AGGLOMERATIVE CLUSTERING BASED INDEX CONSTRUCTION FOR SEARCH ENGINES USING AGENTS

Rahul Kumar Sharma*

Niraj Singhal**

Abstract

The information on World Wide Web is growing at an exponential rate; therefore search engines are required to index the downloaded web documents. Indexing in search engines is an active area of current researches. The main aim of search engine is to provide best relevant documents to the users in minimum possible time. The major issue for performances of Web search engine is to provide efficient and fast access to the index. Indexing is performed on the web pages after they have been collected into the web page repository by the crawling agent community. Web search engines use inverted file index that 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 collected by crawling agent stored in web page repository and indexer agent extract the document from page repository. Where the matcher agent that will take all the document extracted by the indexer agent for checking the document similarity between them and store the same similarity document in min cluster. In this paper, we presents Agglomerativehierarchical clustering algorithm which aims at partitioning the set of documents into ordered cluster so that the document with in the same cluster are similar and are being assigned the closer document identifiers.

Keywords: web search engine, posting list, agent, Agglomerativehierarchical clustering, document identifiers.

* Computer Science Department, Guru Nanak Education Trust's Group of Institutions, Roorkee, Uttarakhand, India

** School of Computer Engineering and Information Technology, Shobhit University, Meerut, India

1. INTRODUCTION

World Wide Web (WWW or Web) is a huge repository of information consisting of hyperlinked documents spread over the Internet. The indexing phase [1] of search engine can be viewed as a Web content mining process. The crawling agent collect the document from the Web and then temporarily store in to web page repository. The indexer agent extracts 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 retrieved information is maintained in an index, which is usually represented using an inverted file (IF). The index consists of posting lists where each posting list is associated with a term and contains the term as well as the documents identifiers 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. 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 ((computer engineering; 5) 1, 4, 14, 20, 27) indicating that the term computer engineering, appears in five documents having the document identifiers 1, 4,14,20,27 respectively. The above posting list can be written as ((computer engineering; 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 appear	Doc Ids of Docs in which term appear
Computer engineering	30	1, 4, 14, 20, 27,…
News	40	12, 17, 19, 21, 40,..
Job	25	3, 9, 15, 24, 41,….

Figure 1: Example show entries in the index file

Clustering is a technique aimed at dividing a collection of data into disjoint groups of homogenous elements. Document clustering [10] has been widely investigated as a technique to improve effectiveness and efficiency in information retrieval. Clustering algorithms attempt to group the documents together based on their similarities. Thus documents related to a certain

term 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 will be further achieved by Agglomerativehierarchical clustering (shown in figure 2). In which the similar clusters (min cluster) are merged together to form higher levels of clustering (sub cluster). 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.

In this paper, 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 hierarchical clustering [11] in which similar min clusters are grouped to form a sub cluster and similar sub clusters are then combined to form main 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 main clusters to sub clusters, then to min clusters and finally to the individual documents so that the user gets the best possible matching results in minimum possible time.

2. SEARCH ENGINE ARCHITECTURE USING AGENT

Information retrieval tools, like search engines [14], download web pages, texts, images and other multimedia from Web. Search engine is a coordinated set of programs that is able to read every searchable page on the web, create an index of the information it finds, compare that information to a user's search request (i.e. query), finally return the result back to the user. Search engine acts as a bridge between web users and web pages. It is a searchable database which collects the information from web pages on the Internet, indexes the information and then stores the result in a huge database where from it can be searched quickly. A general search engine in (Figure 2) comprises the following components:

- **Crawling Agent Community:** It can be described as a group of crawling agents named bots (also known as spiders or robots) that are dedicated to download Hypertext Mark-up Language (HTML) pages from the Web.
- **Indexer Agent:** Once a number of pages have been filtered, this agent groups the XML pages before indexing them. The Indexer agent ends up with an index structure suitable for retrieving purposes.
- **Web Page Repository:** The information retrieved by the crawling agent is stored in a database called page repository. The indexer indexes the various documents contained in repository. The documents are identified by doc ID, length and URL.
- **Filtering Agent:** Each time a bot downloads an HTML page, a filter agent takes and extracts its contents. The agent filters the resulting text and generates an Extendable Mark-up Language (XML) page whose structure is suitable for being indexed.
- **Interface agent:** This module handles the user interface: it takes the user query and displays the pages that the Information Retrieval System returns when answering the users' queries.

