tree. **SPPU : Dec. 16, 3 Marks**

- Suffix tree is a trie data structure which is built over all the suffixes of the text.
- The pointers to the suffixes are stored at the leaf nodes. For better space utilization, trie structure is compacted into a patricia tree.

- It involves compressing unary paths i.e. paths where each node has just one child. An indication of the character position to consider is stored at the nodes which root a compressed path.
- If unary path are absent then the tree has O(n) nodes instead of the worst case $O(n^2)$ of the trie.
- Fig. 2.3.2 shows the suffix trie and suffix tree for the sample text.

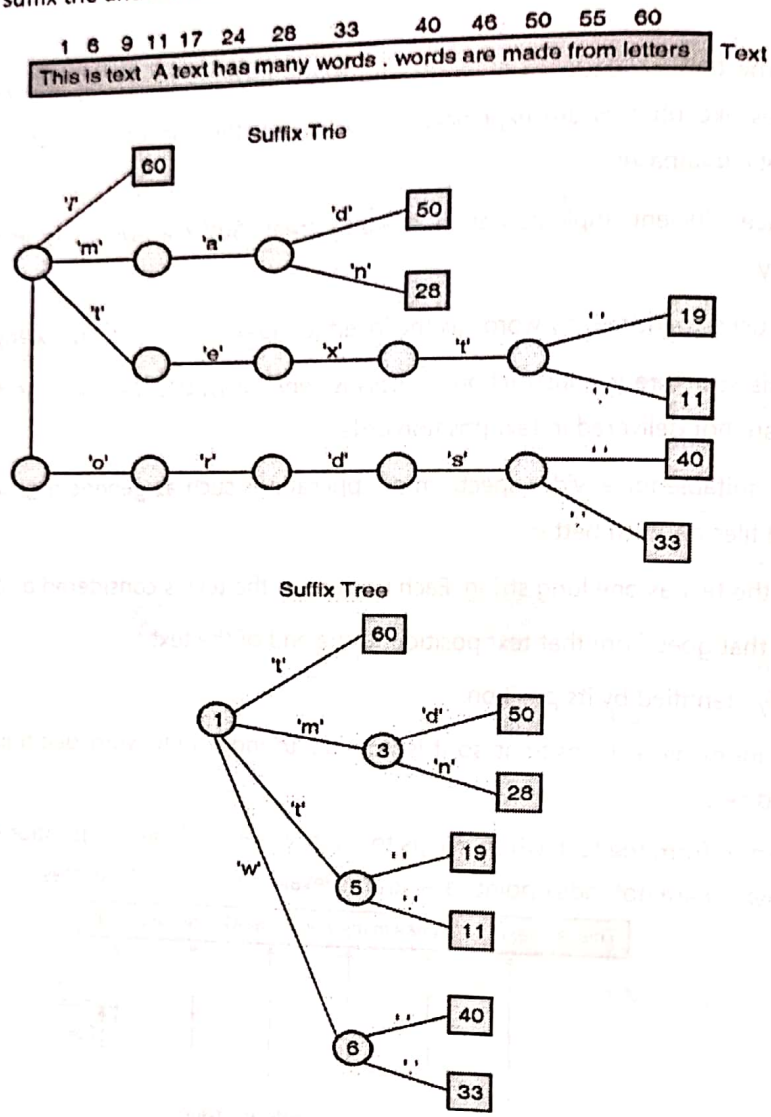


Fig. 2.3.2 : The suffix tree and suffix tree for sample text

- The problem with this structure is its space. Suffix arrays provide the same functionality as suffix trees with space requirements.
- If the leaves of the suffix tree are traversed in left-to-right order then all the suffixes of the text are retrieva lexicographical order.
- A suffix array is like a simple array and it contains all the pointers to the text suffixes listed in lexicographical orde
- This is shown in Fig. 2.3.3. As suffix array stores one pointer per indexed suffix so the space requirements almost the same that of inverted indices.
- Suffix arrays are designed to allow binary searches by comparing the contents of each pointer. If the suffix array large then this binary search performs poorly because of the number of random disk accesses. To avoid t problem supra-indices over the suffix array has proposed.

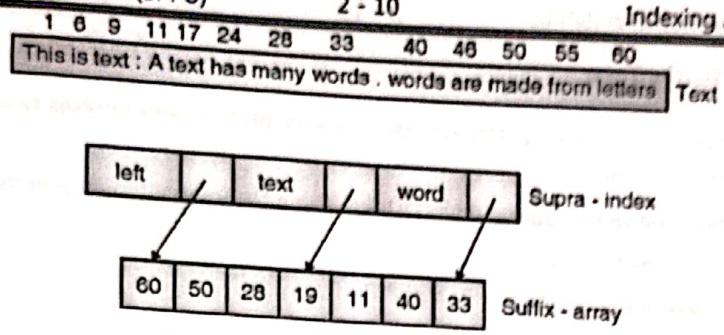


Fig. 2.3.3 : Suffix array for sample text

- In supra-indices a sampling of one out of b suffix array entries where for each sample *l* and suffix characters are stored in the supra-index.
- This supra-index is then used as a first step of the search to reduce external accesses.
- Fig. 2.3.4 shows supra-index over suffix array. In this figure one out of three entries are sampled. Keeping their first four characters.
- In supra-index there is no need to take samples at fixed intervals or of the same length. For word-indexing suffix array.

