A Framework for Hierarchical Clustering Based Indexing in Search Engines

Parul Gupta[1] and **A.K. Sharma**[2]

Submitted in May 2010; Accepted in June 2011

Abstract - ***Granting efficient and fast accesses to the index is a key issue for performances of Web Search Engines. In order to enhance memory utilization and favor fast query resolution, WSEs use Inverted File (IF) indexes that consist of an array of the posting lists where each posting list is associated with a term and contains the term as well as the identifiers of the documents containing the term. Since the document identifiers are stored in sorted order, they can be stored as the difference between the successive documents so as to reduce the size of the index. This paper describes a clustering algorithm that aims at partitioning the set of documents into ordered clusters so that the documents within the same cluster are similar and are being assigned the closer document identifiers. Thus the average value of the differences between the successive documents will be minimized and hence storage space would be saved. The paper further presents the extension of this clustering algorithm to be applied for the hierarchical clustering in which similar clusters are clubbed to form a mega cluster and similar mega clusters are then combined to form super cluster. Thus the paper describes the different levels of clustering which optimizes the search process by directing the search to a specific path from higher levels of clustering to the lower levels i.e. from super clusters to mega clusters, then to clusters and finally to the individual documents so that the user gets the best possible matching results in minimum possible time.***

Index Terms - Inverted files, Index compression, Document Identifiers Assignment, Hierarchical Clustering

1. INTRODUCTION

The indexing phase [1] of search engine can be viewed as a Web Content Mining process. Starting from a collection of unstructured documents, the indexer extracts a large amount of information like the list of documents, which contain a given term. It also keeps account of number of all the occurrences of each term within every document. This information is maintained in an index, which is usually represented using an inverted file (IF). IF is the most widely adopted format for this index due to its relatively small size occupancy and the efficiency involved in resolution of the keywords based queries. The index consists of an array of the posting lists where each posting list is associated with a term and contains the term as well as the identifiers of the documents containing the term. Since the document identifiers are stored in sorted order, they can be stored as the difference between the successive documents so as to reduce the size of the index. Storing the differences require coding of small integer values [7] which can be encoded with a small number of bits and also aids in compressing the index. So if the similar documents [1] are assigned the closer document identifiers, then in the posting lists, the average value of the difference between the successive documents will be minimized and hence storage space would be saved. For example, consider the posting list ((job;5) 1, 4, 14, 20, 27) indicating that the term job appears in five documents having the document identifiers 1,4,14,20,27 respectively. The above posting list can be written as ((job; 5) 1, 3, 10, 6, 7) where the items of the list represent the difference between the successive document identifiers. The figure 1 shows the example entries in the index file.

Term	No. of docs in which term appears	Doc ids of docs in which term appears	Doc ids stored with difference coding scheme
Job	50	12,34,45,49…	12,22,11,4…
Engineer	59	15,20,34,55…	15,5,14,21…
Fresher	15	3,6,9,12…	3,3,3,3…
Analyst	5	2,4,5,8,9	2,2,1,3,1

Figure 1: Example Entries in the Index File

Clustering is a widely adopted technique aimed at dividing a collection of data into disjoint groups of homogenous elements. Document clustering [3] has been widely investigated as a technique to improve effectiveness and efficiency in information retrieval. Clustering algorithms attempt to group together the documents based on their similarities. Thus documents relating to a certain topic will hopefully be placed in a single cluster. So if the documents are clustered, comparisons of the documents against the user's query are only needed with certain clusters and not with the whole collection of documents. The fast information retrieval can be further achieved by hierarchical clustering in which the similar clusters are merged together to form higher levels of clustering. In this paper, the proposed heuristic exploits a text clustering algorithm that reorder the collection of documents on the basis of document similarity. The reordering is then used to assign close document identifiers to similar documents thus reducing differences between the document identifiers and enhancing the compressibility of the IF index representing the collection. The proposed clustering algorithm aims at partitioning the set of documents into k ordered clusters on the basis of similarity measure so that the documents on the web are assigned the identifiers in such a way that the similar documents are being assigned the closer document identifiers. Further the extension of this clustering algorithm has been presented to be applied for