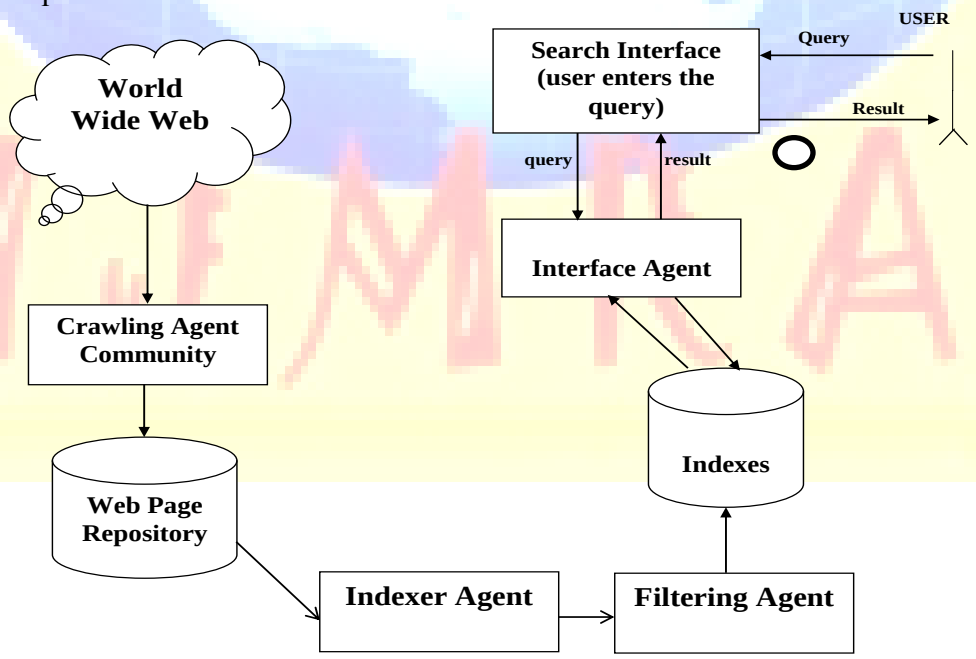


Figure 2: General architecture of search engine using agent.

3. RELATED WORK

Fabrizio Silvestri, Raffaele Perego and Salvatore Orlando [1] proposed the reordering algorithm 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/k' 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.

In the threshold based clustering algorithm [3], 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.

Jain [4] provides an elaborate survey of various clustering techniques. The study presents an overview of pattern clustering methods from statistical pattern recognition perspective, with a goal of providing useful advices and references to fundamental concepts accessible to the broad community of clustering practitioners. It explicates the taxonomy of clustering techniques, and identifies cross-cutting themes and recent advances. Some important applications of clustering algorithms have been applied in various fields such as image segmentation, object recognition, and information retrieval.

Willett [5] presents a detailed study of applying hierarchical clustering algorithms in the document clustering. Agglomerative Hierarchical Clustering algorithms have mostly been used. These algorithms are applied to large document collections. For example, single-link methods typically take O(n^2) time while complete-link methods typically take O(n^3) time.

Cutting et al. [8] adopted various partition-based clustering algorithms for clustering document such as Buckshot and Fractionation. Fractionation is an approximation to Agglomerative Hierarchical Clustering, where the search for the two closest clusters is not performed locally and in a bound region instead of searching globally as in the case of document clustering.

