Assignment No. 1

Problem Statement:

Implementation of Conflation Algorithm to generate document representative of a text file.

Objective:
To study:

1. The various concepts and components of information retrieval.
2. Conflation Algorithm.
3. The role of clustering in information retrieval.
4. Indexing structures for information retrieval

Outcomes:

At the end of the assignment the students will be able to

1. Understand the concept of Information retrieval and to apply clustering in information retrieval.

Scope:

Removal of Stop Words

Suffix Stripping (Any Five Grammar Rules)

Frequency Occurrences of Key Words (Weight Calculation)

Theory

Introduction to Information Retrieval

In today's information explosion era, increase in demand for quicker dissemination of information, from contents stored in a variety of forms requires speedy search and timely retrieval. The values of documents are measured according to the information it contains but they are proved useless until the stored information is brought out for use by the readers. This may be either by subject analysis or representation of the terms through symbols. It has always been the need of the scholars and the lingering turmoil in the minds of library organizers, to suitably facilitate the extraction of the contents expeditiously and exhaustively that has brought forward the concept of information retrieval.

Meaning & Definition:

Calvin Mooers coined the term information retrieval in 1950. In the context of library and information science, we mean to get back information, which is, in a way, hidden, from normal sight or

vision. According to J.H. Shera: It is, "the process of locating and selecting data, relevant to a given requirement." Calvin Mooers: "Searching and retrieval of information from storage, according to specification by subject**."**

Functions:

The major functions that constitute an information retrieval system, comprises of: Acquisition, Analysis, Representation of information, Organisation of the indexes, Matching, Retrieving, Readjustment and Feedback

Components of Information Retrieval System:

A study of the functions of IRS brings forth some of the essential components that constitute the proper functioning of the system. According to Lancaster, an information retrieval system consists of six basic subsystems. They are as follows:

1. The document selection subsystem
2. The indexing subsystem
3. The vocabulary subsystem
4. The searching subsystem
5. The user-system interface
6. The marching subsystem

All the above subsystems may be grouped under two groups' subject/content analysis and search strategy. Subject or content analysis includes the task of analysis, organisation and storage of information. Search strategy includes analysis of user queries, creation of search formula and the actual searching.

Document Representative:

Documents in a collection are frequently represented through a set of index terms or keywords. Such keywords might be extracted directly from the text of the document or might be specified by a human subject. Modern computers are making it possible to represent a document by its full set of words. With very large collections, however, even modern computers might have to reducethe set of representative keywords. This can be accomplished through the elimination of *stop words* (such as articles and connectives), the use of *stemming* (which reduces distinct words to their common grammatical root), and the identification of noun groups (which eliminates adjectives, adverbs, and verbs). Further, compression might be employed. These operations are called *text operations* (or transformations).

The full text is clearly the most complete logical viewof a document but its usage usually implies higher computational costs. A small set of categories (generated by a human specialist) provides the most concise logical view of a document but its usage might lead to retrieval of poor quality. Several intermediate logical views (of a document) might be adopted by an information retrieval system as illustrated in Figure

Besides adopting any of the intermediate representations, the retrieval system might also recognize the internal structure normally present in a document. This information on the structure of the document might be quite useful and is required by structured text retrieval models. As illustrated in Figure we view the issue of logically representing a document as a continuum in which the logical view of a document might shift (smoothly) from a full text representation to a higher-level representation specified by a human subject.
The document representative is one consisting simply of a list of class names, each name representing a class of words occurring in the total input text. A document will be indexed by a name if one of its significant words occurs as a member of that class.

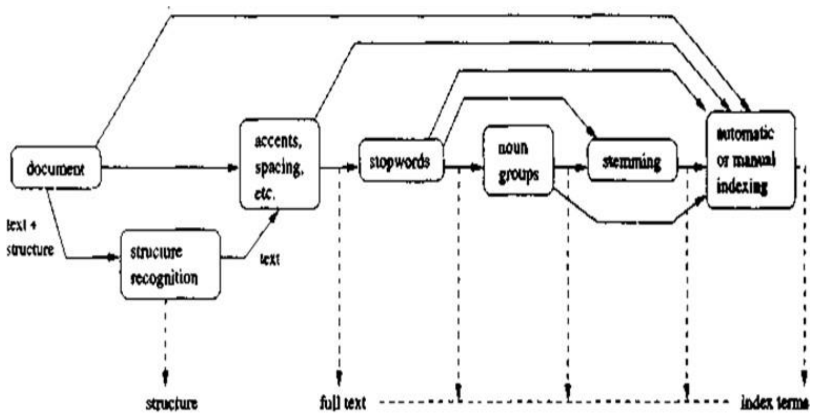


Figure: Logical view of document: full text to set of index terms

Conflation Algorithm:

Ultimately one would like to develop a text processing system which by means of computable methods with the minimum of human intervention will generate from the input text (full text, abstract, or title) a document representative adequate for use in an automatic retrieval system. This is a tall order and can only be partially met. A document will be indexed by a name if one of its *significant* words occurs as a member of that class.

Such a system will usually consist of three parts:

(1) removal of high frequency words,

(2) suffix stripping,

(3) detecting equivalent stems

Luhn's ideas

In one of Luhn's early papers he states: 'It is here proposed that the frequency of word occurrence in an article furnishes a useful measurement of word significance. It is further proposed that the relative position within a sentence of words having given values of significance furnish 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.' This quote fairly summaries Luhn's contribution to automatic text analysis. His assumption is that frequency data can be used to extract words and sentences to represent a document.

The removal of high frequency words, 'stop' words or 'fluff' words is one way of implementing Luhn's upper cut-off. This is normally done by comparing the input text with a 'stop list' of words which are to be removed. The advantages of the process are not only that non-significant words are removed and will therefore not interfere during retrieval, but also that the size of the total document file can be reduced by between 30 and 50 per cent.

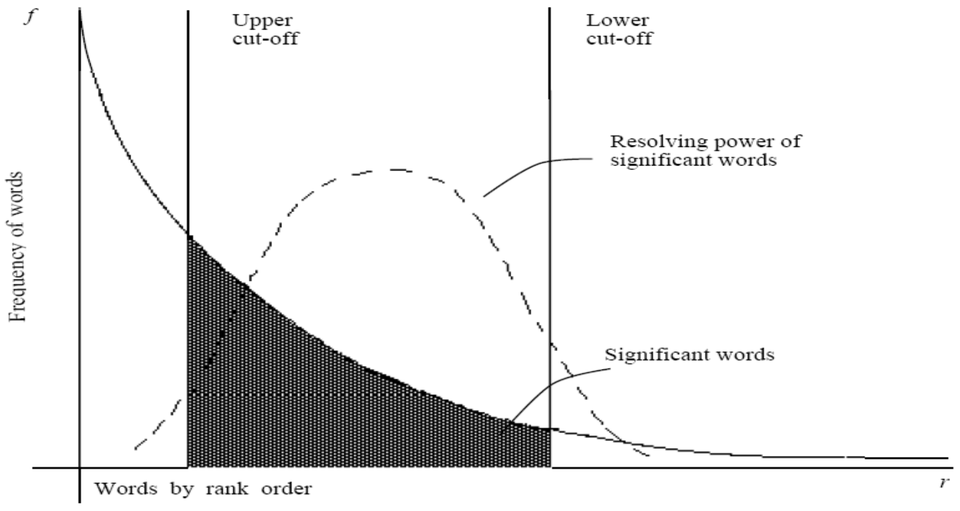


Fig: A plot of Hyperbolic curve relating frequency of words ‘f ‘ vs words by rank order ‘r’

Let f be the frequency of occurrence of various word types in a given position of text and r their rank order, that is, the order of their frequency of occurrence, then a plot relating f and r yields a curve similar to the hyperbolic curve in Figure 2.1. This is in fact a curve demonstrating that the product of the frequency of use of wards and the rank order is approximately constant.

2. Suffix Stripping:

The second stage, suffix stripping, is more complicated. A standard approach is to have a complete list of suffixes and to remove the longest possible one. For example, we may well want UAL removed from FACTUAL but not from EQUAL. To avoid erroneously removing suffixes, context rules are devised so that a suffix will be removed only if the context is right. 'Right' may mean a number of things:

(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.

3. Detecting equivalent stems:

Many words, which are equivalent in the above sense, map to one morphological form by removing their suffixes. The simplest method of dealing with it is to construct a list of equivalent stem-endings. For two stems to be equivalent they must match except for their endings, which themselves must appear in the list as equivalent. For example, stems such as ABSORB- and ABSORPT- are conflated because there is an entry in the list defining B and PT as equivalent stem-endings if the preceding characters match.

The assumption (in the context of IR) is that if two words have the same underlying stem then they refer to the same concept and should be indexed as such. This is obviously a very simplification since words with the same stem, such as NEUTRON AND NEUTRALISE, sometimes need to be distinguished. Even words which are essentially equivalent may mean different things in different contexts. Since there is no cheap way of making these finedistinctions we put up with a certain proportion of errors and assume (correctly) that they will not degrade retrieval effectiveness too much. The final output from a conflation algorithm is a set of classes, one for each stem detected. A 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 often referred to as the documents *index terms* or *keywords.*

Input:

1. A text file containing stop words
2. A document which is searched and index according to frequency of words

Output:

Document containing frequently appearing words without stop words and removing stemming

Program implementation: Code written in C/C++ to implement Conflation Algorithm with proper output.

Conclusion: Thus, we have implemented the Conflation Algorithm to generate document representative of a text file.

Assignment No. 2

Problem Statement:

Implement Single-pass Algorithm for clustering of files.

Objectives:
To study :

1. What is Clustering?

2. Single pass algorithm for clustering.

3. Measure of association

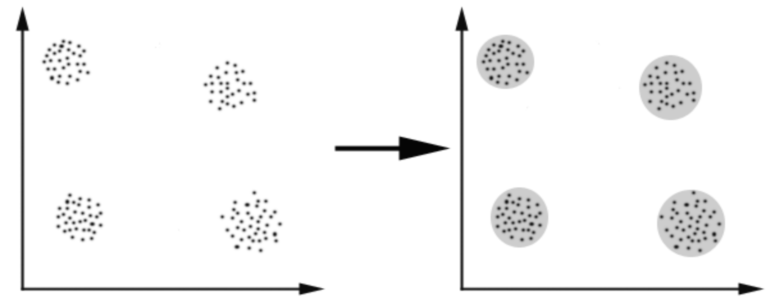
4. The graphical representation of clustering.