[1, 2] *Department of Computer Engineering, Y.M.C.A. University of Science and Technology, Faridabad*
E-Mail: [1]*parulgupta_gem@yahoo.com and* [2]*ashokkale@rediffmail.com*

hierarchical clustering [5] in which similar clusters are clubbed to form a mega cluster and similar mega clusters are then combined to form super cluster. Thus the different levels of clustering have been defined which aids in better indexing. As a result of clustering, the size of the index gets compressed and moreover, it also optimizes the search process by directing the search to a specific path from higher levels of clustering to the lower levels i.e. from super clusters to mega clusters, then to clusters and finally to the individual documents so that the user gets the best possible matching results in minimum possible time.

2. RELATED WORK

In this paper, a review of previous work on document clustering algorithms is given. In this field of clustering, many algorithms have already been proposed but they seem to be less efficient in clustering together the most similar documents thus making the use of clustering less effective. K-means algorithm [4, 6] has been proposed in this direction, which initially chooses k documents as cluster representatives and then assigns the remaining nk documents to one of these clusters on the basis of similarity between the documents. New centroids for the k clusters are recomputed and documents are reassigned according to their similarity with the k new centroids. This process repeats until the position of the centroids become stable. Computing new centroids is expensive for large values of n and the number of iterations required to converge may be large.

Another work proposed was the reordering algorithm [1] which partitions the set of documents into k ordered clusters on the basis of similarity measure. According to this algorithm, the biggest document is selected as centroid of the first cluster and n/k1 most similar documents are assigned to this cluster. Then the biggest document is selected and the same process repeats. The process keeps on repeating until all the k clusters are formed and each cluster gets completed with n/k documents. This algorithm is not effective in clustering the most similar documents. The biggest document may not have similarity with any of the documents but still it is taken as the representative of the cluster.

Another proposed work was the threshold based clustering algorithm [8] in which the number of clusters is unknown. However, two documents are classified to the same cluster if the similarity between them is below a specified threshold. This threshold is defined by the user before the algorithm starts. It is easy to see that if the threshold is small, all the elements will get assigned to different clusters. If the threshold is large, the elements may get assigned to just one cluster. Thus the algorithm is sensitive to specification of threshold.

Fuzzy Co-clustering of Web Documents is a technique to simultaneously cluster data (or objects) and features. In case of web, web documents are the data, and the words inside the documents are the features. By performing simultaneous clustering of documents and words, meaningful clusters of highly coherent documents can be generated relative to the highly relevant words, as opposed to clusters of documents with respect to all the words as in the case of standard clustering algorithm. FCCM algorithm proposed in this direction is aimed at clustering data in which attributes can be categorical (nominal) and the distance or similarity between two patterns is not explicitly available. FCCM accomplishes this task by maximizing the degree of 'aggregation' among the clusters. The major drawback of FCCM is that it poses problems when the no. of documents or words is large. Moreover this algorithm is less effective when data has large number of overlapping clusters.

In this paper, the proposed algorithm has tried to remove the shortcomings of the existing algorithms. It produces a better ordering of the documents in the cluster. This algorithm picks the first document as cluster representative, then selects the most similar document to it and puts it in the cluster, it further selects document which is most similar to the currently selected document and repeats until the first cluster becomes full with n/k documents. The same process is then repeated to form the rest of the clusters. Thus the most similar documents are accumulated in the same cluster and are assigned consecutive document identifiers. Thus the algorithm is more efficient in compression of the index.

3. PROPOSED ALGORITHM FOR CLUSTERING BASED INDEXING

Let D ={D1, D2,. . ., DN} be a collection of N textual documents to which consecutive integer document identifiers 1, . . . ,N are initially assigned. Moreover, let T be the number of distinct terms ti, i = 1, . .. , T present in the documents, and t the average length of terms. The total size CSize (D) [27] of an IF index for D can be written as:

$$\boxed{\text{Csize (D)} = \text{CSize}_{\text{lexicon}} (\text{T}. \mu\text{t}) + \sum \text{Encode}_{\text{m}} (\text{d\_gaps (ti))}}$$

where $\text{CSize}_{\text{lexicon}}$ (T. μt) is the number of bytes needed to code the lexicon, while d_gaps (ti) is the d_gap [28] representation of the posting list associated to term ti, and Encodem is a function that returns the number of bytes required to code a list of d gaps according to a given encoding method m.