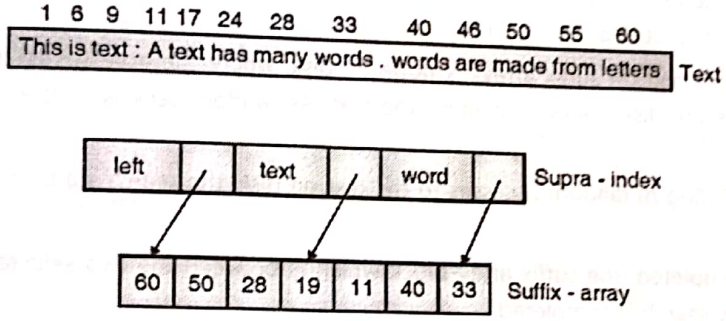


Fig. 2.3.4 : Supra index over suffix array

- A new sample could be taken each time the first word of the suffix changes and word is stored instead of *l* characters.
- The difference between suffix and inverted index is that, the occurrences of each word in an inverted index are sorted by text position. While in suffix array they are sorted lexico-graphically by the text following the word. This is shown in Fig. 2.3.5.
- The space requirements of the suffix array with a vocabulary supra index are exactly the same as for inverted indices.

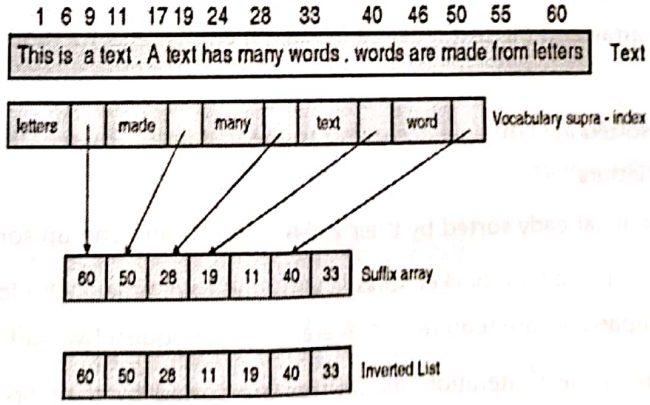


Fig. 2.3.5 : Relation between inverted list and suffix array and suffix array with vocabulary supra-index

2.3.3 Searching in Suffix Tree

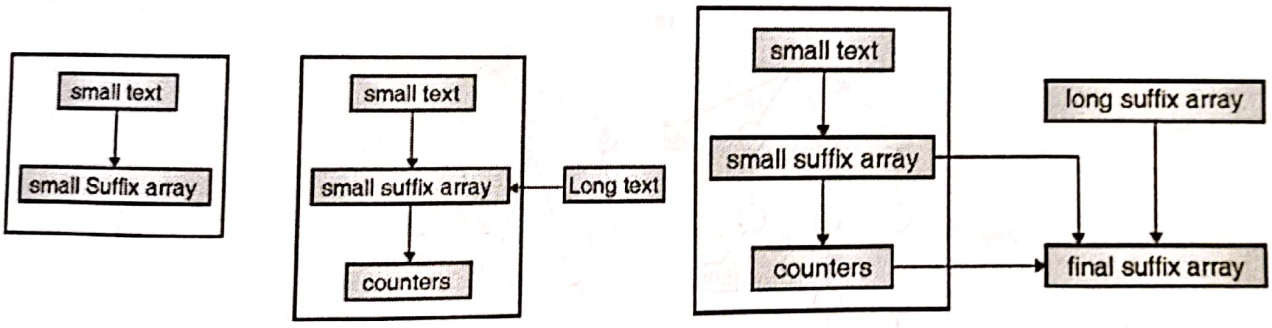
- By using simple trie search, many basic patterns such as words, prefixes and phrases can be searched in O(
if suffix tree on the text is afforded.
- Suffix array perform the same search operation in O(1.9 m) time by doing a binary search instead of a tree se
- This searching is performed as follows.
  1. Search starts with two 'limiting patterns' P_1 and P_2.
  2. We have to search for any suffix s such that $P_1 \leq S \leq P_2$.
  3. Both limiting patterns are binary searched in the suffix array.
  4. All the elements lying between both positions point to exactly those suffixes that start like the original pa
  5. For example in Fig. 2.3.5 to find the word 'text' who have to search for 'text' and 'texu' obtain the port
the array that contains the pointers 19 and 11.
- Simple phase searching is good for this kind of indices. A simple phrase of words can be searched as if it w
simple pattern because the suffix tree or array sorts with respect to the complete suffixes and not only the
word.
- A proximity search has to be solved element-wise. The matches for each element must be collected and sorted
then they have to be intersected as for inverted files.
- The binary search performed on suffix arrays is done on disk, where the accesses to text positions force a
operation which spans the disk tracks containing the text. As random seek is O(n) in size, binary search co
O(n log n) time.
- To avoid performing O (log n) random accesses to the text on disk, the search starts in the supra-index which
in main memory.
- After this search is completed, the suffix array block which is between the two selected samples is bought
memory and the binary search is completed.

2.3.4 Construction in Main Memory

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

2.3.5 Construction of Suffix Arrays for Large Texts

- Large text database will not fit in main memory so for such databases it is possible to apply an external memory sorting algorithm.
- But in such kind of sorting algorithm each comparison involves accessing the text at random positions on the disk and this random access decreases the performance of the sorting process.
- So the algorithm is especially designed for large texts and it consists of following steps.
  1. Divide the text into different blocks that can be sorted in main memory.
  2. For each block build suffix array in main memory.
  3. Build the suffix array for first block.
  4. Build the suffix array for second block.
  5. Merge the suffix arrays of both the blocks.
  6. Build the suffix array for the third block.
  7. Merge this suffix array with the array formed by merging suffix array of first and second block.
  8. Then perform this building suffix array and merging with previous suffix array for each and every block.
- It is difficult to merge large suffix array with small suffix array because for merging we have to compare text positions which are spread in a large text.
- We can do this by determining the number of elements of the large array to be placed between each pair of elements in the small array and then using this information to merge the arrays without accessing the text and counters are used to stoke this information.
- The counters are computed without using large suffix array. The text corresponding to large suffix array is sequentially dead into main memory.
- Then each suffix of that text is searched in the small suffix array. Once we know the inter-element position where the suffix lie we have to increment the appropriate counter. This is illustrated in Fig. 2.3.6.
- Fig.2.3.6 shows the building of suffix array, computation of counters and merging of suffix arrays.



