# Query Based Duplicate Data Detection on WWW

Ranjna Gupta [#1], Neelam Duhan [*2], A.K. Sharma [*3], Neha Aggarwal [#4]

[#] *Computer Science & Engineering*
*B.S.A. Institute of Technology & Management*
*Faridabad, India*
Ranjna.gupta@gmail.com
Aggarwalneha2k@gmail.com

**Department of Computer Engineering*
*Y.M.C.A. University of Science &Technology*
*Faridabad, India*
Neelam.duhan@gmail.com

**Abstract— The problem of finding relevant documents has become much more prominent due to the presence of duplicate data on the WWW. This redundancy in results increases the users' seek time to find the desired information within the search results, while in general most users just want to cull through tens of result pages to find new/different results. The identification of similar or near-duplicate pairs in a large collection is a significant problem with wide-spread applications. Another contemporary materialization of the problem is the efficient identification of near-duplicate Web pages. This is certainly challenging in the web-scale due to the voluminous data. Therefore, a mechanism needs to be introduced for detecting duplicate data so that relevant search results can be provided to the user. In this paper, architecture is being proposed that introduces methods that run online as well as offline on the basis of favored and disfavored user queries to detect duplicates and near duplicates.**

***Keywords-WWW; Query log; Cluster; Search Engine; Ranking Algorithm***

## I. INTRODUCTION

The development of Internet has resulted in flooding of numerous copies of web documents in the search results making them futilely relevant to the users thereby creating a serious problem for internet search engines. The outcome of perpetual growth of Web and e-commerce has led to an increased demand of new Web sites and Web applications. The tremendous volume of web documents poses challenges to the performance and scalability of web search engines. Duplicate is an inherent problem that search engines have to deal with.

It has been reported that about 10% hosts are mirrored to various extents in a study including 238,000 hosts [1]. Consequently, many identical or near-identical results would appear in the search results if search engines do not solve this problem effectively. Such duplicates will significantly decrease the perceived relevance of search engines. Therefore, automatic duplicate documents detection is a crucial problem in front of search engines. "Duplicate documents"refer not only to completely identical documents but also to nearly identical documents.

In this paper, an approach has been proposed that detects duplicates and near duplicates using offline as well as online techniques so that relevant search results can be displayed to the users. The paper has been organized as follows: section II describes the current research that has been carried out in this area; section III illustrates the proposed work to detect duplicate web pages based on query clusters formed by using query logs; section IV shows the performance of proposed work and last section concludes the proposed work.

## II. RELATED WORK

The notion of Duplicate data detection has been a subject of interest since many years. A number of researchers have discussed the problem of finding relevant search results from the search engines.

A technique for estimating the degree of similarity among pairs of documents was presented in 1997 [2], known as *shingling*, does not rely on any linguistic knowledge other than the ability to tokenize documents into a list of words, i.e., it is merely syntactic. In this, all word sequences (shingles) of adjacent words are extracted. If two documents contain the same set of shingles they are considered equivalent and if their sets of shingles appreciably overlap, they are exceedingly similar.

A new approach that performs copy detection on web documents [3] determines the similar web documents, similar sentences and graphically captures the similar sentences in any two web documents. Besides handling wide range of documents, this copy detection approach is applicable to web documents in different subject areas as it does not require static word lists.

A novel algorithm, Dust Buster, for uncovering DUST (Different URLs with Similar Text) [4] has also been proposed in the literature. This method intended to discover rules that transform a given URL to others that are likely to have similar content. Dust Buster employs previous crawl logs or web server logs instead of probing the page contents to

mine the dust efficiently. It is necessary to fetch few actual web pages to verify the rules via sampling.

However, a critical look at the available literature [2,3,4] indicates that although there are many efficient methods available for duplicate data detection but they are not much scalable due to the abundance of web pages on WWW. Therefore, a mechanism needs to be introduced for overcoming the problem of scalability and efficiently detect duplicates. A framework for duplicate data detection on the basis of favored and disfavored queries is proposed, wherein the offline method uses Query log that keeps record of user queries to identify favored query on the basis of occurrence of query in the query cluster which is formed by clustering similar queries on the basis of keywords and clicked URLs. These favored queries are run offline to detect duplicates so that throughput and performance of search engine is not affected. On the contrary, the online method focuses on removing duplicated pages in the search results at run time.