The compression of index is achieved by applying clustering to the web pages so that the similar web pages are in the same cluster and hence assigned closer identifiers. A clustering algorithm has been proposed, which converts the individual documents into k ordered clusters, and hence documents are reassigned new document identifiers so that the documents in the same cluster get the consecutive document identifiers. The clustering of the documents is done on the basis of similarity between the documents, which is first of all calculated using some similarity measure. The proposed architecture for the clustering based indexing system is given in figure 2.

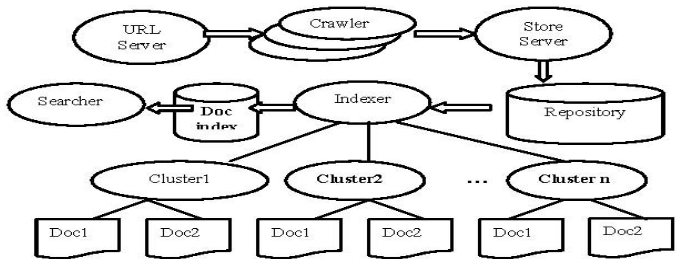


Figure 2: Architecture of Clustering based indexing System in Search Engines

A. Computing The Similarity Matrix

Let D={D1, D2,……Dn) be the collection of N textual documents being crawled to which consecutive integers document identifiers 1…n are assigned. Each document Di can be represented by a corresponding set Si such that Si is a set of all the terms contained in Di. Let us denote that set by D* such that D*={S1,S2,……….. Sn}. The similarity of any two documents Si and Sj can be computed using the similarity measure [1]:

Similarity_measure (Si, Sj) = |Si ∧ Sj | / |Si U Sj |

INPUT – The set D*= {S1, S2, S3, S4…Sn} where Si is a set of all the terms of document Di.

–The number k of clusters to create.

OUTPUT – k ordered clusters representing a reordering of D

The algorithm that calculates the similarity of each document with every other document using the similarity_measure given above is given in figure 3.

```
Algorithm docum_similarity:

for i=1 to n
begin
  sim[i][i]=0;
  for j=i+1 to n
  begin
      sim[i][j]=similarity_measure(Si, Sj)
      sim[j][I]=sim[i][j]
  end for
end for
```

Figure 3: Algorithm for computing similarity matrix

The above algorithm constructs the document similarity matrix [15]. The number of calculations performed leads to formation of the upper triangular matrix. The rest of the values in the similarity matrix are assigned automatically as we know similarity_measure (i, j)= similarity_measure (j, i).

B. The Algorithm

The clustering algorithm which clusters together the similar documents is given below:

```
Algorithm docum_clustering
i=1
for f=1 to k                    // for number of clusters
begin
cf=0      //Initially cluster is empty with no document in it
for e=1 to n/k          // for number of documents in one cluster
begin
for j=1 to n
        Select max from sim[i][j]
                cf=cf U Si
                D*=D*- Si
                for l= 1 to n
                begin
                        sim[i][l]=0
                        sim[l][i]=0
                end
i=j
end
end
```

Figure 4: Algorithm for Clustering (docum_clustering)

It may be noted that the algorithm starts with the first cluster which is empty initially. The first document from the collection is considered and put in the first cluster. Now, using the similarity matrix, the most similar document to it is considered. All the entries of the row and column associated with the first document are made zero as this document cannot be added to any other cluster. The most similar document picked is put in the same cluster. Now the second document that was considered takes the role of the first document and the most similar document to it is considered and this procedure repeats for n/k times when the first cluster gets full. Thus at the end, we get k clusters each with n/k number of similar documents.