(a) Building of local suffix array **(b) Computation of counters** **(c) Merging of suffix array**

Fig. 2.3.6 : Construction of suffix array for large sets

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

2.3.6 Difference between Suffix Array and Suffix Tree

University Questions

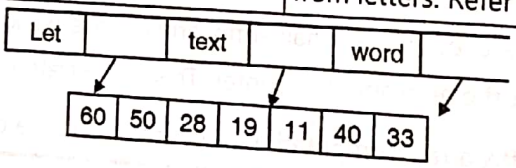
Q. Explain the difference between suffix array and suffix tree. SPPU : May 13, 8 Marks

Q. Compare with example suffix array and suffix tree. SPPU : May 15, 8 Marks

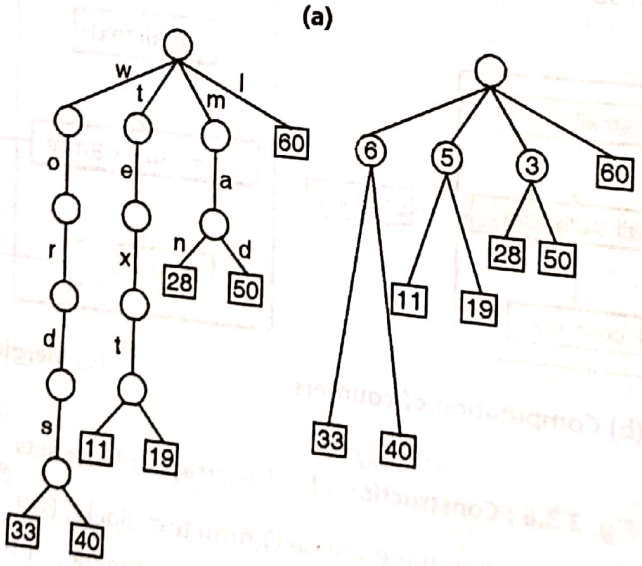
Q. Give the difference between suffix array and suffix tree. SPPU : May 16, 5 Marks

Table 2.3.1

Sr. No.	Suffix Array	Suffix Tree
1.	Memory efficient-Requires less memory	Requires more memory.
2.	Can answer more complex queries.	Cannot answer complex queries.
3.	Construction is costly.	Construction is cheap.
4.	Results are not delivered in text position order.	Results are delivered in text position order.
5.	Binary search is possible.	Binary search is not possible.
6.	Uses supra index for fast retrieval.	Uses Patricia tree to save memory.
7.	Uses Array, linear data structure for construction.	Uses tree, non linear data structure for Construction.
8.	Uses Arrays as supporting data structures.	Uses trie as supporting data structures
9.	Can be used for large texts.	Not practical for large texts.
10.	Example 1 11 19 28 33 40 46 50 60 This is a text. A text has many words. Words are made from letters. Refer Fig. 2.3.7(a).	Example- 1 11 19 28 33 40 46 50 60 This is a text. A text has many words. Words are ma from letters. Refer Fig. 2.3.7(b).



(a)



(b)

Fig. 2.3.7

2.4 Signature Files

University Questions

Q. Explain Signature file with example. SPPU : Dec. 12, 8 Marks

Q. What is signature file? How can it be constructed? SPPU : May 14, 4 Marks

- Signature files are index structured based on hashing and are word oriented.
- They posses a low overhead at the cost of forcing a sequential search over the index.
- Search complexity is linear but constant.
- It is not suitable for very large texts.

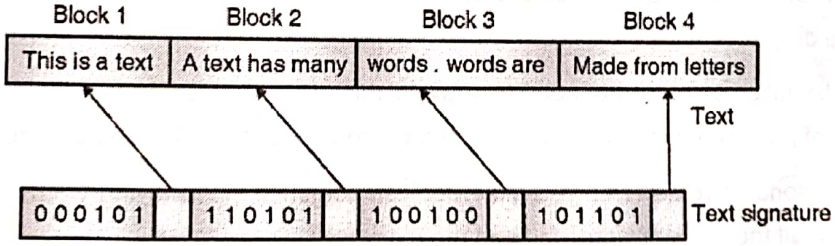
2.4.1 Structure of Signature Files

University Questions

Q. Explain signature structure in detail. SPPU : May 19, 5 Marks

Q. Explain with example signature file structure. SPPU : May 15, 8 Marks

- Signature file used hash function which maps words to bit masks of B bits.
- It divides text in blocks of 3 words each.
- Then it assigns a bit mask of size B to each text block of size b.
- The mask is obtained by performing bitwise ORing the signatures of all the words in the text block.
- So signature file is the sequence of bits masks of all blocks with a pointer to each block.
- If the word is present in a text block, then all the bits set in its signature are also set in the bit set in its signature are also set in the bit mask of the text block.
- Whenever a bit is set in the mask of the query word and not in the mask of the text block then the word is not present in the text block.
- Fig. 2.4.1 shows the example. In this, the sample text is cut into blocks.



h (text)	= 0 0 0 1 0 1
h (many)	= 1 1 0 0 0 0
h (words)	= 1 0 0 1 0 0
h (made)	= 0 0 1 1 0 0
h (letters)	= 1 0 0 0 0 1

Signature function

Fig. 2.4.1 : Signature file for sample text

- Even if the word is not there then also it is possible to set all the corresponding bits. And this is called as false drop.