## III. PROPOSED WORK

The proposed method of *Query based Duplicate Data Detection on WWW* introduces a technique to detect duplicate and near duplicate web pages. The proposed architecture (shown in Fig.1) works offline for favored queries as well as online for disfavored queries.

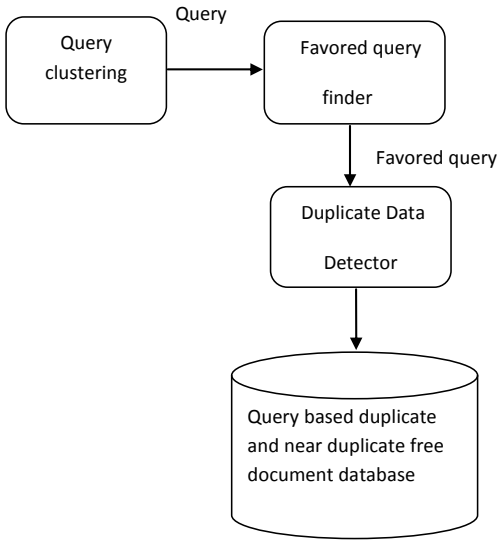


Fig.1. Architecture of duplicate data detection [Offline]

### *A. Architecture of duplicate data detection [Offline]*

Initially Query clustering tool produces query clusters and then with the help of favored query finder, favored queries are identifies from the clusters. These favored queries are then executed offline and at last duplicate and near duplicate documents are extracted from the retrieved results corresponding to favored query and stored to query-document database.

The proposed architecture that works offline consists of the following functional components:

1) Query Clustering Tool
2) Favored Query Finder
3) Duplicate data Detector

The working of these component modules is explained below.

*1) Query Clustering Tool:*

This tool is used to cluster user queries using query logs built by search engines and for this it assigns query log entries to cluster generator, which in result produces query clusters as shown in Fig.2.

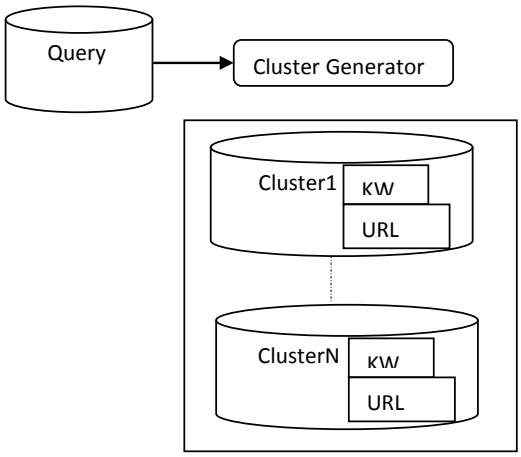


Fig.2 Query Clustering Tool (KW: Keywords).

It uses cluster generator method that works on the following principles:

***Principle 1 (using query contents):*** If two queries contain the same or similar terms, they denote the same or similar information needs. Keyword-based similarity function is defined as follows:

$$Sim_{keyword}(qi, qi+1) = \frac{KW(qi) \cap KW(qi+1)}{KW(qi) \cup KW(qi+1)} \quad (1)$$

where *KW*(q) is the number of keywords in a query.

***Principle 2 (using document clicks):*** If two queries lead to the selection of the same documents (which is known as document clicks), they are similar. ClickedURL-based similarity function is defined as follows:

$$Sim_{clickedurl\ s}(qi, qi+1) = \frac{URL(qi) \cap URL(qi+1)}{URL(qi) \cup URL(qi+1)} \quad (2)$$

where URL(q) is the number of clicked Documents.

These two principles have their own advantages. Therefore, a combined measure is defined to take advantage of both principles which is defined as follows:

$$Sim_{combined} = \alpha * Sim_{keywords} + \beta * Sim_{clickedurl} \quad (3)$$

where α and ß are constants with their values between 0 and 1. The working of cluster generator method is defined in Fig.3.