Theory:
Clustering
Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabeled data.

A definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A *cluster* is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters.

Clustering is the process of grouping the documents which are relevant. It can be shown by a graph with nodes connected if they are relevant to the same request. A basic assumption is that documents relevant to a request are separated from those which are not relevant i.e. the relevant documents are more like one another than they are like non relevant one.

A simple graphical example:



To identify the 4 clusters into which the data can be divided; the similarity criterion is *distance*: two or more objects belong to the same cluster if they are "close" according to a given distance (in this case geometrical distance). This is called *distance-based clustering*.

Another kind of clustering is *conceptual clustering*: two or more objects belong to the same cluster if this one defines a concept *common* to all that objects. In other words, objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

The Goals of Clustering

The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best criterion which would be independent of the final aim of the clustering. Consequently, it is the user which must supply this criterion, in such a way that the result of the clustering will suit their needs. For instance, user could be interested in finding representatives for homogeneous groups (*data reduction*), in finding "natural clusters" and describe their unknown properties (*"natural" data types*), in finding useful and suitable groupings (*"useful" data classes*) or in finding unusual data objects (*outlier detection*).

Clustering Requirements

The main requirements that a clustering algorithm should satisfy are:

1. Scalability;
2. Dealing with different types of attributes;
3. Discovering clusters with arbitrary shape;
4. Minimal requirements for domain knowledge to determine input parameters;
5. Ability to deal with noise and outliers;
6. Insensitivity to order of input records;
7. High dimensionality;
8. Interpretability and usability.

Single Pass Clustering

Single Pass clustering quickly by which we make incremental clustering to stream data. This clustering technique provides us with a simple yet flexible technique for stream data1. Given a collection of clusters and a threshold value h, if a new document n has the highest similarity more than h to some cluster, the document n is appended to the cluster, and if there exists no cluster, a new cluster is generated which contains only the document n. Clearly Single Pass Clustering is suitable for incremental clustering to temporal data (or data stream) since, once a document is assigned to a cluster, it is not changed in the future.

The algorithm is as follows

(1) Let h be a threshold value.

(2) Let S be an empty set and d1 be the first document. We generate a new cluster C1 consisting of d1.

(3) When a new document di(i > 1) comes in, calculate the similarity values to all the clusters C.

(4) Let simmax be the highest value and Cdi the most similar cluster. If simmax > h, add di to Cdi and adjust the center of Cdi . Otherwise, we generate a new cluster Cdi that contains only di.

(5) Repeat the process above until no data comes. In (4) we define simmax = MAX(sim(~di, ~C)). Also we define similarity of a document d and a cluster C where the center is VC as below (called cosine similarity):

sim(~d, ~C) = d~ · V~C / |~d| | V~C|

Measures of association

Association is the similarity between objects characterized by discrete state attributes. The measure of similarity or association is designed to quantify likeness between the objects in such a way that an object in a group is more like the other members of the group that is like any object outside the group then a cluster method enables such a group structure to be discovered.

There are five commonly used measures of association in IR.

1. | X ∩ Y |	Simple matching coefficient
2. | X ∩ Y | / | X | + | Y |	Dices coefficient
3. | X ∩ Y | / | X U Y |	Jaccard’s coefficient

4. $| X \cap Y | / | X |^{1/2} * | Y |^{½}$ Cosine coefficient
5. $| X \cap Y | / \min (| X |, | Y |)$ Overlap coefficient

In short, measure of association is calculated by this program by taking into account frequency of occurrence of words in both the documents i.e least value of frequency of occurrence of a common word in both documents is considered for finding out the measure of association.

Classification methods

1. Multi state attribute (E.g. : Colour)
2. Binary state (E.g. :Keyword)
3. Numerical (E.g.: Hardness scale or weighted keyword)
4. Probability distribution

Cluster hypothesis

The hypothesis can be simply stated as closely associated document tend to be relevant to the same request. This hypothesis is referred as Cluster hypothesis.

The basic assumption in retrieval system is that documents relevant to a request are separated from those which have not relevant. Compute the association between all pairs of documents.

a. Both of which are relevant to a request and
b.One of which is relevant and the other is not

Two approaches of clustering

1.The clustering is based on a measure of similarities between the objects to be clustered. The example of first approach is graph theoretic method which can be used to define clusters in terms of graphs derived from measure of similarity.
2.The cluster method proceeds directly from the object description.

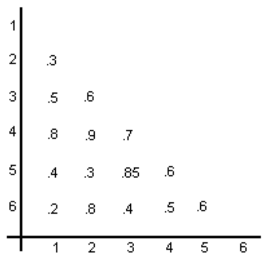
Graphical representation of clustering

Here similarity matrix is used in order to draw the graph, the documents having measure of association greater than threshold value can be represented by the edge in the graph. This is an identical cluster which can shown as connected graph.

From the graph below , it can be easily understood that all documents are associated. But documents like 2 & 5 are not directly associated & same is the case for documents 4 & 5.In this way clusters can be depicted

Example:

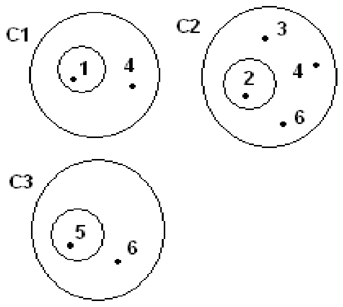
Objects {1, 2, 3, 4, 5, 6}



	1	2	3	4	5	6
1						
2	.3					
3	.5	.6				
4	.8	.9	.7			
5	.4	.3	.85	.6		
6	.2	.8	.4	.5	.6	

Threshold: 0.59

Clusters are:



Input: Document representative (minimum 5 files)

Scope: Use of matching function (dice coefficient) for comparing document representative. define appropriate threshold value.

Program Implementation: Code written in c/c++ to implement single pass algorithm for clustering with proper output.

Output: Cluster of documents

<u>**Conclusion:**</u> Thus, we have implemented the single pass algorithm for clustering.

Assignment No. 3

Problem Statement:

To implement a program for Retrieval of documents using inverted files

Objectives:

To study Indexing, Inverted Files and searching information with the help of inverted file.

Outcomes:

At the end of the assignment the students should have:

1. Understood use of indexing in fast retrieval
2. Understood working of inverted index

Infrastructure: Desktop/ laptop system with Linux or its derivatives.

Software used: LINUX/ Windows OS/ Virtual Machine/ IOS/C/C++/Java/python

Theory:

Indexing

In searching for a basics query is to scan the text sequentially. Sequential or online text searching involves finding the occurrences of a pattern in a text. Online searching is appropriate then the text is small and it is the only choice if the text collection is very volatile or the index space overhead cannot be afforded. A second option is to build data structures over the text to speed up the search. It is worthwhile building and maintaining an. index when the text collection is large and semi-static. Semi-static collections can be updated at reasonably regular intervals but they are not deemed to support thousands of insertions of single words per second. This is the case for most real text databases not only dictionaries or other slow growing literary works. There are many indexing Techniques.

Three of them are inverted files, suffix arrays and signature files.

Inverted Files:

An inverted file is a word-oriented mechanism for indexing a test collection in order to speed up the matching task. The inverted file structure is composed of two elements: vocabulary and occurrence. The vocabulary is the set of all different words in the text. For each such word a list of all the text portions where the word appears is stored. The set of all those lists is called the occurrences. These positions can refer to words or characters. Word positions simplify phrase and proximity queries, while character positions facilitate direct access to the matching text position.

Searching with the help of inverted file:

The search algorithm on an inverted index has three steps.

1. Vocabulary Search

Searching with the help of inverted file:

The search algorithm on an inverted index has three steps.

1. Vocabulary Search
2. Retrieval of occurrence
3. Manipulation of occurrences

Single-word queries can be searched using any suitable data structure to speed up the search, such as hashing, tries, or B-trees. The first two give O(m) search cost. However, simply storing the words in lexicographically order is cheaper in space and very competitive in performance. Since the word can be binary searched at O (log n) cost. Prefix and range queries can also be solved with binary search, tries, or B-trees but not with hashing. If the query is formed by single words then the process ends by delivering the list of occurrences. Context queries arc more difficult to solve with inverted indices. Each element must be searched separately and a list generated for each one. Then, the lists of all elements are traversed in synchronization to *find* places where all the words appear in sequence (for a phrase) or appear close enough (for proximity). If one list is much shorter than the others, it may be better to binary search its elements into the longer lists instead of performing a linear merge. If block addressing is used itis necessary to traverse the blocks for these queries, since the position information is needed. It isthen better to intersect the lists to obtain the blocks which contain all the searched words andthen sequentially search the context query in those blocks. Some care has to be exercised at block boundaries, since they can split a match. Example:

Text:

1 6 9 11 17 19 ….
This is a text. A text has many words. Words are made from letters.

Inverted Index:

Vocabulary	Occurrences
Letters	60…
Made	50…
Many	28…
Text	11,19…
Words	33,40….

Algorithm

1. Input the conflated file
2. Build the index file for input file
3. Input the query
4. Print the index file and result of query

Conclusion: Implementation is concluded by stating analysis of Retrieval of documents using Inverted Files.

Assignment No. 4

Problem Statement:

Implement a program to calculate precision and recall for sample input.
(Answer set A, Query q1, Relevant documents to query q1- Rq1)

Objectives:

1. To understand precision and recall in information retrieval
2. To study indexing structures for information retrieval.