- So, while designing signature file, the probability of a false drop should be low enough.
- The hash function is forced to deliver bit masks which have at least *l* bits set. A good model assumes that *l* bits a randomly set in the mask.
- Let $\propto = l \mid B$. As each of the b words sets *l* bits at random. The probability that a given bit of the mask is set i word signature is

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

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

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

- The false drop probability under the optimal selection is

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

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

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

2.4.2 Searching in Signature Files

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

2.4.3 Construction of Signature File

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

- This reduces the disk times to search for a query. Since only the files corresponding to bits which are set in the query have to be traversed.

2.5 Scatter Storage or Hash Addressing

University Question

Q. Explain the concept of hash addressing. **SPPU : May 17, 5 Marks**

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

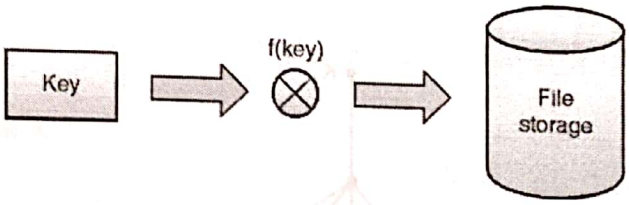


Fig. 2.5.1

2.6 Search Strategies

2.6.1 Introduction

- This section will discuss the process by which the required information is located.
- In the case of document retrieval the information is the subset of documents which are deemed to be relevant to the query.
- The kind of search that is of interest, is not the usual kind where the result of the search is clear cut, either yes, the item is present, or no, the item is absent.
- They are of considerable importance when dictionaries need to be set-up or consulted during text processing. However, we are more interested in search strategies in which the documents retrieved may be more or less relevant to the request.
- All search strategies are based on comparison between the query and the stored documents. Sometimes this comparison is only achieved indirectly when the query is compared with clusters i.e. with the profiles representing the clusters.
- The distinctions made between different kinds of search strategies can sometimes be understood by looking at the query language that is the language in which the information need is expressed. The nature of the query language often dictates the nature of the search strategy.

- For example, a query language which allows search statements to be expressed in terms of logical combinations keywords normally dictates a Boolean search. This is a search which achieves its results by logical (rather numerical) comparisons of the query with the documents.

2.6.2 Boolean Search

- A Boolean search strategy retrieves those documents which are 'true' for the query.
- This formulation only makes sense if the queries are expressed in terms of index terms or keywords and combi by the usual logical connectives AND, OR and NOT.
- For example, if the query Q = ($K1$ AND $K2$) OR ($K3$ AND (NOT $K4$)) then the Boolean search will retriev documents indexed by $K1$ and $K2$, as well as all documents indexed by $K3$which are *not* indexed by $K4$.
- Some systems, which operate by means of Boolean search, allow the user to narrow or broaden the search giving the user access to a structured dictionary which, for any given keyword, stores related keywords which be more general or more precise.
- For example, in the tree structure as shown in Fig. 2.6.1 the keyword K_1^1 is contained in the more ge keyword K_0^1, but it can also be split up into the 4 more precise keywords K_2^1, K_2^2, K_2^3 and K_2^4.
- Therefore, if one has an interactive system the search can easily be reformulated using some of these re terms.

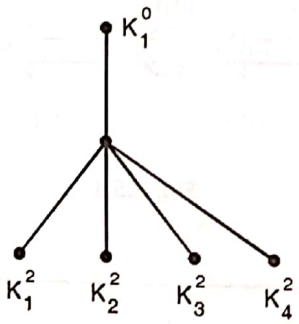


Fig. 2.6.1 : A set of hierarchically related keywords

- Boolean search is implemented using the inverted file.
- We store a list for each keyword in the vocabulary, and in each list put the addresses (or numbers) o documents containing that particular word.
- To satisfy a query we now perform the set operations, corresponding to the logical connectives, on the *Ki*-lists
- For example, if-

  K1 -list : *D1, D2, D3, D4*

  K2 -list : *D1, D2*

  K3 -list : *D1, D2, D3*

  K4 -list : *D1*and *Q* = (*K1* AND *K2*) OR (*K3* AND (NOT *K4*))

  Then to satisfy the (*K1* AND *K2*) part we intersect the *K1* and *K2* lists, to satisfy the (*K3*AND (NOT *K4*)) p subtract the *K4* list from the *K3* list. The OR is satisfied by taking the union of the two sets of documents obt for the parts. The result is the set(*D1, D2 and D3*) which satisfies the query and each document in it is 'true' f query.

- A slight modification of the full Boolean search is one which only allows AND logic but takes into account the actual *number* of terms the query has in common with a document. This number has become known as the *co-ordination level.*
- The search strategy is often called *simple matching*. Because at any level we can have more than one document and the documents are said to be *partially* ranked by the co-ordination levels.
- For the same example as before with the query $Q = K1$ AND $K2$ AND $K3$ we obtain the following ranking :

  Co-ordination level

  3 *D*1, *D*2

  2 *D*3

  1 *D*4

  In fact, simple matching may be viewed as using a primitive matching function. For each document *D* we calculate $|D \cap Q|$, which is the size of the overlap between *D* and *Q*, each represented as a set of keywords.