**Initial requirement:**

1. **Query log** with the following fields:
   a. **IP address** of the users.
   b. **QueryID:** each query is assigned a unique ID
   c. **Query:** query issued by the user.
   d. **Item rank:** if the user clicked on such result, then rank of the item on which they clicked is listed.
   e. **Clicked URL:** if the user clicked on search result, the domain position of the URL in the clicked result is listed.
2. Assign each query a flag and set its value equal to 0 initially.
3. Parse the query and generate its tokens

**Assumptions:**

- Each cluster would contain three data objects:
  1. a cluster-ID
  2. a collection of related queries on the basis of clicked-URLs and keywords
  3. a set of clusterkeywords(clustKW) and clusterURL (clustURL)
- Initially set of cluster keywords and cluster URL strings are set to NULL

**Algorithm:**

For each query $q_i$
{
//sure that qi is not a part of any existing cluster
If flag ($q_i$) = =1 then
break;
else
//create a new cluster with respect to query $q_i$

$$clustURL = clustURL \cup url(q_i)$$

$$clustKW = clustKW \cup token(q_i)$$

For each query $q_{i+1}$
{// find the similarity score of two queries on the basis of clicked URL and keywords

$$\sigma(qi,qi+1) = \alpha * Sim_{keywords}(qi,qi+1) + \beta * Sim_{clickedurl}(qi,qi+1)$$

if ($\sigma$ ($q_i$,$q_{i+1}$)> = threshold value) then
add query $q_{i+1}$ to cluster and append its related information in the cluster and set its flag($q_{i+1}$)=1
} }

**Figure 3. Cluster Generator**

*2) Favored Query Finder:*

Once query clusters are formed, next step is to find a set of favored and disfavored queries from each cluster. A query is said to be favored query that occupy a major portion of the whole search request in a cluster and it is assumed that their corresponding search results have more duplicates in search results than disfavored queries. Therefore, their respective search results are processed offline so that it could not affect the response time and throughput of search engine. The process of finding favored query is shown in Fig.4.

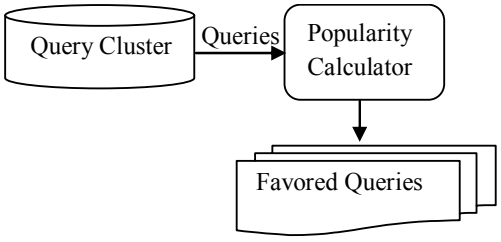


Fig.4. Favored Query Finder

A popularity calculator discussed below in Fig. 5 is used to determine whether a query is favored or not.

**Initial Requirement:**

a. Queries in the cluster.

**Algorithm: Popularity Calculator**

1. Sort the queries in alphabetical order in the cluster.
2. Queries which are exactly same club them and make a set of their corresponding IP addresses.
3. calculate the wt of query as:

$$Wt = \frac{\text{No. of IP addresses which fired the query}}{\text{Total no. of IP addresses in that cluster}}$$

4. if calculated Wt.>= threshold value then
   Query is considered as favored query
   else
   query is considered as disfavored.

Fig.5. Popularity Calculator

*3) Duplicate Data Detector:*

Once favored queries from their query clusters are identified, next step is to retrieve a set of search results and remove duplicates and near-duplicate documents from the results.

The process of detecting duplicate documents is shown in Fig. 6.

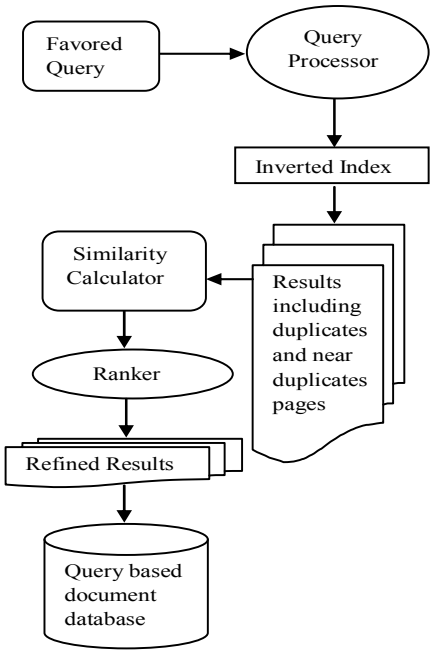


Fig. 6. Duplicate data detector

The query processor includes a simple GUI that takes favored query as a list of terms, retrieves a ranked list of documents that are deemed relevant to the query. Many scoring functions or similarity measures exist which, for a given query-document combination, compute the relevance of the query to the document. The retrieved documents from the query processor include both duplicate and near-duplicate documents which are not useful for the users. Therefore, this

result is given to the similarity calculator, which uses (4) for finding the similarity between the two documents.

$$SS(D_i, D_j) = \frac{\sum_{k=1}^{n} KW_{k,ti} * KW_{k,tj}}{\text{length of doc } i * \text{length of doc } j} \quad (4)$$

where ti and tj represent the tables containing tokens of documents Di and Dj respectively. The numerator gives the summation of products of term frequencies of common tokens/keywords (which is n in number) in ti and tj. The length of a document i can be calculated by (5)

$$length\ of\ doc\ Di = \sqrt{\sum_{k=1}^{n} w_{i,k}^2} \quad (5)$$

where k represent to the tokens in document i. The length is calculated by summation of products of term frequencies of tokens/keywords (which is n in number) in document *di*.