Outcomes:

At the end of the assignment the students should have:

1. Understood precision and recall in information retrieval.
2. Understood use of indexing in fast retrieval.

Theory:

Precision and Recall in Information Retrieval

Information Systems can be measured with two metrics: precision and recall. When a user decides to search for information on a topic, the total database and the results to be obtained can be divided into 4 categories:

1. Relevant and Retrieved
2. Relevant and Not Retrieved
3. Non-Relevant and Retrieved
4. Non-Relevant and Not Retrieved

Relevant items are those documents that help the user in answering his question. Non-Relevant items are items that don't provide actually useful information. For each item there are two possibilities it can be retrieved or not retrieved by the user's query. Precision is defined as the ratio of the number of relevant and retrieved documents(number of items retrieved that are actually useful to the user and match his search need) to the number of total retrieved documents from the query. Recall is defined as ratio of the number of retrieved and relevant documents(the number of items retrieved that are relevant to the user and match his needs) to the number of possible relevant documents(number of relevant documents in the database).Precision measures one aspect of information retrieval overhead for a user associated with a particular search. If a search has 85 percent precision then 15(100-85) percent of user effort is overhead reviewing non-relevant items. Recall measures to what extent a system processing a particular query is able to retrieve the relevant items the user is interested in seeing. Recall is a very useful concept but due to the denominator is non-calculable in operational systems. If the system is made known the total set of relevant items in the database, recall can be made calculable.

- Precision (P) is the fraction of retrieved documents that are relevant

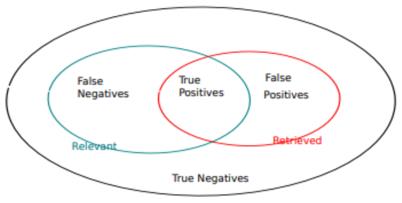
$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved})$$

- Recall (R) is the fraction of relevant documents that are retrieved

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant})$$

THE TRUTH

WHAT THE SYSTEM THINKS		Relevant	Nonrelevant
	Retrieved	true positives (TP)	false positives (FP)
	Not retrieved	false negatives (FN)	true negatives (TN)



$$\begin{aligned} P &= TP/(TP + FP) \\ R &= TP/(TP + FN) \end{aligned}$$

Precision/recall trade-off

You can increase recall by returning more docs. Recall is a non-decreasing function of the number of docs retrieved. A system that returns all docs has 100% recall! The converse is also true (usually): It’s easy to get high precision for very low recall.

Consider an Information retrieval (IR) system returning relevant documents

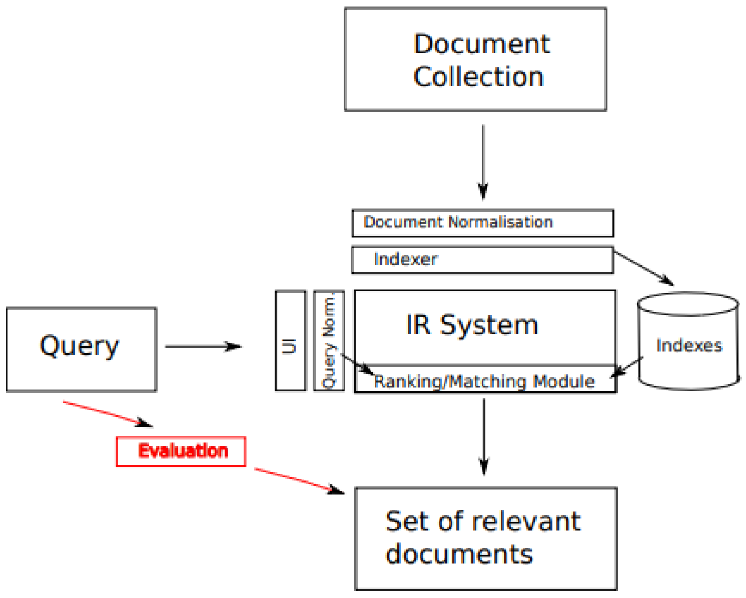
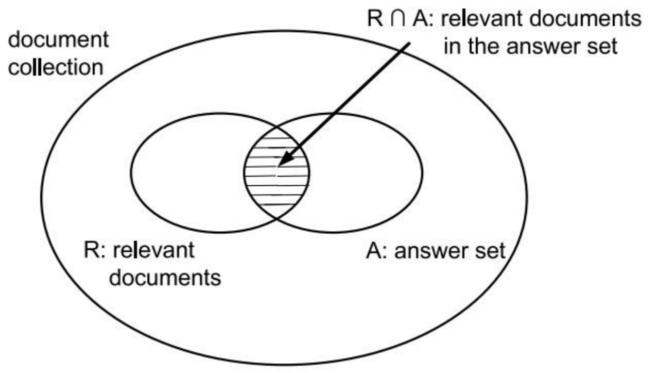


Fig 1: IR system returning relevant documents

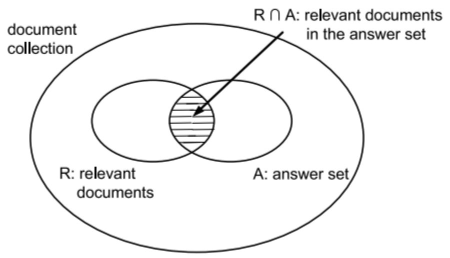
Precision and Recall explanation:
Consider,
I: an information request
R: the set of relevant documents for I
A: the answer set for I, generated by an IR system
R ∩ A: the intersection of the sets R and A
|A|-number of documents in the set A
|Ra |-number of documents in the intersection of sets R and A



- **Recall** is the fraction of the relevant documents (the set R) which has been retrieved i.e.,

$$Recall = \frac{|R \cap A|}{|R|}$$

- **Precision** is the fraction of the retrieved documents (the set A) which is relevant i.e.,



$$Precision = \frac{|R \cap A|}{|A|}$$

Recall

$$R = \frac{\text{Number of relevant items retrieved}}{\text{Total number of relevant items in collection}}$$

Precision

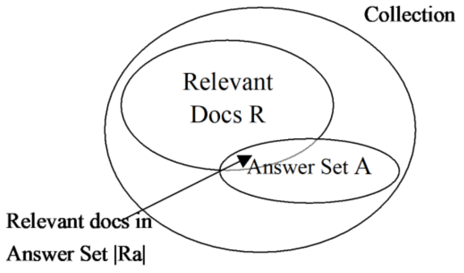
$$P = \frac{\text{Number of relevant items retrieved}}{\text{Total number of items retrieved}}$$

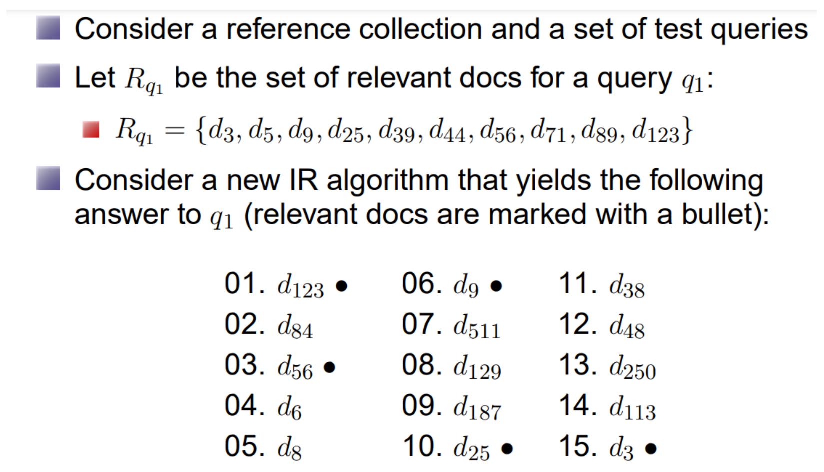
The goal is to achieve **high precision** and **high recall**. The definition of precision and recall assumes that all docs in the set A have been examined However, the user is not usually presented with all docs in the answer set A at once User sees a ranked set of documents and examines them starting from the top Thus, precision and recall vary as the user proceeds with their examination of the set A. Most appropriate then is to plot a curve of precision versus recall.

I	Information request
R	set of relevant documents
|R|	number of documents in set R
A	Document answer set
|A|	Number of documents in the set A
|R_a |	Number of documents in the intersection of sets R and A