2.6.3 Serial Search

- Even though the serial searches are slow, they are frequently used as parts of larger systems.
- Serial search provide a convenient demonstration of the use of matching functions.
- Let there are N documents Di in the system, then the serial search proceeds by calculating N values M(Q, Di) i.e. the set of documents to be retrieved is determined.

Two ways to calculate serial search

- There are two ways to calculate this-

  1. The matching function is given a suitable threshold then retrieves the documents above the threshold and discards the documents below threshold.

     If *T* is the threshold, then the retrieved set *B* is the set$\{Di \mid M(Q, Di) > T\}$.

  2. The documents are ranked in increasing order of matching function value.

     A rank position *R* is chosen as cut-off and all documents below the rank are retrieved so that $B = \{Di \mid r(i) < R\}$ where $r(i)$ is the rank position assigned to *Di.*

     So the relevant documents are contained in the retrieved set. The main difficulty with this kind of search strategy is the specification of the threshold or the specification of the cut-off. It will always be arbitrary since there is no way of telling in advance what value for each query will produce the best retrieval.

2.6.4 Cluster-Based Retrieval

- Cluster hypothesis states that the closely associated documents tend to be relevant to the same requests which are foundation for cluster-based retrieval.
- Clustering selects closely associated documents and groups them together into one cluster.
- Suppose we have a hierarchic classification of documents then a simple search strategy goes as shown in Fig. 2.6.2.
- The search starts at the root of the tree, node 0 in the example. It proceeds by evaluating a matching function at the nodes immediately descendant from node 0, in the example the nodes 1 and 2. This pattern repeats itself down the tree.

- The search is directed by a decision rule, which on the basis of comparing the values of a matching fu each stage decides which node to expand further. Also, it is necessary to have a stopping rule which termin search and forces retrieval.
- In Fig. 2.6.2 the decision rule used is- expand the node corresponding to the maximum value of the m function achieved within a filial set.

  The stopping rule used is- stop if the current maximum is less than the previous maximum.
- For this strategy we assume that :
  1. The effective retrieval can be achieved by finding just one cluster.
  2. Each cluster can be adequately represented by a cluster representative for the purpose of locating the containing the relevant documents;
  3. If the maximum of the matching function is not unique some special action, such as a look-ahead, will be taken;
  4. The search always terminates and will retrieve at least one document.

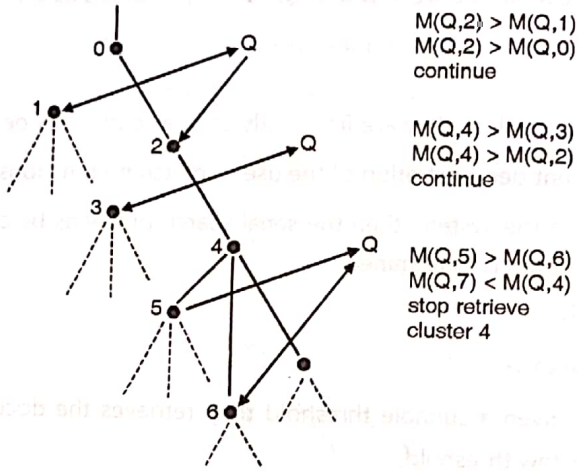


Fig. 2.6.2 : A search tree and the appropriate values of a matching function illustrating the action of a decision rule and a stopping rule

- Generalization of this search is to allow the search to proceed down more than one branch of the tree so allow retrieval of more than one cluster. But the decision rule and stopping rule will be slightly more complica
- The main difference being that provision must be made for *back-tracking*. This will occur when the search st estimates (based on the current value of the matching function) that further progress down a branch is a wa time, at which point it may or may not retrieve the current cluster. The search then returns (back-tracks previous branching point and take an alternative branch down the tree. The above strategy is called a stop searches.
- A *bottom-up* search is one which enters the tree at one of its terminal nodes, and proceeds in an upward dir towards the root of the tree. In this way it will pass through a sequence of nested clusters of increasing s decision rule is not required; we only need a stopping rule which could be simply a cut-off.
- A typical search would seek the largest cluster containing the document represented by the starting node an exceeding the cut-off in size. Once this cluster is found, the set of documents in it is retrieved. To initiate the s in response to a request it is necessary to know in advance one terminal node appropriate for that request. It unusual to find that a user will already known of a document relevant to his request and is seeking documents similar to it. This 'source' document can thus be used to initiate a bottom-up search.

- The search strategy is in part a serial search. It proceeds by first finding the best (or nearest) cluster(s) and then looking within these. The second stage is achieved by doing a serial search of the documents in the selected cluster(s). The output is frequently a ranking of the documents so retrieved.

2.6.5 Matching Function

- Most of the search strategies are implemented by means of a matching function.
- Matching function is a function which is similar to an association measure, but differing in that a matching function measures the association between a query and a document or cluster profile, whereas an association measure is applied to objects of the same king.
- Mathematically the two functions have the same properties and they only differ in their interpretations.
- There are many examples of matching functions; the simplest is the one associated with the simple matching search strategy.

  If M is the matching function, D is the set of keywords representing the document, and Q is the set representing the query, then:

  $M = 2|D \cap Q| / |D| + |Q|$ is another example of a matching function.

  It is of course the same as Dice's coefficient.

- A popular one used by the SMART project, which they call cosine correlation, assumes that the document and query are represented as numerical vectors in t-space, that is $Q = (q_1, q_2, ..., q_t)$ and $D = (d_1, d_2, ..., d_t)$ where qi and di are numerical weights associated with the keyword i.