The functionality of Similarity Calculator is shown in Fig 7.

**Initial Requirement:**

**a. Results corresponding to the user query.**

**Algorithm: Similarity Calculator**

**1)** Perform steps 2 to 4 for each retrieved document.
**2)** Perform steps 3 to 5 for each retrieved document.
**3)** Parse the retrieved documents.
**4)** Remove Stop words from the retrieved documents.
**5)** Apply Stemming algorithm.
**6)** Assign a set Si corresponding to each retrieved document di that will be used to store similar documents. Initially set Si={di}.
**7)** for each document di
**8)** { j=i+1
**9)** for each document dj
**10)** **{**Calculate Similarity score between di,dj using following formula

$$SS(D_i, D_j) = \sum_{i=1}^{n} \frac{KW_{i,t1} * KW_{i,t2}}{\text{length of doc } i * \text{length of doc } j}$$

**11)** if SS(di,dj)>= threshold value then
**12)** **{** $Si = Si \cup dj$

$Sj = Sj = di$

**}}}**

Fig.7. Similarity Calculator

The output will come out in the range of [0, 1] and if the result is above a predefined threshold value then documents are considered as near duplicates. This module stores the refined results into the query-document database after removing all but one out of the similar pages on the basis of retaining the page with higher page rank[14].

### *B. Architecture of Query based Duplicate data detection [Online]*

Due to the huge size of WWW, it is not possible to detect duplicates and near duplicate web documents offline, therefore in order to enhance the scalability of this process, duplicate data detection is also performed online. The online methods focus on removing duplicated pages in the search results at run time.

The online method for duplicate data detection is shown in Fig 8.

When user fires a query via GUI, that query first matched

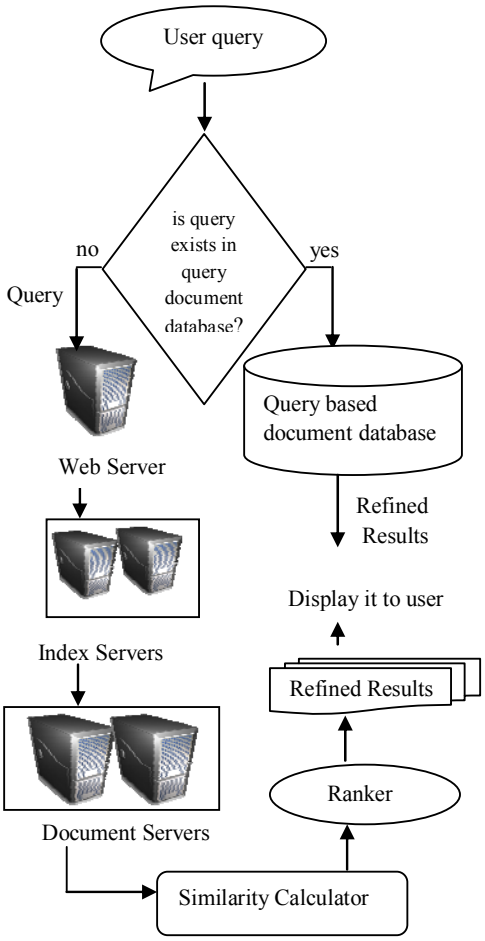


Fig. 8. Duplicate data detection [Online]

in query based document database to ensure if its corresponding duplicate data has been already refined offline or not. If it returns true then it directly retrieve results from query based document database and in turn display it to end-user otherwise query is first sent to the web server. The web server sends the query to index servers. The content inside the index servers is similar to index in the back of a book- it tells which page contains thee words that match any particular query term. After this, query travels to the document server, which actually retrieves the stored documents.

Once the documents are retrieved from document server, a similarity calculator is applied over these documents to detect duplicates and near duplicate web pages. And then with the help of ranker which determines the ranking of the web pages, documents with highest rank factor are selected from duplicate ones and finally displayed to the user.