C. Example Illustrating Clusters Formation

Having discussed the algorithm, let us now have panoramic view as to how the clustering of the documents takes place. For e.g. if we have 10 documents – A, B, C, D, E, F, G, H, I, J & value of k is 2 i.e. 2 clusters are to be made, then according to the algorithm, the similarity among the documents is computed using the similarity measure and hence the formed upper triangular similarity matrix will be:

	A	B	C	D	E	F	G	H	I	J
A	0	5	3	6	9	8	2	3	4	1
B		0	5	4	6	2	3	5	7	8
C			0	5	2	3	6	9	4	7
D				0	2	3	6	5	4	9
E					0	8	5	3	6	5
F						0	8	9	5	2
G							0	6	5	4
H								0	3	6
I									0	5
J										0

Figure 5: Initial Similarity Matrix

Now from the computed values in the upper triangular matrix, the matrix can be completed as follows using the property that similarity_measure(i,j) = similarity_measure(j,i). The full similarity matrix is given in the figure 6. According to the clustering algorithm,

- 1st cluster will have A, then E, then F, then H & lastly C
- 2nd cluster will have J, then D, then G, then I & lastly B

	A	B	C	D	E	F	G	H	I	J
A	0	5	3	6	9	8	2	3	4	1
B	5	0	5	4	6	2	3	5	7	8
C	3	5	0	5	2	3	6	9	4	7
D	6	4	5	0	2	3	6	5	4	9
E	9	6	2	2	0	8	5	3	6	5
F	8	2	3	3	8	0	8	9	5	2
G	2	3	6	6	5	8	0	6	5	4
H	3	5	9	5	3	9	6	0	3	6
I	4	7	4	4	6	5	5	3	0	5
J	1	8	7	9	5	2	4	6	5	0

Figure 6: Full Similarity Matrix

The output after calculating similarity for first five documents will be :

	A	B	C	D	E	F	G	H	I	J
A	0	0	0	0	0	0	0	0	0	0
B	0	0	0	4	0	0	3	0	7	8
C	0	0	0	0	0	0	0	0	0	0
D	0	4	0	0	0	0	6	0	4	9
E	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0
G	0	3	0	6	0	0	0	0	5	4
H	0	0	0	0	0	0	0	0	0	0
I	0	7	0	4	0	0	5	0	0	5
J	0	8	0	9	0	0	4	0	5	0

Figure 7: Matrix after formation of first cluster

4. PROPOSED HIERARCHICAL CLUSTERING ALGORITHM

The lack of a central structure and freedom from a strict syntax is responsible for making a vast amount of information available on the web, but retrieving this information is not easy. One possible solution is to create a static hierarchical categorization of the entire web and using these categories to organize the web pages. Organizing Web pages into a hierarchy of topics and subtopics facilitates browsing the collection and locating results of interest. In hierarchical clustering algorithm, after the cluster of similar documents have been formed, the similar clusters are merged together to form the mega clusters using the same similarity measure as is used to cluster together the documents. The framework for the hierarchical clustering is shown in figure 8.

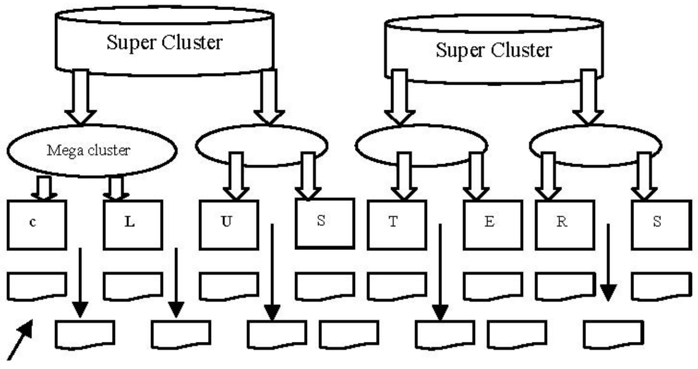


Figure 8: Hierarchical Clustering

A. Computation of Similarity Matrix of Clusters

The algorithm that computes the similarity matrix for the similar clusters is given below. In this algorithm, D={S1,S2….Sk} where Si is a set of terms in the cluster ci.

```
Algorithm cluster_similarity
for i = 1 to k
begin
   sim [i][i] = 0
   for j = i+1 to k
   begin
         sim [i][j] = similarity_measure (Si, Sj)
         sim[j][i] = sim[i][j]
   end
end
```

Figure 9: Algorithm for similarity matrix of clusters

B. Algorithm for Hierarchical Clustering

The hierarchical clustering [9] algorithm that aims at forming the mega clusters out of the similar clusters is given in figure 10.