  The cosine correlation is now simply

$$r = \frac{\sum_{i=1}^{t} q_i d_i}{\left(\sum_{i=1}^{t} (qi)^2 \sum_{i=1}^{t} (q_i)^2 \right)^{\frac{1}{2}}}$$

  Or in the notation for a vector space with a Euclidean norm :

$$r = \frac{(Q, D)}{\| Q \| \| D \|} = \text{cosine } \theta$$

  Where is the angle between vectors Q and D.

2.7 Query languages

Language used for retrieving data by the user from database or any information system is known as query language.

2.7.1 Types of Queries

- Broadly queries are classified in two types
  - Queries for Databases
  - Queries for Information Retrieval
- Languages used for these types are different.

- Also these queries have different purposes and targets.
- For queries on database query languages like Structured Query Language (SQL) are used.
- For queries in Information Retrieval systems languages like Contextual Query Language (CQL) are used.

2.7.2 Pattern Matching

- A pattern is a set of features that must occur in a text segment
- Features can be
  - Words
  - Prefixes/Suffixes
  - Substrings
  - Ranges
  - Regular expressions.
- In pattern matching importance is given to all possible data rather than exact match.
- Patterns represent the context of the user's query.
- This is one of the important searching strategies.
- In information retrieval systems as natural language queries are encouraged, pattern matching becomes esse to find the most relevant result for the user query.
- Popular pattern matching algorithms use edit-distance technique.
- Results of the pattern matching may contain which does not exactly match the query but they match the cont of the pattern. (Ex. Query "mobile prices" includes price for "Samsung Galaxy mobile")

Simple Patterns

- Dinosaur
- "complete dinosaur"
- title = "complete dinosaur"
- title exact "the complete dinosaur"

All through these entire patterns user wants to search for information related to dinosaur. Record needs to ma the context of this pattern to be useful for the user.

2.7.3 Structural Queries

- These queries although close to natural language have a fixed structure in them.
- They have mix of both natural words and structures.
- Ex. : Structure of Email has recipient, subject, contents and attachments. So query can be all emails to recipient.
- There are major three types of structures :

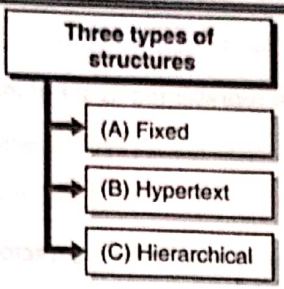


Fig. 2.7.1 : Types of structures

(A) Fixed Structure

- Here the records are of fixed structure.
- All parts of the structure are compulsory in every record.
- Records contain keywords as well as natural language words.
- Each part of the substructure can be searched for.
- Ex. : Structure of Email is FIXED. It has recipient, subject, contents and attachments. So query can be all emails to "xyz" recipient.

(B) Hypertext Structure

- This structure of records is similar to a structure of an HTML document.
- Contents are linked to different documents.
- Links form a graph or tree like structure.
- Links represent connectivity between to nodes/words/documents.
- While searching such structure, traversal takes place by exploring different branches of the graph.
- Ex. : WebGlimpse Application, HTML documents, Web documents, Websites, Web applications.

(C) Hierarchical Structure

- Document structure is arranged in form of a hierarchy.
- In search queries target the hierarchy for getting results.
- Ex. : Under a book there are chapters, under a chapter there are sections.
- Query can be In chapter XYZ in section PQR search the keyword.

2.8 Information Retrieval Models

University Question

Q. Explain various IR models in details with their advantages and disadvantages. **SPPU : May 13, 8 Marks**

The three classic models information retrieval system are called are as shown in Fig. 2.8.1.

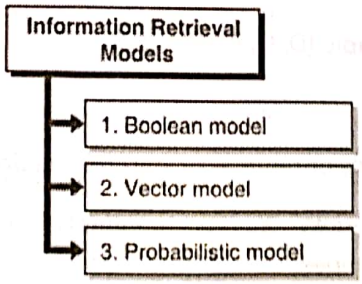


Fig. 2.8.1 : Information retrieval model

1. **Boolean model**
   - In this model document and queries are represented as set of index terms.
   - So the model is set theoretic.
2. **Vector model**
   - In this model document and queries are represented as vectors in a t-dimensional space. So the mo algebraic.
3. **Probabilistic model**
   - In this model document and queries are represented based on probability theory.
   - So the model is probabilistic.
   - Over the years, alternative modelling paradigm for each type of classic models have been proposed.
   - We can distinguish the generalized vector, the latent semantic indexing and the neural network mode alternative to algebraic models.
   - We can distinguish the inference network and the belief network models as alternative to probabilistic mo
   - Fig. 2.8.2 illustratest axonomy of these information retrieval models.
   - So the fundamental premises which the basis for a ranking algorithm determine the IR models.