### *C. Incremental Query Clustering*

Query clustering tool as discussed in section *A* (1) is applied to a static query log, however, the query log may have frequent new entries and thus may be rather dynamic. And it may also happen that previous disfavored queries in the cluster may become favoured queries after insertion of these new queries. Therefore, there is a need for incremental

updates of a clustering after additions of new queries in the query log.

The architecture of incremental query clustering is shown in Fig.9 and its corresponding functioning is explained in Fig. 10.

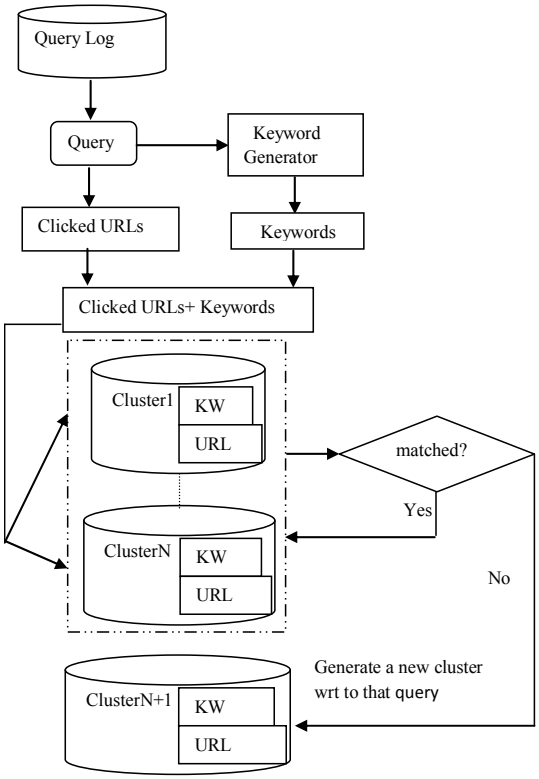


Fig.9. Architecture of Incremental clustering of query log

**Initial Requirement: Queries from Query log which are not a part of any cluster.**
**Algorithm: Process of Incremental clustering of query log**

**Step1:** Parse, remove stop-words from the query using keyword generator and generate its corresponding keywords.

**Step2:** Retrieve its corresponding clickedURLs from the log .

**Step3:** Compare its keywords and clickedURL against each cluster until all the clusters are not compared.

a. If query matches with any cluster's keyword and URLcluster set then assign that query into respective cluster.

**Step4:** Otherwise a new cluster wrt to that query will be formed dynamically using previous discussed clustering algorithm in section *A*(1).

Fig.10. Process of Incremental clustering of query log

## IV. EXPERIMENTAL RESULTS

The results of experiments are presented in this section. The keywords extracted from the web documents are stored in MS Access. The near duplicates have been detected efficiently by the proposed approach. This has been achieved with the aid of the similarity scores. The similarity score would find all pairs of web pages whose duplications are identified by a given predefined threshold. Upon considering the similarity scores, a lower distance measure indicates that the documents are more similar and hence are regarded as near duplicates.

If the extracted keywords from the two web documents are almost same, then it is considered as near duplicate and their distance is of minimum value. One such example for near duplicate web documents, the extracted keywords and their calculated similarity scores are shown in table1:

For this purpose, the query fired is given below.

**Query: CBI query sought in Nirupama Pathak Murder Case.**

Corresponding to this query, sample selected URLs are given below

**URL1:http://timesofindia.indiatimes.com/india/Ambiguities-in-Nirupama-autopsy-report-AIIMS-expert/articleshow/5910173.cms**

**URL2:http://kraran.com/nirupama-pathak-autopsy-report-has-ambiguities/**

**URL3:http://beta.thehindu.com/news/national/article425770.ece**

Similarity Score between doc1, doc2 and doc3 using similarity calculator discussed in Figure 7 are shown in Table 1.

TABLE 1 SIMILARITY SCORE VALUES FOR SAMPLE DOCUMENTS

| | **Doc1** | **Doc2** | **Doc3** |
|---|---|---|---|
| **Doc1** | **1** | **0.824868** | **0.868965** |
| **Doc2** | **0.824868** | **1** | **0.813696** |

Consequently this calculated Similarity Score value is compared with the predefined threshold value. Based on this, it is concluded that the URL 1, URL2 and URL 3 are near duplicates.

## V. CONCLUSION

In this paper, Architecture of query based duplicate data detection has been proposed that not only detects duplicate data offline from existing query log but also add new queries into existing query clusters using Incremental query clustering architecture so that query clusters need not to be generate periodically from the scratch. It also works for disfavored queries at runtime; therefore it works offline as well as online. As the future work, the architecture of a search engine based on query based duplicate data detection can be designed.