In this algorithm, the first mega cluster is considered which is initially empty. The first cluster from the collection is considered and put in the first mega cluster. Now, using the similarity matrix, the most similar cluster to it is considered. All the entries of the row and column associated with the first cluster are made zero as this cluster cannot be added to any other mega cluster. The most similar cluster picked is put in the same mega cluster. Now the second cluster that was considered takes the role of the first cluster and the most similar cluster to it is considered and this procedure repeats for k/m times when the first mega cluster gets full. Now the second mega cluster is considered and the same procedure repeats until all the mega clusters get full. Thus at the end, we get m mega clusters each with k/m number of clusters such that the clusters within the same mega cluster are similar.

```
Algorithm mega_clustering
i=1
for f=1 to m
begin
      cf = 0
       for e = 1 to k/m
       begin
               for j = 1 to k
               select max from sim [i][j]
               cf = cf U si
               D= D-si
               for l=1 to k
               begin
                    sim [l][i] = 0
                    sim [i][l] = 0
              end
              i=j
      end
end
```

Figure 10: Algorithm for Mega Clustering

C. Example Illustrating Hierarchical Clustering

For e.g. user has to fire a query "Jobs for Computer Engineers having 5 to 10 years experience". Now since the hierarchy of clusters has been formed, so the search will proceed in the manner as shown in the figure 12. The search will start from the super cluster "Job", will be directed to the mega cluster "COMP. ENGG.", then will reach the cluster "5 TO 10 YRS EXPERIENCE" and finally will reach the individual relevant documents. Thus the search follows a specific path from super cluster to the individual document as shown in figure 11.

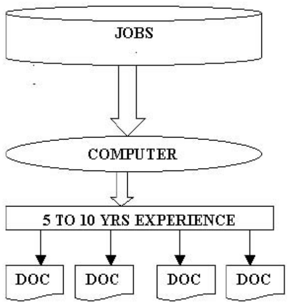


Figure 11: Search Path

5. IMPLEMENTATION OF PROPOSED WORK

For indexing the documents, firstly we have to parse the documents. After that similarity matrix is created and then k means algorithm is applied for creating the clusters. Clusters will be created at first level .For creating clusters at second level same procedure is applied again and then finally hierarchical clustering is done for indexing.

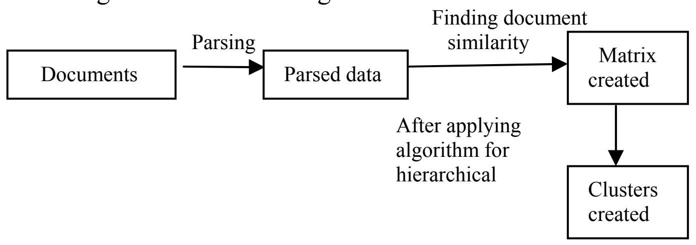
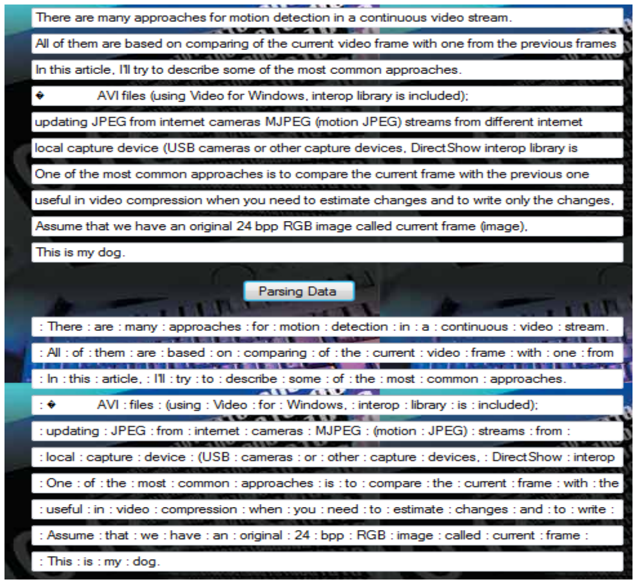