C. Zhou, W. Ding and Na Yang [9], in their paper introduce a double indexing mechanism for search engines based on campus Net. The CNSE consists of crawl machine, Chinese automatic segmentation, index and search machine. The proposed mechanism has document index as well as word index. The document index is based on, where the documents do the clustering, and ordered by the position in each document. During the retrieval, the search engine first gets the document id of the word in the word index, and then goes to the position of corresponding word in the document index. Because in the document index, the word in the same document is adjacent, the search engine directly compares the largest word matching assembly with the sentence that users submit. The mechanism proposed, seems to be time consuming as the index exists at two levels.

4. PROPOSED WORK

In our work, we first parse the documents for document indexing. 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 Agglomerative hierarchical clustering is done for indexing.

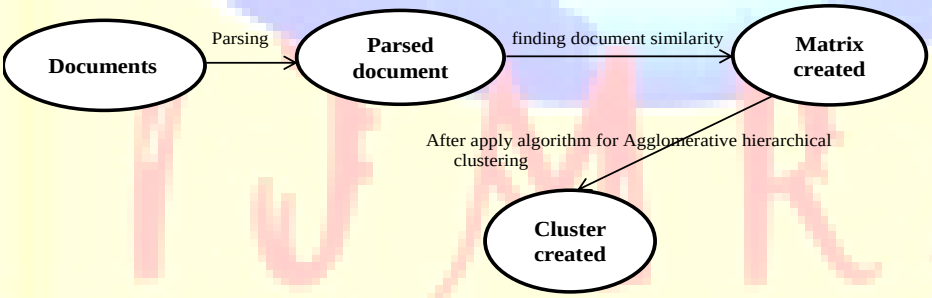


Figure 3: work flow implementation show in this paper

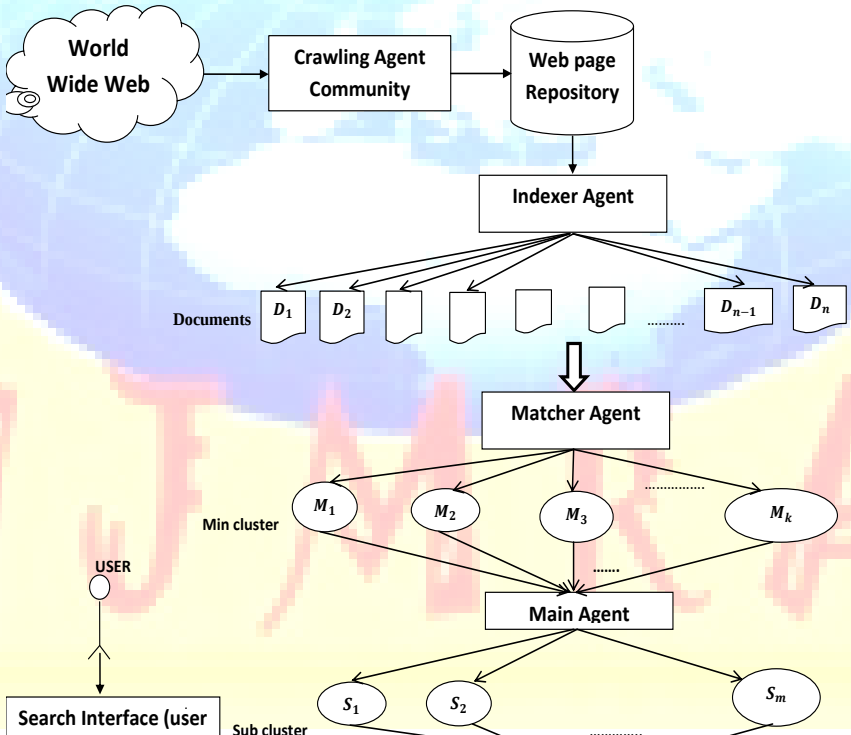
Let D = [D_1, D_2, . . . ,D_N] be a collection of N textual documents to which consecutive integer doc ids d = 1, 2, 3,. . . ,N are initially assigned. Moreover, let T be the number of distinct termst_i; i = 1, . . . ,T present in the documents, and t is the average length of terms. The total size CSize (D) of an IF index for D can be written as:

$$\text{CSize (D)} = \text{CSize}_{\text{lexicon}}(T \cdot \mu_t) + \sum_{i=1}^{T} \text{Encodem (d\_gaps } (t_i))$$

where $\mathbf{CSize_{lexicon}\ (T \cdot \mu_t)}$ is the number of bytes needed to code the lexicon, while d gaps (t_i) is the d_gap representation of the posting list associated to term t_i, 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 4.



Term	No. of doc. Containing term	Doc. Ids
Job	15	3, 7, 12, 13...
Engineer	3	1, 6, 9…
Fresher	5	2,3,8,11,14

Figure 3: Architecture of Agglomerative Hierarchical Clustering based indexing in Search Engines.

Algorithm for checking the document similarity: Let D=[D_1, D_2, . . . ,D_n] be the collection of N textual documents being crawled to which consecutive integers document identifiers 1…N are assigned. Each document D_i can be represented by a corresponding set S_i such that S_i is a set of all the terms contained in D_i. Let us denote that set by D* such that D*= [S_1, S_2, . . . ,S_n]. The similarity of any two documents S_i and S_j can be computed using the check_similarity (S_i ,S_j) function given below:

INPUT – The set D*= [S_1, S_2, . . . ,S_i], where S_i is a set of all the terms of document D_i.

–The 'k' number 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 check_similarity algorithm given below.

```
Algorithm doc_similarity (D1, D2, . . . ,Dn)
for (i = 1 to n)
{
        doc_similar[i][i]=0;     // initialize the document similarity matrix.
        for (j=i+1 to n)
        {
                doc_similar[i][j]=check_similarity(Si,Sj);
                doc_similar[j][i]= doc_similar[i][j];
        }
}
Algorithm check_simiilarity (Si , Sj)
{
        Counter=0;
        // where Si and Sj are the ith and jth  document.
        Parse (Si);
```

```
        Store keywords of document S_i in List_i;
        While (List_i! = NULL)
        {
                Token=List_i-> keyword;
                File *fp=fopen (S_[i+1] , "+R")
                While (fp! = eof)
                {
                        found = string_matcher (token, fp);
                        if (found != 1)
                        {
                        List_i=List_i->next;
                        }
                        else
                        {
                                counter=counter + 1;
                                List_i=List_i -> next;
                        }
                }
                Rewind (fp);
        }
}
```

Algorithm for string matching[15]:

```
String_matcher(token , fp)
{
        n=length[token];
        m=length[fp];
        for s=0 to n-m
        {
                If fp[1……..m]= token[s+1…….s+m] then
                        return 1;
                else
```

```
                return 0;
        }
}
```

Algorithm for document clustering: The Agglomerative hierarchical clustering algorithm which groups together the similar documents in cluster is given below:

```
Algorithm docum_clustering ( )
{
        i=1;
        CE=0;                   // initially we take variable 'CE=0' which denote cluster is
empty.
        for (c=1 to k)          // k number of clusters.
        {
        For (f=1 to n/k)    //  (n/k) no. of document cluster contain.
                {
                        For (j=1 to n)
                        {
                                Select max from sim[i][j];
                                CE=CE U S_i;
                                D*=D*- S_i;
                                for (i= 1 to n)
                                {
                                        sim[i][1]=0;
                                        sim[1][i]=0
                                }

                i=j
                        }
                }
        }
}
```

The algorithm starts with the first cluster which is empty initially. The first document from the document collection is considered and put the first document 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 means we compare the content of first document with document (2, 3, 4……N). 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.