## REFERENCES

[1] Bharat, K., Broder, A.Z.: Mirror, mirror on the Web: A study of host pairs with replicated content. In: Proceedings of the 8th International World Wide Web Conference (WWW). (1999) 501-512

[2] Broder, A., Glassman, S., Manasse, M., and Zweig, G., 1997. "Syntactic Clustering of the Web", In 6th International World Wide Web Conference, pp: 393-404.

[3] Yerra, R., and Yiu Kai, NG., 2005. "A sentence-Based Copy Detection Approach for Web Documents", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Vol. 3613, pp. 557-570.

[4] BarYossef, Z., Keidar, I., Schonfeld, U., 2007. "Do Not Crawl in the DUST: Different URLs with Similar Text", 16th International world Wide Web conference, Alberta, Canada, Data Mining Track, 8-12 May.
[5] Manku, G. S., Jain, A., Sarma, A. D., 2007. "Detecting near-duplicates for web crawling," Proceedings of the 16th international conference on World Wide Web, pp: 141 – 150.
[6] Henzinger, M., 2006. "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM Press, pp. 284-291.
[7] Zamir, O., Etzioni, O., 1998. "Web document clustering: A feasibility demonstration", In: Proceedings of the 21st Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 46-54.
[8] Broder, A., Glassman, S., Manasse, M., and Zweig, G., 1997. "Syntactic Clustering of the Web", In 6th International World Wide Web Conference, pp: 393-404.
[9] Fetterly, D., Manasse, M., Najork, M., 2003. "On the Evolution of Clusters of Near Duplicate Web Pages", Proceedings of the First Conference on Latin American Web Congress, pp. 37.
[10] Ilyinsky, S., Kuzmin, M., Melkov, A., Segalovich, I., 2002. "An efficient method to detect duplicates of Web documents with the use of inverted index", Proceedings of the Eleventh International World Wide Web Conference.
[11] Yerra, R., and Yiu Kai, NG., 2005. "A sentence-Based Copy Detection Approach for Web Documents", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Vol. 3613, pp. 557-570.
[12] Andrei Z. Broder., 2000. "Identifying and Filtering Near-Duplicate Documents", Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching. UK: Springer-Verlag, pp.1-10,.
[13] V.A. Narayana, P. Premchand and A. Govardhan,2009." Effective Detection of Near Duplicate Web Documents in Web Crawling", ", Proceedings of International Journal of Computational Intelligence Research, ISSN 0973-1873 Volume 5, Number 1 (2009), pp. 83–96
[14] Lawrance page and sergey Brin.The Page Rank Citation ranking: Bringing order to the web. In to appear: Proceedings of seventh web conference(WWW 98),1998.

**Ranjna Gupta** received B.E. degree in Computer Science & Engineering with Hons. from Maharshi Dayanand University in 2005 and is persuing M.Tech. in Computer. Presently, she is working as Senior Lecturer in Computer Engineering department in B.S.A. Institute of Technology & Management, Faridabad. Her areas of interests are Search Engines, Web Mining.

**Neelam Duhan** received B.Tech degree in Computer Science Engineering with Hons. from Kurukhetra University and M.Tech degree with hons. in computers from Maharshi Dayanand University in 2002 and 2005 respectively. Presently, she is working as Lecturer in Computer Engineering Department in YMCA University of Science & Technology, Faridabad. She is also persuing Ph.D in Computer Engineering and her areas of interests are Databases and Web Mining.

**Prof. A. K. Sharma** received his M.Tech. (Computer Science & Technology) with Hons. from University of Roorkee in the year 1989 and Ph.D (Fuzzy Expert Systems) from JMI, New Delhi in the year 2000. From July 1992 to April 2002, he served as Assistant Professor and became Professor in Computer Engg. at YMCA University of Science & Technology, Faridabad in April 2002. He obtained his second Ph.D. in IT from IIIT & M, Gwalior in the year 2004. His research interests include Fuzzy Systems, Object Oriented Programming, Knowledge representation and Internet Technologies.

**Neha Aggarwal** received B.E. degree in Information technology with Hons. from Maharshi Dayanand University in 2007 and is persuing M.Tech. in IT. Presently, she is working as Lecturer in Computer Engineering department in B.S.A. Institute of Technology & Management, Faridabad. Her areas of interests are Data Mining, Search Engine & Web Mining.