Documents
Parsing
Parsed data
Finding document similarity
Matrix created
After applying algorithm for hierarchical
Clusters created

Figure 11: Work Flow of Implementation

A. Snapshots of Implemented Work

1. Given input & parsed data, the following snapshot represents the parsed data which is the initial step for indexing the data.



There are many approaches for motion detection in a continuous video stream.
All of them are based on comparing of the current video frame with one from the previous frames
In this article, I'll try to describe some of the most common approaches.
◆ AVI files (using Video for Windows, interop library is included);
updating JPEG from internet cameras MJPEG (motion JPEG) streams from different internet
local capture device (USB cameras or other capture devices, DirectShow interop library is
One of the most common approaches is to compare the current frame with the previous one
useful in video compression when you need to estimate changes and to write only the changes,
Assume that we have an original 24 bpp RGB image called current frame (image),
This is my dog.

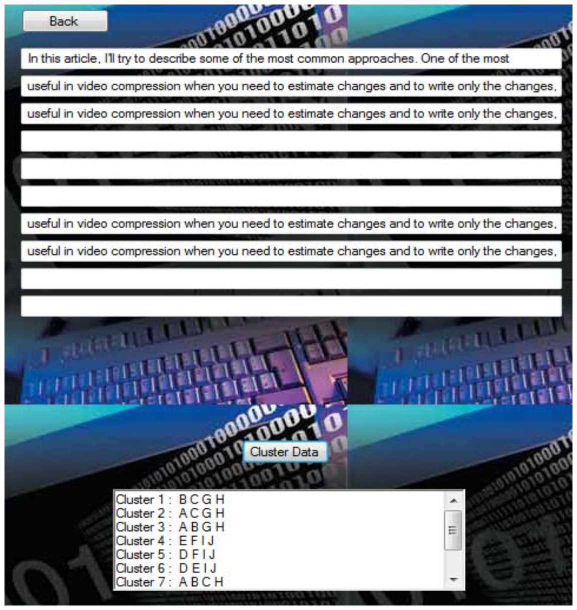
Parsing Data

: There : are : many : approaches : for : motion : detection : in : a : continuous : video : stream.
: All : of : them : are : based : on : comparing : of : the : current : video : frame : with : one : from
: In : this : article, : I'll : try : to : describe : some : of : the : most : common : approaches.
: ◆ AVI : files : (using : Video : for : Windows, : interop : library : is : included);
: updating : JPEG : from : internet : cameras : MJPEG : (motion : JPEG) : streams : from :
: local : capture : device : (USB : cameras : or : other : capture : devices, : DirectShow : interop
: One : of : the : most : common : approaches : is : to : compare : the : current : frame : with : the
: useful : in : video : compression : when : you : need : to : estimate : changes : and : to : write :
: Assume : that : we : have : an : original : 24 : bpp : RGB : image : called : current : frame :
: This : is : my : dog.

Figure 12: Given and Parsed data

2. Clustered data

The data is now clustered according to the similarity of words.The following figure shows the clusters created that are created after matching the similarity of documents with each other.



Back

In this article, I'll try to describe some of the most common approaches. One of the most
useful in video compression when you need to estimate changes and to write only the changes,
useful in video compression when you need to estimate changes and to write only the changes,
useful in video compression when you need to estimate changes and to write only the changes,
useful in video compression when you need to estimate changes and to write only the changes,

Cluster Data

Cluster 1 : B C G H
Cluster 2 : A C G H
Cluster 3 : A B G H
Cluster 4 : E F I J
Cluster 5 : D F I J
Cluster 6 : D E I J
Cluster 7 : A B C H

Figure 13: Clustered data

B. Results

A- Graph Representing the Created Clusters at First Level

At first level, less no of clusters are created. As at first level the similarity between two documents is less.