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

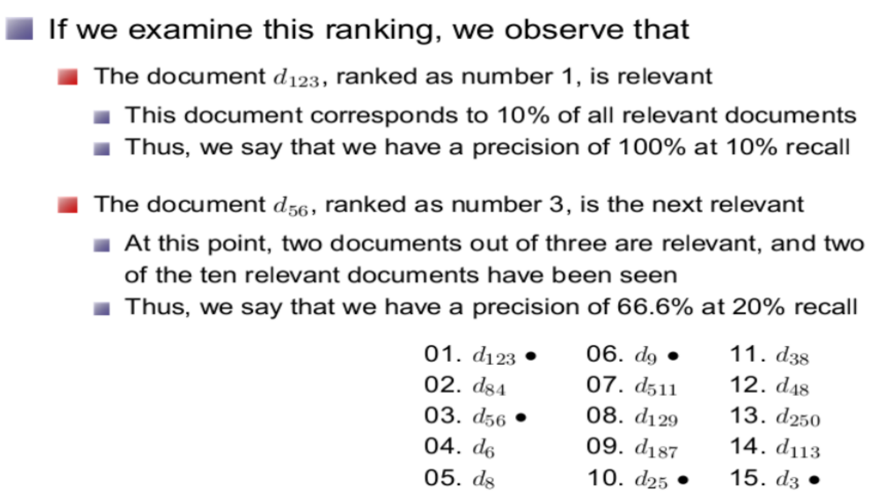
$$\text{Pr}ecision = \frac{|Ra|}{|A|}$$





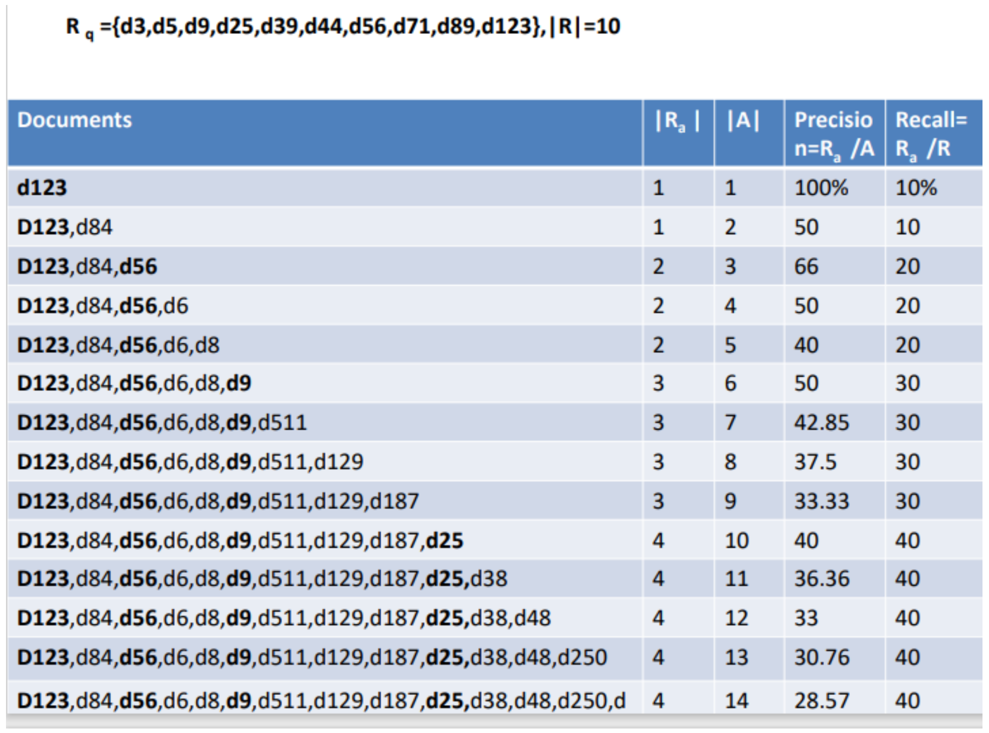
- Consider a reference collection and a set of test queries
- Let R_{q_1} be the set of relevant docs for a query q_1:
  - $R_{q_1} = \{d_3, d_5, d_9, d_{25}, d_{39}, d_{44}, d_{56}, d_{71}, d_{89}, d_{123}\}$
- Consider a new IR algorithm that yields the following answer to q_1 (relevant docs are marked with a bullet):

01. d_{123} ●	06. d_9 ●	11. d_{38}
02. d_{84}	07. d_{511}	12. d_{48}
03. d_{56} ●	08. d_{129}	13. d_{250}
04. d_6	09. d_{187}	14. d_{113}
05. d_8	10. d_{25} ●	15. d_3 ●



- If we examine this ranking, we observe that
  - The document d_{123}, ranked as number 1, is relevant
    - This document corresponds to 10% of all relevant documents
    - Thus, we say that we have a precision of 100% at 10% recall
  - The document d_{56}, ranked as number 3, is the next relevant
    - At this point, two documents out of three are relevant, and two of the ten relevant documents have been seen
    - Thus, we say that we have a precision of 66.6% at 20% recall

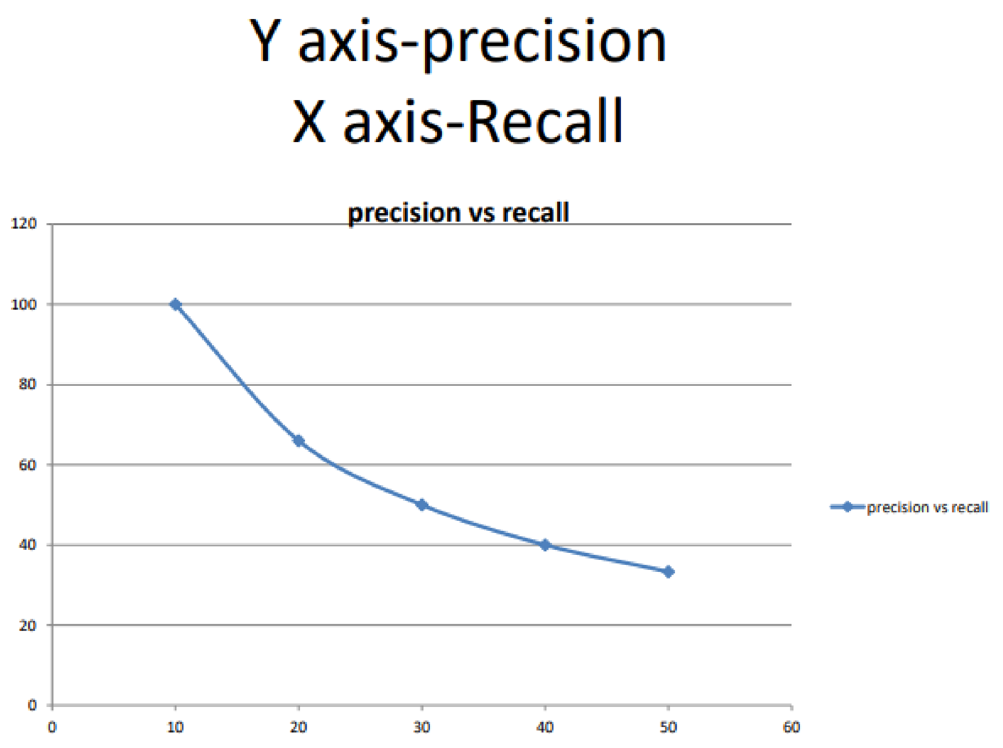
01. d_{123} ●	06. d_9 ●	11. d_{38}
02. d_{84}	07. d_{511}	12. d_{48}
03. d_{56} ●	08. d_{129}	13. d_{250}
04. d_6	09. d_{187}	14. d_{113}
05. d_8	10. d_{25} ●	15. d_3 ●



R_q ={d3,d5,d9,d25,d39,d44,d56,d71,d89,d123},|R|=10

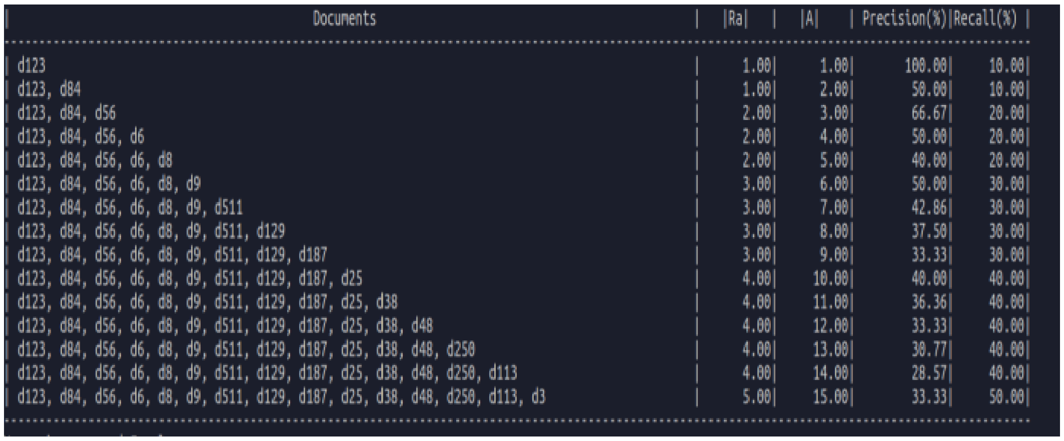
Documents	$\|R_a\|$	|A|	Precision=R_a /A	Recall= R_a /R
d123	1	1	100%	10%
D123,d84	1	2	50	10
D123,d84,**d56**	2	3	66	20
D123,d84,**d56**,d6	2	4	50	20
D123,d84,**d56**,d6,d8	2	5	40	20
D123,d84,**d56**,d6,d8,**d9**	3	6	50	30
D123,d84,**d56**,d6,d8,**d9**,d511	3	7	42.85	30
D123,d84,**d56**,d6,d8,**d9**,d511,d129	3	8	37.5	30
D123,d84,**d56**,d6,d8,**d9**,d511,d129,d187	3	9	33.33	30
D123,d84,**d56**,d6,d8,**d9**,d511,d129,d187,**d25**	4	10	40	40
D123,d84,**d56**,d6,d8,**d9**,d511,d129,d187,**d25**,d38	4	11	36.36	40
D123,d84,**d56**,d6,d8,**d9**,d511,d129,d187,**d25**,d38,d48	4	12	33	40
D123,d84,**d56**,d6,d8,**d9**,d511,d129,d187,**d25**,d38,d48,d250	4	13	30.76	40
D123,d84,**d56**,d6,d8,**d9**,d511,d129,d187,**d25**,d38,d48,d250,d	4	14	28.57	40

If we proceed with our examination of the ranking generated, we can plot a curve of precision versus recall as follows:



Thus, Precision and recall have been extensively used to evaluate the retrieval performance of IR systems or algorithms. However, a more careful reflection reveals problems with these two measures: First, the proper estimation of maximum recall for a query requires detailed knowledge of all the documents in the collection Second, in many situations the use of a single measure could b e more appropriate Third, recall and precision measure the effectiveness over a set of queries processed in batch mode Fourth, for systems which require a weak ordering though, recall and precision might be inadequate.

Sample code in C++
• Code
• Output



```
|                              Documents                              |   |Ra|   |   |A|   | Precision(%)|Recall(%) |
---------------------------------------------------------------------------------------------------------------------
| d123                                                                |     1.00|     1.00|      100.00|    10.00|
| d123, d84                                                           |     1.00|     2.00|       50.00|    10.00|
| d123, d84, d56                                                      |     2.00|     3.00|       66.67|    20.00|
| d123, d84, d56, d6                                                  |     2.00|     4.00|       50.00|    20.00|
| d123, d84, d56, d6, d8                                              |     2.00|     5.00|       40.00|    20.00|
| d123, d84, d56, d6, d8, d9                                          |     3.00|     6.00|       50.00|    30.00|
| d123, d84, d56, d6, d8, d9, d511                                    |     3.00|     7.00|       42.86|    30.00|
| d123, d84, d56, d6, d8, d9, d511, d129                              |     3.00|     8.00|       37.50|    30.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187                        |     3.00|     9.00|       33.33|    30.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25                   |     4.00|    10.00|       40.00|    40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38              |     4.00|    11.00|       36.36|    40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48         |     4.00|    12.00|       33.33|    40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48, d250   |     4.00|    13.00|       30.77|    40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48, d250, d113 |  4.00|    14.00|       28.57|    40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48, d250, d113, d3 | 5.00|  15.00|       33.33|    50.00|
---------------------------------------------------------------------------------------------------------------------
```

Conclusion: Implementation is concluded by executing a program to calculate precision and recall for sample input with relevant documents Rq1 for query q1.

Assignment No. 5

Problem Statement:

Write a program to calculate harmonic mean (F-measure) and E-measure for above example

Objectives:

1. To evaluate the retrieval performance of IR systems
2. To understand importance of harmonic mean (F-measure) and E-measure in information retrieval
3. To study indexing structures for information retrieval.