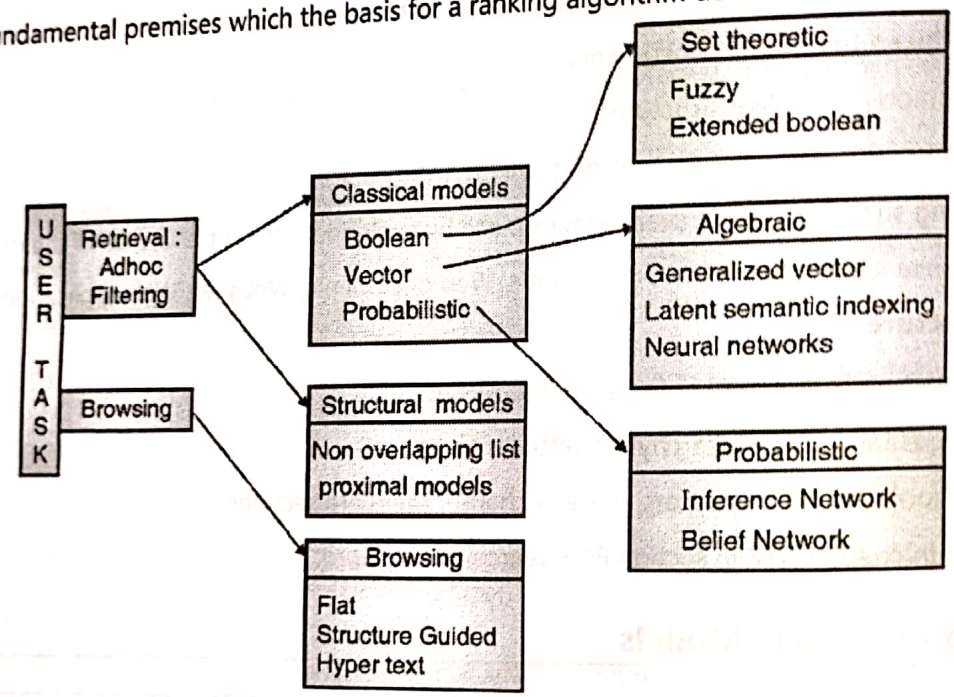


Fig. 2.8.2 : A taxonomy of information retrieval models

- *So the definition of IR model is as follows.*

Definition of information retrieval models

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

Where,

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

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

F → *Framework for modeling document representations, queries and their relationships*

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

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

2.8.1 Basic Concepts

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

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

- g_i be a function that returns the weight associated with the index term K_i in any t-dimensional vector

  i.e. $g_1(\vec{d_j}) = \omega_{i,j}$

2.8.2 Boolean Model

University Questions

Q. Explain Boolean model in detail. **SPPU : Dec. 12, May16, 5/8 Marks**

Q. Describe Boolean model. **SPPU : Dec. 13, 4 Marks**

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

- Boolean model suffer from following drawbacks.
  1. Retrieval strategy is based on a binary decision criterion i.e. a document is predicted to be either relevant non-relevant without any notion of a grading scale which prevents good retrieval performance.
     So in reality Boolean model is much more a data retrieval model.
  2. Every time it is not simple to translate an information need into a Boolean expression as user finds it difficult and awkward to express their query requests in terms of Boolean expressions.
- Even though it has above mentioned drawbacks, it is still the dominant model with commercial document database systems and provides a good starting point for those new to the field.
- Boolean model considers that index terms are present or absent in a document. As a result the index term weights are assumed to be all binary i.e. $\omega_{i,j} \in \{0, 1\}$.
- A query q is composed of index terms linked by three connectives not, and, or.
- Thus, a query is essentially a conventional Boolean expression which can be represented as a disjunction of conjunctive vectors i.e. in disjunctive normal form.
- The query $[q = K_a \wedge (K_b \upsilon \vee K_c)]$ can be written in disjunctive normal form as $[\vec{q}_{dnf} = (1, 1, 1) \vee (1, 1, 0) \vee (1, 0,$ where each of the components is a binary weighted vector associated with the tuple (K_a, K_b, K_c). These binary weighted vectors are called the conjunctive components of $\underrightarrow{q}_{dnf}$.
- Fig. 2.8.3 illustrates the three conjunctive components for the query q.

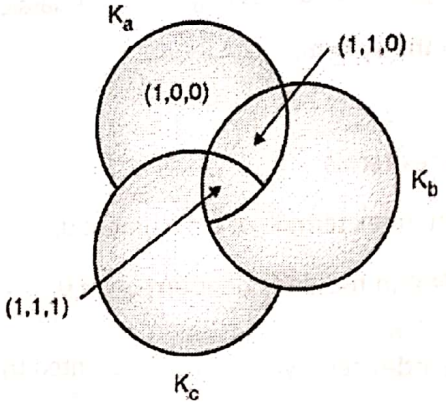


Fig. 2.8.3 : Three conjunctive component for the query $[q = K_a \wedge (K_b \vee \upsilon K_c)]$

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

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

- *For the Boolean model :*
  - *The index term weight variable are all binary i.e. $\omega_{i,j} \in \{0, 1\}$.*
  - *A query q is a conventional Boolean expression.*
  - *$\vec{q}_{dnf}$ be the disjunctive normal form for the query q.*
  - *$\vec{q}_{cc}$ any of the conjunctive components or $\vec{q}_{dnf}$.*
- *If sim $(d_j, q) = 1$ then Boolean model predicts that the document d_j is relevant to the query q otherwise it is not relevant. There is not partial match to the query conditions.*

- If d_j is the document for which $\vec{d_j}$ = (0, 1, 0), so document includes the index HTM K_b but is considered non-relevant to the query [q = $K_a \wedge (K_b \vee \upsilon K_c)$]

Advantages of Boolean model

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

Disadvantage of Boolean model

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

2.8.3 Vector Model

University Questions

Q. Explain vector model in detail. SPPU : May 12, 8 Marks

Q. Describe Vector model. SPPU : Dec. 13, 4 Marks

- The use of binary weights is to limiting in Boolean model so vector model recognized that and proposed a framework in which partial matching is possible.
- This partial matching is accomplished by assigning non-binary weights to index terms in queries and in documents.
- These weights are used to compute the degree of similarity between each document stored in the system and the user query.
- Retrieved documents are sorted in decreasing order by using degree of similarity and then vector model takes into consideration the documents which match the query terms only partially.
- So the ranked document answer set is more precise than the document answer set retrieved by the Boolean model.