Algorithm for cluster similarity: The algorithm that computes the similarity matrix for the similar clusters is given below. In this algorithm, D= [S_1,S_2,…….. S_n], where S_i is a set of terms in the cluster C_i.

```
Algorithm cluster_similarity ( )
{
        for (i = 1 to k)
        {
                similarity[i][i] = 0;
                for (j = i+1 to k)
                {
                similarity[i][j] = check_similarity (Si, Sj);
                similarity[j][i] = similarity[i][j];
                }
        }
}
```

Algorithm for sub clustering: The Agglomerative hierarchical clustering algorithm that goal at forming the sub cluster out of the similar min clusters is given below:

```
Algorithm subcluster( )
{
        i=1
        for (f=1 to m)
```

```
{
SC = 0;                    // initially sub cluster (SC) is empty.
for (e = 1 to k/m)         // where k no. of min cluster & m is the no of mega cluster.
{
        for (j = 1 to k)
        {
                select max from sim [i][j]
                SC = SC U Si;
                D= D- Si;
                for (i=1 to k)
                {
                        sim [1][i] = 0
                        sim [i][1] = 0
                }
        i=j
        }
}
}
```

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

5. CONCLUSION

In this paper, an efficient algorithm for computing a reordering of a collection of web documents has been presented that effectively enhances the compressibility of the IF index built over the reordered collection. Further, the proposed Agglomerative hierarchical clustering algorithm goal 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.

The algorithm produces better ordering of index size compression, reduction in search time and fast retrieval of relevant documents.

References

[1] FabrizioSilvestri, RaffaelePerego and Salvatore Orlando, "Assigning Document Identifiers to Enhance Compressibility of Web Search Engines Indexes", in the proceedings of SAC, 2004.

[2] M. Lopez - Sánchez, F. Martin, J. Garcia, X. Canals, X. Drudis, N. Ruiz, and A. Reyes, "Agent Communication within a Search Engine Architecture"

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

[4] A.K. Jain, M. N. Murty and P. J. Flynn, "Data Clustering: A Review, Acm Computing Surveys", 31(3):264-323, Sep 1999.

[5] P. Willet, "Recent Trends in Hierarchical Document Clustering: A Critical Review", Information Processing and Management, 24: 577-97, 1988.

[6] D. Fasulo, "An analysis of recent work on clustering algorithms," Department of Computer Science and Engineering, University of Washington, Tech. Rep. # 01-03-02, 1999. [Online]. Available: <citeseer.nj.nec.com/fasulo99analysi.html>Stanford University, Stanford, CA 94305, USAsergey@cs.stanford.edu and page@cs.stanford.edu.

[7] Stephen P. Borgatti: "How to explain hierarchical clustering"http://www.analytictech.com/networks/hiclus.htm.

[8] D. R. Cutting, D. R. Karger, J. O. Pedersen and J. W. Tukey, "Scatter/Gather: Cluster- Based Approach To Browsing Large Document Collections", in Proceedings of the 15th InternationalAcmSigirConference on Research and Development in Information Retrieval, Pages 318-29, 1992.

[9] Chang Shang Zhou, Wei Ding and Na Yang, "Double Indexing Mechanism of Search Engine based on Campus Net", Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06), 2006.

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

[11] Sanjiv K. Bhatia, "Adaptive KMeans Clustering", American Association for Artificial Intelligence, 2004.

[12] SoumenChakrabarti, Martin Van den Berg, Byron Dom "Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery", The Eighth International World Wide Web Conference, May 11-14, 1999.

[13] Dan Bladford and Guy Blelloch. "Index compression through document reordering" In IEEE, editor, Proc. Of DCC'02. IEEE, 2002.

[14] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hyper textual Web Search Engine", Computer Science Department, Stanford University, Stanford, CA 94305, USA.

[15] Cormen, T. H., Leiserson, C. E., and Rivest, R. L..,"Introduction to algorithms", 6th ed. MIT Press and McGraw-Hill Book Company, 1992.