Outcomes:

At the end of the assignment the students should have:

1. Understood to calculate harmonic mean (F-measure) and E-measure in information retrieval.
2. Understood method to evaluate the retrieval performance of IR systems.

THEORY:

F-Score / F-Measure)

F1 Score considers both precision and recall.

It is the harmonic mean(average) of the precision and recall.

F1 Score is best if there is some sort of balance between precision (p) & recall (r) in the system.

Oppositely F1 Score isn't so high if one measure is improved at the expense of the other.

For example, if P is 1 & R is 0, F1 score is 0.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Information Systems can be measured with two metrics: precision and recall. Thus, Precision and recall have been extensively used to evaluate the retrieval performance of IR systems or algorithms. However, a more careful reflection reveals problems with these two measures: First, the proper estimation of maximum recall for a query requires detailed knowledge of all the documents in the collection Second, in many situations the use of a single measure could be more appropriate Third, recall and precision measure the effectiveness over a set of queries processed in batch mode Fourth, for systems which require a weak ordering though, recall and precision might be inadequate.

Alternative measures: The harmonic mean/F measure

The F-measure is also a single measure that combines recall and precision

$$F(j) = \frac{2}{\frac{1}{r(j)} + \frac{1}{P(j)}}$$

Where,
r (j) is the recall at the j-th position in the ranking
P (j) is the precision at the j-th position in the ranking
F (j) is the harmonic mean at the j-th position in the ranking

Determining max value of F can be interpreted as an attempt to find the best possible compromise between recall and precision. The function F assumes values in the interval [0, 1] It is 0 when no relevant documents have been retrieved and is 1 when all ranked documents are relevant Further, the harmonic mean F assumes a high value only when both recall and precision are high To maximize F requires finding the best possible compromise between recall and precision.

Alternative measures: E measure

E measure was proposed by Van Rijesbergn which combines recall and precision. User is allowed to specify whether he is more interested in recall or in precision.
E measure is defined as

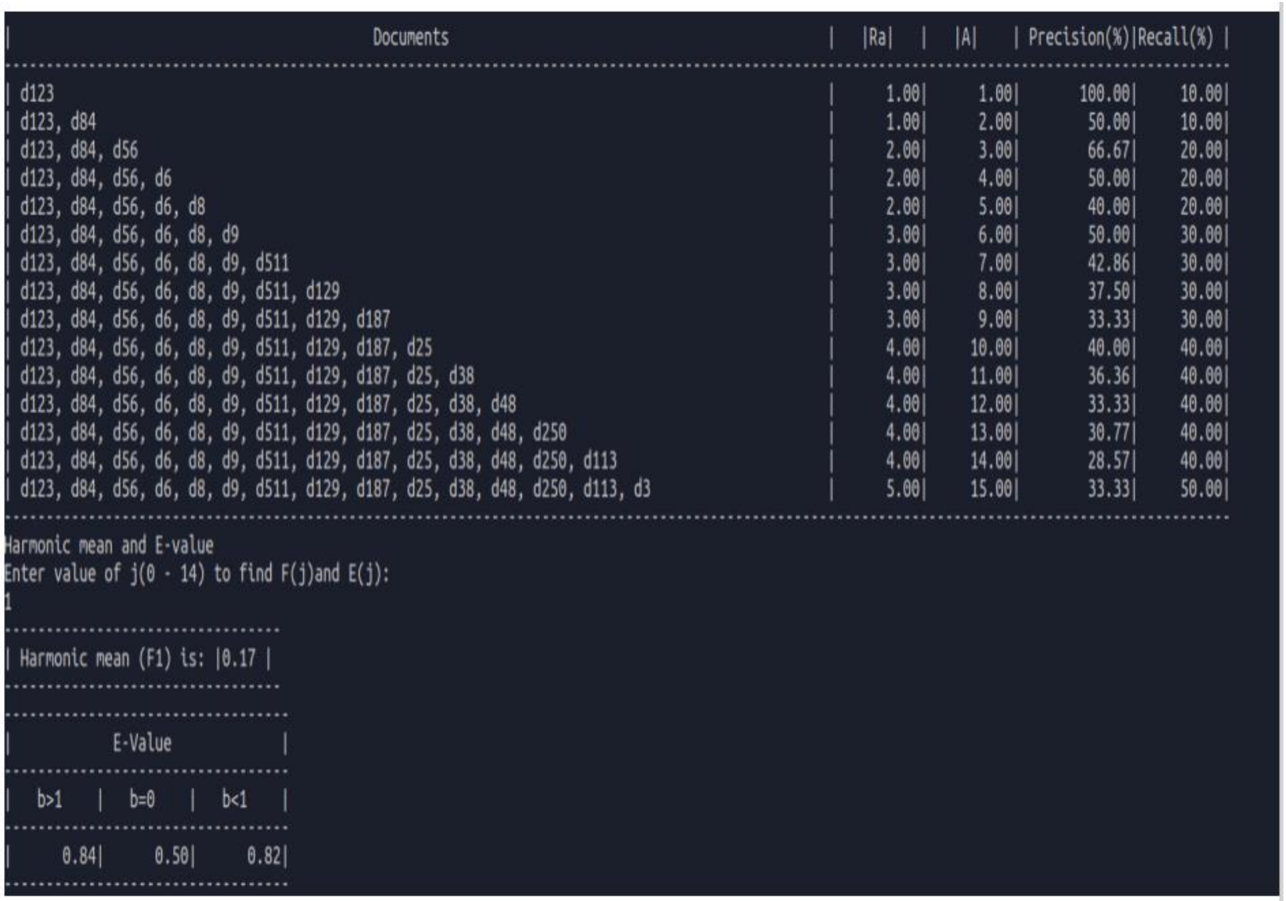
$$E(j) = 1 - \frac{1 + b^2}{\frac{b^2}{r(j)} + \frac{1}{P(j)}}$$

Where,
r (j) is the recall at the j-th position in the ranking
P (j) is the precision at the j-th position in the ranking
b ≥ 0 is a user specified parameter
E (j) is the E metric at the j-th position in the ranking
If b=1 E(j) measure works as complement of the Harmonic mean F(j)
If b>1 indicates that the user is more interested in precision than in recall
If b<1 Indicates that user is more interested in recall than in precision
Notice that setting b = 1 in the formula of the E-measure yields F (j) = 1 − E (j)

The parameter b is specified by the user and reflects the relative importance of recall and

Precision.
If b = 0 E (j) = 1 − P (j) low values of b make E (j) a function of precision.
If b → ∞ lim b→∞ E (j) = 1 − r (j) high values of b make E (j) a function of recall
For b = 1, the E-measure becomes the F-measure.
Thus. Single value measures can also be stored in a table to provide a statistical summary For instance, these summary table statistics could include the number of queries used in the task the total number of documents retrieved by all queries the total number of relevant docs retrieved by all queries the total number of relevant docs for all queries, as judged by the specialists.

Sample code in C++
• Code
• Output



```
|                           Documents                                    |   |Ra|   |   |A|    | Precision(%)|Recall(%) |
-------------------------------------------------------------------------------------------------------------------------
| d123                                                                   |     1.00|     1.00|       100.00|     10.00|
| d123, d84                                                              |     1.00|     2.00|        50.00|     10.00|
| d123, d84, d56                                                         |     2.00|     3.00|        66.67|     20.00|
| d123, d84, d56, d6                                                     |     2.00|     4.00|        50.00|     20.00|
| d123, d84, d56, d6, d8                                                 |     2.00|     5.00|        40.00|     20.00|
| d123, d84, d56, d6, d8, d9                                             |     3.00|     6.00|        50.00|     30.00|
| d123, d84, d56, d6, d8, d9, d511                                       |     3.00|     7.00|        42.86|     30.00|
| d123, d84, d56, d6, d8, d9, d511, d129                                 |     3.00|     8.00|        37.50|     30.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187                           |     3.00|     9.00|        33.33|     30.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25                      |     4.00|    10.00|        40.00|     40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38                 |     4.00|    11.00|        36.36|     40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48            |     4.00|    12.00|        33.33|     40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48, d250      |     4.00|    13.00|        30.77|     40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48, d250, d113 |    4.00|    14.00|        28.57|     40.00|
| d123, d84, d56, d6, d8, d9, d511, d129, d187, d25, d38, d48, d250, d113, d3 | 5.00|    15.00|        33.33|     50.00|
-------------------------------------------------------------------------------------------------------------------------
Harmonic mean and E-value
Enter value of j(0 - 14) to find F(j)and E(j):
1
-----------------------------
| Harmonic mean (F1) is: |0.17 |
-----------------------------
------------------------------
|          E-Value           |
------------------------------
|  b>1   |  b=0    |  b<1    |
------------------------------
|     0.84|     0.50|     0.82|
------------------------------
```

Conclusion: Implementation is concluded by executing a program to calculate (F-measure) and E-measure for sample input used in above example.

Assignment No. 6

Problem Statement:

To Implement a program for feature extraction in 2D color images (any features like color, texture etc. and to extract features from input image and plot histogram for the features.

Objective:

1. To study Program for Feature Extraction in 2D Colour Images for features like,Colour and Textures and plot Histogram.
2. Given Feature Extraction source code is implemented using Java or python language .
3. The input to the program is image file that is to be modified using program by changing colour

Outcomes:

At the end of the assignment the students should have

1. Understood the feature extraction process and its applications
2. Apply appropriate tools in analyzing the web information.

Infrastructure: Desktop/ laptop system with Linux or its derivatives.

Software used: LINUX/ Windows OS/ Virtual Machine/ IOS, python 3.9.12

Input: Image file

Output: Features of Image file

Theory:

Introduction:

Technology determines the types and amounts of information we can access. Currently, a large fraction of information originates in silicon. Cheap, fast chips and smart algorithms are helping digital data processing take over all sorts of information processing Consequently, the volume of digital data surrounding us increases continuously. However, an information-centric society has additional requirements besides the availability and capability to process digital data. We should also be able to find the pieces of information relevant to a particular problem. Having the answer to a question but not being able to find it is equivalent to not having it at all. The increased volume of information and the wide variety of data types make finding information a challenging task. Current searching methods and algorithms are based on assumptions about technology and goals that seemed reasonable before the widespread use of computers.