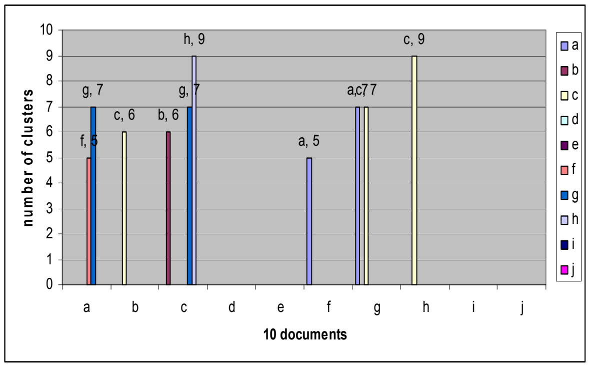


Figure 14: Clusters created at first level

B-Graph Representing the Created Clusters at Second Level

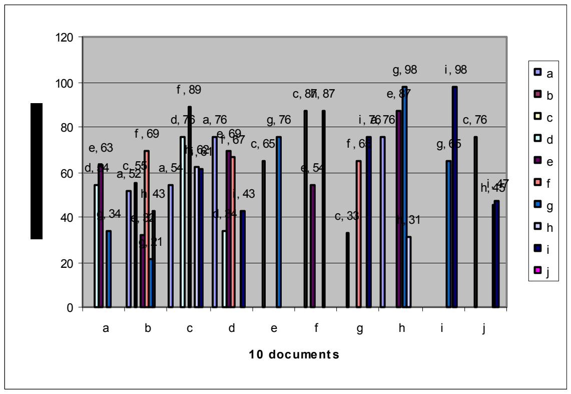


Figure 15: Clusters created at second level

At second level, more no of clusters are created in comparison to first level. As the similarity between two documents is more.

6. CONCLUSIONS

In this paper, an efficient algorithm for computing a reordering of a collection of textual documents has been presented that effectively enhances the compressibility of the IF index built over the reordered collection. Further, the proposed hierarchical clustering algorithm aims at optimizing the search process by forming different levels of hierarchy. The proposed algorithm is superior to the other algorithms as a summarizing and browsing tool. A critical look at the literature indicates that in contrast to the earlier proposed algorithms, the proposed work produces a better ordering of the following advantages:

1. Compression of Index Size: The size index of the index is compressed as similar documents are assigned closer document identifiers.

2. Reduction in Search Time: The search time gets reduced as the search gets directed to a specific path from super cluster to mega clusters, then to clusters and finally to the individual documents.

3. Fast retrieval of relevant documents: Since the similar documents get clustered together in the same cluster, the specific query relevant documents can be rapidly picked from that cluster.

REFERENCES

[1]. Fabrizio Silvestri, Raffaele Perego and Salvatore Orlando. “Assigning Document Identifiers to Enhance Compressibility of Web Search Engines Indexes” In the proceedings of SAC, 2004.

[2]. Van Rijsbergen C.J. “Information Retrieval” Butterworth 1979

[3]. Oren Zamir and Oren Etzioni. “Web Document Clustering: A feasibility demonstration” In the proceedings of SIGIR, 1998.

[4]. Jain and R. Dubes. “Algorithms for Clustering Data.” Prentice Hall, 1988

[5]. Sanjiv K. Bhatia. “Adaptive KMeans Clustering” American Association for Artificial Intelligence, 2004.

[6]. Bhatia, S.K. and Deougan , J.S. 1998. “Conceptual Clustering in Information Retrieval” IEEE Transactions on Systems, Man and Cybernetics.

[7]. Dan Bladford and Guy Blelloch. “Index compression through document reordering” In IEEE, editor, Proc. Of DCC’02. IEEE, 2002.

[8]. Chris Staff: Bookmark Category Web Page Classification Using Four Indexing and Clustering Approaches. AH 2008:345-348

[9]. Khaled M. Hammouda, Mohamed S. Kamel: Efficient Phrase-Based Document Indexing for Web Document Clustering. IEEE Trans. Knowl. Data Eng. (TKDE) 16(10):1279-1296 (2004).