The query vector $\vec{q}$ is defined as $\vec{q} = (\omega_{1,q}, \omega_{2,q}, ..., \omega_{t,q})$.

The vector for a document d_j is represented by $d_j = (\omega_{1,j}, \omega_{2,j}, ..., \omega_{t,j})$.

Where, for the vector model

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

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

Index terms in the query are weighted.

A document d_j and a user query q are represented as t-dimensional vectors as shown in Fig. 2.8.4.

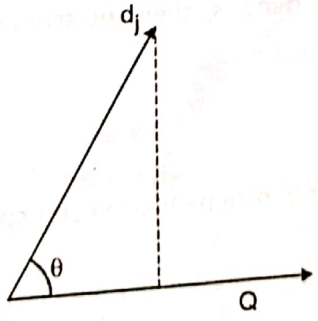


Fig. 2.8.4 : The cosine of θ is adopted as sin (d_j, q)

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

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

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

Where,

$|\vec{d_j}|$ and $|\vec{q}| \rightarrow$ The norms of the document and query vectors.

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

Advantages of vector model

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

Disadvantages of vector model

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

2.8.3(A) Vector Space

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

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

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

x_1 and x_2 are vectors,

nner Product : Inner product or dot product is given by,

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

osine of angle between the vector x_1 and x_2

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

imilarity :

nding similarity between documents ⇒

Documents	Text	Terms
d1	ant ant bee	ant bee
d2	dog bee dog hog dog ant dog	ant bee dog hog
d3	cat gnu dog eel fox	Cat dog eel fox gnu

erm vector space

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

tij = 1 ; if term i is present in doc. j

= 0 ; otherwise

mparing documents (no weighing)

$$\text{Sim}(d_1, d_2) = \frac{d_1 \cdot d_2}{|d_1| \cdot |d_2|}$$

$$= \frac{2}{\sqrt{2} \cdot \sqrt{4}} = \frac{2}{2\sqrt{2}} = \frac{1}{\sqrt{2}} = 0.71$$

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

2. Similarity of query and documents

Term vector space :

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

Similarly of doc. to query :

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

Ex. 2.8.1 : Find the similarity of the following query with documents - D1, D2, D3using vector model.

Query	Keywords	
Q	mouse, dog	
Document	Text	Terms
D1	mouse mouse bee	mouse bee
D2	dog bee dog hog hog dog mouse dog	mouse bee dog hog
D3	cat gnu dog eel fox	Cat dog eel fox gnu

Soln. :

Term vector space :

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

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

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

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

∴ Similarly of docs to query :

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

Ex. 2.8.2 : Find the similarity of following query with D1, D2, D3using vector model.

Query	keywords	
q	ant, dog	
document	Text	Terms
D1	ant ant bee	ant bee
D2	dogbeedoghogdogantdog	ant bee dog hog
D3	cat gnu dog eel fox	catdogeelfoxgnu

SPPU - May 17, 6 Marks

Soln. :

Term vector space :

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

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

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

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

∴ Similarity of documents to query :

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

2.8.3(B) Difference between Boolean and Vector Models

University Question

Q. Compare Boolean and vector model. **SPPU : May 19, 6 M**

Table 2.8.1

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

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

Probabilistic Model

ersity Question

Explain with formulae basic probability model. **SPPU : Dec. 13, 8 Marks**

e probabilistic model tries to fit the IR problem within a probabilistic framework.

e fundamental idea used in this model is as follows :

With given user query, there are set of documents which contain exactly the relevant documents only.

This set of documents is reformed as ideal answer set.

If the description of this ideal answer set is given then we can easily retrieve its documents.

us, we can think of the querying process as a process of specifying the properties of an ideal answer set.

e do not know exactly what these properties are, we know only there are index terms whose semantic should be d to characterize these properties.

properties are not known at query time so effort has to be made at initially guessing these properties.

s initial guess allows up to generate a preliminary probabilistic description of the ideal answer set which is used retrieve a first set of documents.

interaction with the user is then initiated with the purpose of improving the probabilistic description of the al answer set.

h user interaction is performed as follow :

The user takes a look at the retrieved documents and decides which ones are relevant and which are not relevant.

The system then uses this information to define the description of the ideal answer set.

By replacing this process many times such a description will evolve and become closer to the real description of the ideal answer set.

probability model is based on following probabilistic principle :

n a user query q and a document d_j in the collection.

The probabilistic model tries to estimate the probability that the user will find the document d_j relevant.

It assumes that this probability of relevance depends on the query and the document representations.

3. The mode

query q.

- Given a query q, the probabilistic model assigns to each document d_j, as a measure of its similarity to the ratio $p(d_j \text{ relevant to } q) \mid p(d_j \text{ non relevant to } q)$ which computes the odds of the document d_j to the query q.
- The similarity sim (d_j, q) of the document d_j to the query q is defined as -

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

Where,

$\omega_{i,j} \in \{0, 1\}, \omega_{i,q} \in \{0, 1\}$ → Index term weight variables are all binary

R → Set of documents known to be relevant

$\vec{\bar{R}}$ → Complement of R, set of documents non relevant

$P(R \mid \vec{d_j})$ → Probability that document d_j is relevant to the query q

$P(\vec{R} \mid \vec{d_j})$ → Probability that document d_j is non-relevant to the query q

Advantages of Probabilistic Model

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

Disadvantages of Probabilistic Model

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

Review Questions

Q. 1 Explain with example signature file structure.

Q. 2 Compare with example suffix array and suffix tree.

Q. 3 Draw the generalized structure of an inverted file? Explain the algorithm for building an inverted file of a document.

Q. 4 Generate an inverted file for a given text.