However, these assumptions no longer hold in the context of information retrieval systems. The pattern originated in the information retrieval domain. However, information retrieval has expanded into other fields like office automation, genome databases, fingerprint identification, medical imaging, data mining, multimedia, etc. Since the pattern works with any kind of data, it is applicable in many other domains. You will see examples from text searching, telecommunications, stock prices, medical imaging and trademark symbols. The key idea of the pattern is to map from a large, complex problem space into a small, simple feature space. The mapping represents the creative part of the solution. Every type of application uses a different kind mapping. Mapping into the feature space is also the hard part of this pattern. Traditional searching algorithms are not viable for problems typical to the information retrieval domain. Since they were designed for exact matching, their use for similarity search is cumbersome. In contrast, feature extraction provides an elegant and efficient alternative. With information retrieval expanding into other fields, this pattern is applicable in a wide range of applications. Work with an alternative, simpler representation of data. The representation contains some information that is unique to each data item. This computation is actually a function. It maps from the problem space into a feature space. For this reason, it is also called feature extraction process.

Feature Extraction:

Texture is an important feature that identifies the object present in any image. The texture is defined by the spatial distribution of pixels in the neighborhood of an image. The gray level spatial dependency is represented by a two-dimensional matrix known as GLCM and it is used for texture analysis. The GLCM matrix specifies that how often the pairs of pixels with certain values occur in an image. The statistical measures are then derived using the GLCM matrix. The textural features represent the spatial distribution of gray tonal variations within a specified area. In images, the neighboring pixel is correlated and spatial values are obtained by the redundancy between the neighboring pixel values. The color features are represented by color histograms in six color spaces namely RGB, HSV, LAB, CIE, HUE and OPP.

The textural features are considered for classifying the image. These textural features are calculated in the spatial domain and a set of gray tone spatial dependency matrix was computed. The textural features are computed using GLCM matrix in four different orientation angles. The textural features are based on the fact that describes how the gray tone appears in a spatial relationship to another.

GRAY LEVEL CO-OCCURENCE MATRIX (GLCM)

In statistical texture analysis, from the distribution of intensities the texture features are obtained at specified position relative to one another in an image. The statistics of texture are classified into first order, second order and higher order statistics. The method of extracting second order statistical texture features is done using Gray Level Co-occurrence Matrix (GLCM). First order texture measure is not related to pixel neighbor relationships and it is calculated from the original image. GLCM considers the relation between two pixels at a time, called reference pixel and a neighbor pixel. A GLCM is defined by a matrix in which the number of rows and columns are equal to the number of gray levels G in an image. The matrix element P (I, j | Ax, By) is the relative frequency where I and j represent the intensity and both are separated by a pixel distance Ax, By. The different textural features such as energy, entropy, contrast, homogeneity, correlation, dissimilarity, inverse difference moment and maximum probability can be computed using GLCM matrix.

Significance of Extracted Feature:

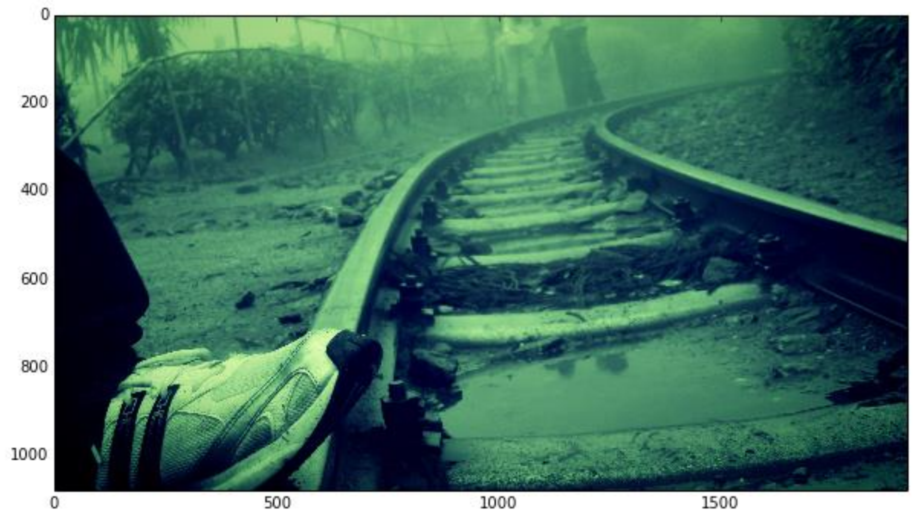
1. Color: It signifies the object identification and extraction of from scene.
2. Brightness: Brightness is one of the most significant pixel characteristics. It should be used only for no quantitative references to physiological sensations and perceptions of light.
3. Entropy: It characterized the texture in image
4. Contrast: Contrast is the dissimilarity or difference between things.
5. Shape of image
6. Size of image
7. Owner, file name, file type etc.

Steps:-

A. Importing an Image:

Importing an image in python is easy. Following code will help you import an image on Python :

```
image = imread(r"C:\Users\Tavish\Desktop\7.jpg")
show_img(image)
```



B. Understanding the underlying data:

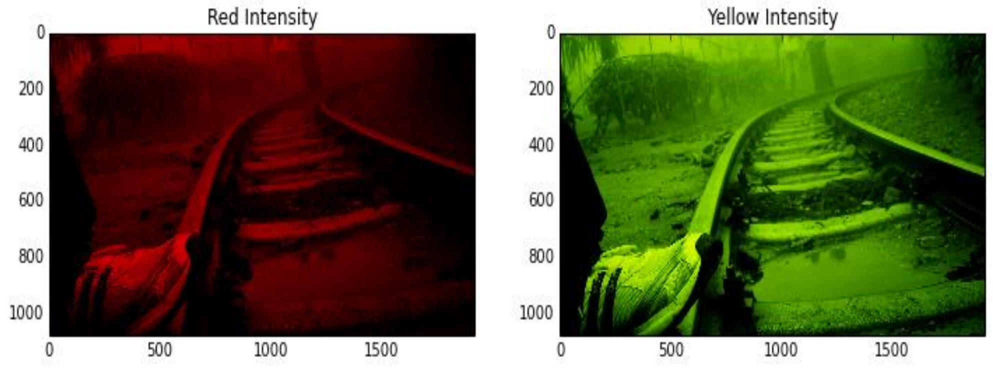
This image has several colors and many pixels.

1. To visualize how this image is stored, think of every pixel as a cell in matrix.
2. Now this cell contains three different intensity information, catering to the color Red, Green and Blue. So a RGB image becomes a 3-D matrix.
3. Each number is the intensity of Red, Blue and Green colors.

```
red, yellow =   image.copy(), image.copy()
```

```
red[:,:,(1,2)] = 0
yellow[:,:,2]=0
```

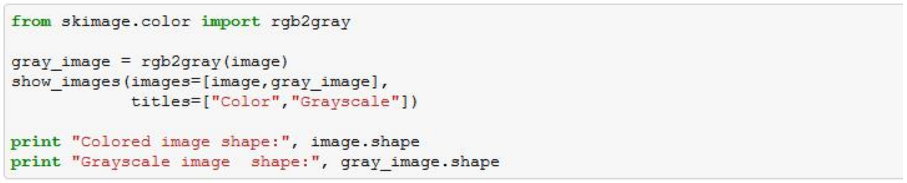
```
show_images(images=[red,yellow], titles=['Red Intensity','Yellow Intensity'])
```



C. Converting Images to a 2-D matrix:-

1. Handling the third dimension of images sometimes can be complex and redundant.
2. In feature extraction, it becomes much simpler if we compress the image to a 2-D matrix.

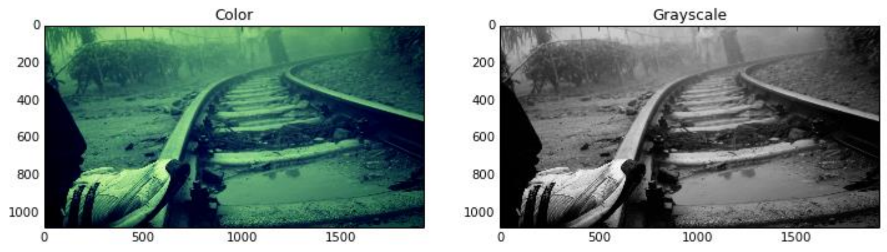
3. This is done by Gray-scaling ,Here is how you convert a RGB image to Gray scale.



```
from skimage.color import rgb2gray

gray_image = rgb2gray(image)
show_images(images=[image,gray_image],
            titles=["Color","Grayscale"])

print "Colored image shape:", image.shape
print "Grayscale image  shape:", gray_image.shape
```

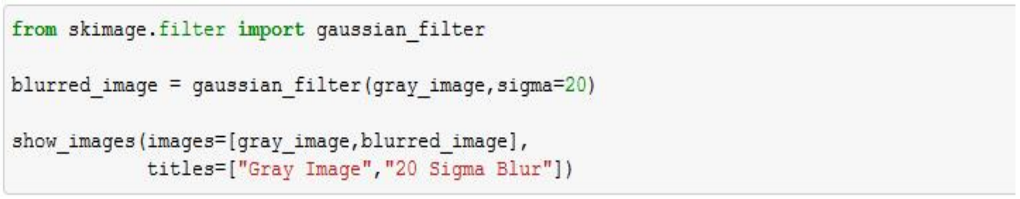


```
Colored image shape: (1080L, 1920L, 3L)
Grayscale image  shape: (1080L, 1920L)
```

Now let’s try to binarize this Gray scale image:-

Blurring an Image:-

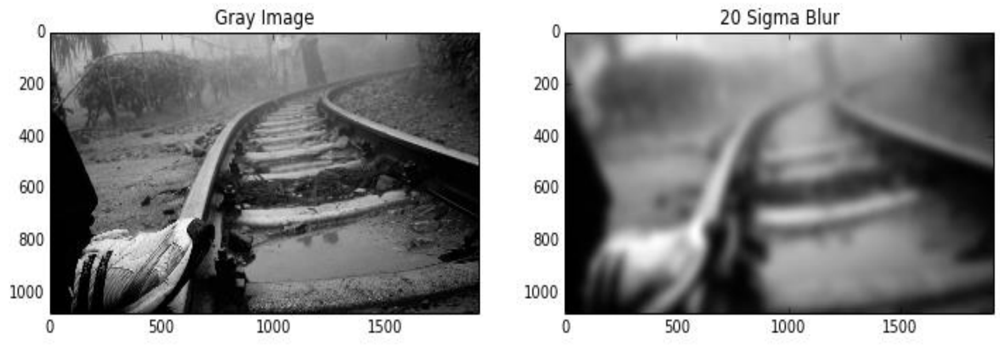
Last part of this assignment is more relevant for feature extraction : Blurring of images.



```
from skimage.filter import gaussian_filter

blurred_image = gaussian_filter(gray_image,sigma=20)

show_images(images=[gray_image,blurred_image],
            titles=["Gray Image","20 Sigma Blur"])
```



Example:-

image = imread(r"C:\Users\Tavish\Desktop\7.jpg")

show_img(image)

red, yellow =   image.copy(), image.copy()

red[:,:,(1,2)] = 0

yellow[:,:,2]=0

show_images(images=[red,yellow], titles=['Red Intensity','Yellow Intensity'])

from skimage.color import rgb2gray

gray_image = rgb2gray(image)

show_images(images=[image,gray_image],titles=["Color","Grayscale"]) print "Colored image shape:", image.shape.

print "Grayscale image shape:", gray_image.shape

from skimage.filter

import threshold_otsu

thresh = threshold_otsu(gray_image)

binary = gray_image > thresh

show_images(images=[gray_image,binary_image,binary],titles=["Grayscale","Otsu Binary"])

from skimage.filter import gaussian_filter

blurred_image = gaussian_filter(gray_image,sigma=20)

show_images(images=[gray_image,blurred_image],titles=["Gray Image","20 Sigma Blur"])

Second Part of the Assignment –Plotting the Histogram

histogram is a graphical representation showing how frequently various colour values occur in the image.

Steps:-

Importing image data:-

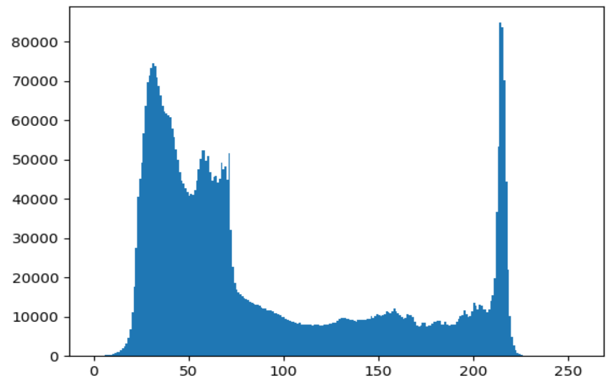
import matplotlib.pyplot as plt #importing matplotlib

The image should be used in a PNG file as matplotlib supports only PNG images.

img = plt.imread('flower.png') #reads image data

Histogram creation using numpy array:-

- To create a histogram of our image data, we use the hist() function.
- plt.hist(n_img.ravel(), bins=256, range=(0.0, 1.0), fc='k', ec='k') #calculating histogram



Histogram Calculation:-

- Here, we use cv2.calcHist()(in-built function in OpenCV) to find the histogram.
- cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])

images : it is the source image of type uint8 or float32 represented as "[img]". channels : it is the index of channel for which we calculate histogram.

For grayscale image, its value is [0] and color image, you can pass [0], [1] or [2] to calculate histogram of blue, green or red channel respectively.

mask : mask image. To find histogram of full image, it is given as "None".

histSize : this represents our BIN count. For full scale, we pass [256].

ranges : this is our RANGE. Normally, it is [0,256].

Example:-

# load an image in grayscale mode

img = cv2.imread('ex.jpg',0)

# calculate frequency of pixels in range 0-255

histg = cv2.calcHist([img],[0],None,[256],[0,256])

Then, we need to plot histogram to show the characteristics of an image.

Plotting Histograms

Analysis using Matplotlib:

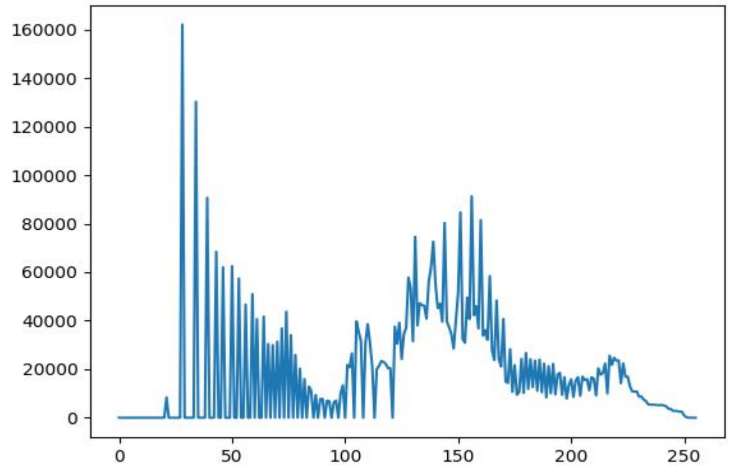
# importing required libraries of opencv

import cv2

```
# importing library for plotting
from matplotlib import pyplot as plt
# reads an input image
img = cv2.imread('ex.jpg',0)
# find frequency of pixels in range 0-255
histr = cv2.calcHist([img],[0],None,[256],[0,256])
# show the plotting graph of an image
plt.plot(histr)
plt.show()
```

Input:

Output:



Algorithm
1. Open colored 2D bitmap file in binary mode.
2. Read the header structure
3. Extract the various feature
4. Print the values of features

Conclusion: Thus, we have successfully implemented a program for feature extraction in 2D colour images and plotted histogram for the features.

Assignment No. 7

Problem Statement:

Build the web crawler to pull product information and links from an e-commerce website. (Python)

Objective: -

To understand the working of web crawler and implement it

Outcomes:

At the end of the assignment the students should have

1. Understood how web crawler works

Infrastructure: Desktop/ laptop system with Linux or its derivatives.

Software used: LINUX/ Windows OS/ Virtual Machine/ IOS/C/C++/Java

Theory:

Search Engines

A program that searches documents for specified keywords and returns a list of the documents where the keywords were found is a search engine. Although search engine is really a general class of programs, the term is often used to specifically describe systems like Google, Alta Vista and Excite that enable users to search for documents on the World Wide Web and USENET newsgroups.

Typically, a search engine works by sending out a spider to fetch as many documents as possible. Another program, called an indexer, then reads these documents and creates an index based on the words contained in each document. Each search engine uses a proprietary algorithm to create its indices such that, ideally, only meaningful results are returned for each query. Search engines are special sites on the Web that are designed to help people find information stored on other sites. There are differences in the ways. Various search engines work, but they all perform three basic tasks:

- They search the Internet - based on important words.
- They keep an index of the words they find, and where they find them.
- They allow users to look for words or combinations of words found in that index.

Fig.1 shows general search engine architecture. Every engine relies on a crawler module to provide the grist for its operation. Crawlers are small programs that browse the Web on the search engine's behalf, similar to how a human user would follow links to reach different pages. The programs are given a starting set of URLs, whose pages they retrieve from the Web. The crawlers extract URLs appearing in the retrieved pages, and give this information to the crawler control module. This module determines what links to visit next, and feeds the links to visit back to the crawlers. (Some of the functionality of the crawler control module may be implemented by the crawlers themselves.) The crawlers also pass the retrieved pages into a page repository. Crawlers continue visiting the Web, until local resources, such as storages, are exhausted.

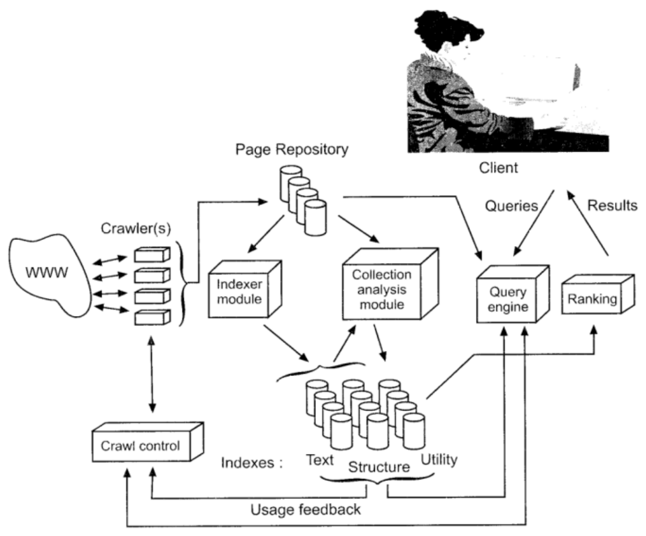


Fig: General search engine architecture

Web Crawlers

Web crawlers are programs that exploit the graph structure of the Web to move from page to page. It may be observed that the noun 'crawler' is not indicative of the speed of these programs, as they can be considerably fast. A key motivation for designing Web crawlers has been to retrieve Web pages and add them or their representations to a local repository. Such a repository may then serve particular application needs such as those of a Web search engine. In its simplest form, a crawler starts from a seed page and then uses the external links within it to attend to other pages. The Crawler is the means by which Web crawler collects pages from the Web. It operates by iteratively downloading a web page, processing it, and following the links in that page to other Web pages, perhaps on other servers.

The end result of crawling is a collection of Web pages, HTML or plain text at a central location practice.

In a more traditional IR system, the documents to be indexed are available locally in a database or file system. Web crawler's first information retrieval system was based on Salton's vector- space retrieval model. The first system used a simple vector-space retrieval model. In the vector- space model, the queries and documents represent vectors in a highly dimensional word space. The component of the vector in a particular dimension is the significance of the word to the document. For example, if a particular word is very significant to a document, the component of the vector along that word's axis would be strong. In this vector space, then, the task of querying becomes that of determining what document vectors are most similar to the query vector. Practically speaking, this task amounts to comparing the query vector, component by component, to all the document vectors that have a word in common with the query vector. WebCrawler determined a similarity number for each of these comparisons that formed the basis of the relevance score returned to the user.

Web crawler's first IR system had three pieces: a query processing module, an inverted full-text index, and a metadata store. The query processing module parses the searcher's query, looks up the words in the inverted index, forms the result list, looks up the metadata for each result, and builds the HTML for the result page. The query processing module used a series of data structures and algorithms to generate results for a given query. First, this module put the query in a canonical form, and parsed each space- separated word in the query. If necessary, each word was converted to its singular form using a modified Porter stemming algorithm and all words were filtered through the stop list to obtain the final list of words. Finally, the query processor looked up each word in the dictionary, and ordered the list of words for optimal query execution. Web crawler's key contribution to distributed systems is to show that a reliable, scalable, and responsive system can be built using simple techniques for handling distribution, load balancing, and fault tolerance.

Robot Exclusion

The Robot Exclusion Standard, also known as the Robots Exclusion Protocol or robots.txt protocol, is a convention to prevent co-operating web crawlers and other web robots from accessing all or a part of a website which is otherwise publicly viewable. Robots are often used by search engines to categorize and archive web sites, or by webmasters to proofread source code.

The standard is different but can be used in conjunction with sitemaps, a robot inclusion standard for websites A robots.txt file on a website will function as a request that specified robots ignore specified files or directories in their search. This might be, for example, out of preference for privacy from search engine results, or the belief that the content of the selected directories might be misleading or irrelevant to the categorization of the site as a whole, or out of desire that an application only operates on certain data. A person may not want certain pages indexed. Crawlers should obey the Robot Exclusion Protocol.

The Robots Exclusion Protocol (REP) is a very simple but powerful mechanism available to webmasters and SEOs alike. Perhaps it is the simplicity of the file that means it is often overlooked and often the cause of one or more critical SE0 issues. To this end, we have attempted to pull together tricks, tips and examples to assist with the implementation and management of your robots.txt file. As many of the non-standard REP declarations supported by Google, Yahoo and Bing may change, we will be providing updates to this in the future. The robots.txt file defines the Robots Exclusion Protocol (REP) for a website. The file defines directives that exclude Web robots from directories or files per website host. The robots.txt file defines crawling directives, not indexing directives. Good Web robots adhere to directives in your robots.txt file. Bad Web robots may not. Do not rely on the robots.txt file to protect private or sensitive data from search engines. The robots.txt file is publicly accessible and so do not include any files or folders that may include business critical information. For example: Website analytics folders (/web stats/, /stats/ etc.) Test or development areas (/test/, /dev/) XML Sitemap element if your URL structure contains vital taxonomy.

If a URL redirects to a URL that is blocked by a robots.txt file, the first URL will be reported as being blocked by robots.txt in Google Webmaster Tools. Search engines may cache your robots.txt file (For example: Google may cache your robots.txt file for 24 hours). When deploying a new website from a development environment always check the robots.txt file to ensure no key directories are excluded. Excluding files using robots.txt may not save the crawl budget from the same crawl session. For example: if Google cannot access a number of files it may not crawl other files in their place. URLs excluded by REP (Robots Exclusion Protocol) may still appear in a search engine index.

Program Implementation: Code written in Python to implement of the same with appropriate output.

Algorithm

1. Make User Interface
2. Input the URL of any website
3. Establish HTTP connection
4. Read HTML page source code
5. Extract Hyperlinks of HTML page
6. Display the list of hyperlinks on the same page

Conclusion: Implementation is concluded by stating the basic working of web crawler.

Assignment No. 8

Problem Statement:

Write a program to find the live weather report (temperature, wind speed, description, and weather) of a given city. (Python).

Objective: -

1. To Get Weather Information using Python
2. To evaluate the performance of the IR system and understand user interfaces for searching.
3. To understand information sharing on the web

Outcomes:

At the end of the assignment the students should have

1. Understood and implemented the program to find the live weather report using python.

Infrastructure: Desktop/ laptop system with Linux or its derivatives.

Software used: LINUX/ Windows OS/ Virtual Machine/ IOS/Python 3.9

Theory:

Weather Information using Python

Python is a growing language that has become increasingly popular in the programming community. One of Python's key features is that it's easy to work with APIs on websites and many weather entities have their API which you can access with just a couple of lines of code. One such great API is the Open Weather Map's API, with it you can build a small program to access any given location's weather forecast anywhere across the globe! This article will help you to get weather information using Python.

What is OpenWeatherMap?

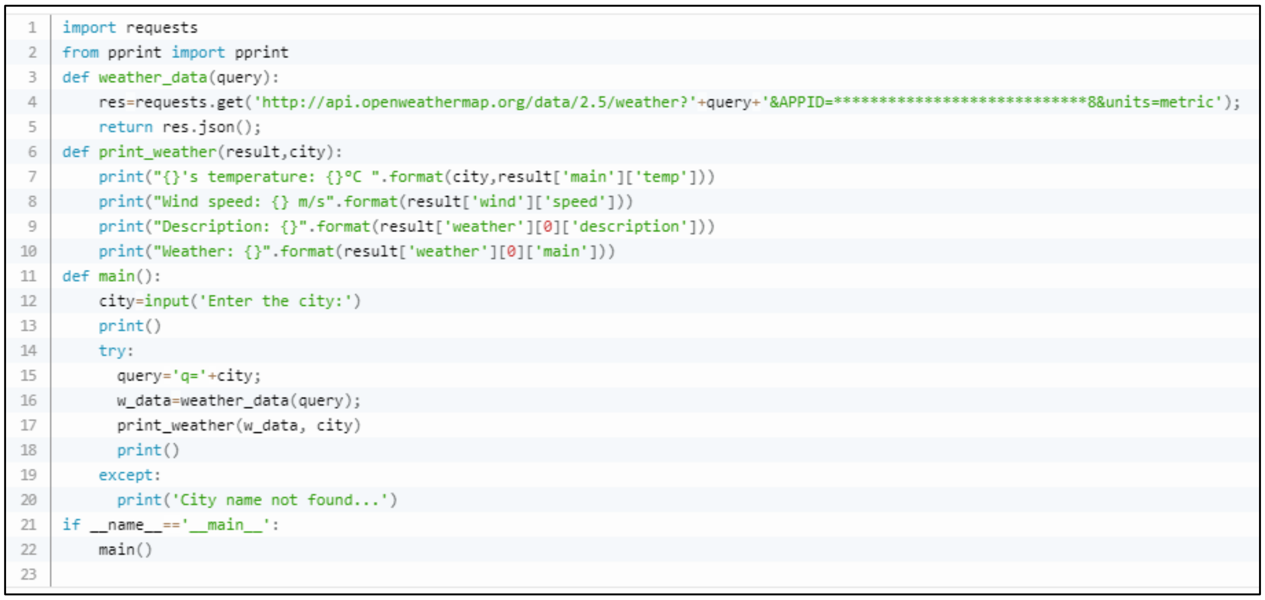
The OpenWeatherMap (OWM) is a helpful and free way to gather and display weather information. Because it's an open-source project, it's also free to use and modify in any way. OWM offers a variety of features and is very flexible. Because of these qualities, it offers a lot of benefits to developers. One of the major benefits of OWM is that it's ready to go. Unlike other weather applications, OWM has a web API that's ready to use.

You don't have

to install any software or set up a database to get it up and running. This is a great option for developers who want to get weather reading on a website quickly and efficiently.

It has an API that supports HTML, XML, and JSON endpoints. Current weather information extended forecasts, and graphical maps can be requested by users. These maps show cloud cover, wind speed as well as pressure, and precipitation.

Python Code:



```python
import requests
from pprint import pprint
def weather_data(query):
    res=requests.get('http://api.openweathermap.org/data/2.5/weather?'+query+'&APPID=****************************8&units=metric');
    return res.json();
def print_weather(result,city):
    print("{}'s temperature: {}°C ".format(city,result['main']['temp']))
    print("Wind speed: {} m/s".format(result['wind']['speed']))
    print("Description: {}".format(result['weather'][0]['description']))
    print("Weather: {}".format(result['weather'][0]['main']))
def main():
    city=input('Enter the city:')
    print()
    try:
      query='q='+city;
      w_data=weather_data(query);
      print_weather(w_data, city)
      print()
    except:
      print('City name not found...')
if __name__=='__main__':
    main()

```

Sample Output:

```
Enter the city: Brazil
Brazil's temperature: 16.45°C
Wind speed: 2.1 m/s
Description: clear sky
Weather: Clear
```

Conclusion: Thus, we have successfully implemented a program to find the live weather report (temperature, wind speed, description, and weather) of a given city using Python.

Assignment No. 9

Problem Statement:

Case study on recommender system for a product / Doctor / Product price / Music

Objective: -

1. To study recommender system

Outcomes:

At the end of the assignment the students should have

1. Understood the concept of collaborative recommender system

Theory:

(Do study of collaborative or content based recommender system)

The E-Commerce platform has seen enormous growth in online platforms in recent years. Product recommendations are extremely complex. This leads to a large number of combinations that can be overwhelming and extremely difficult to calculate recommendations.

The paradigm of machine learning and natural language processing comes in picture in achieving this goal of product recommendation. Through the implementation of these approaches, the products can be effectively reviewed and realized for their potential for recommendation to a particular user for a product / Doctor / Product price / Music.

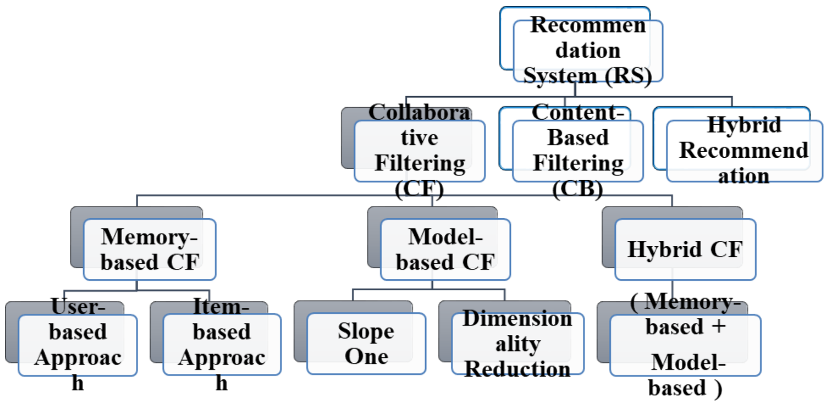


Fig: Recommendation system block diagram

Conclusion: Thus we have studied collaborative recommender system