International Journal of Computer Applications (0975 – 8887)
Volume 17– No.8, March 2011

A Novel Technique for Database Selection and Document Selection

Anil Agrawal
Ambalika IMT Lucknow
UP, INDIA

Mohd. Husain
AZAD IET, Lucknow
UP, INDIA

Raj Gaurang Tiwari
AZAD IET, Lucknow
UP, INDIA

Subodh Kumar
Ambalika IMT, Lucknow
UP, INDIA

ABSTRACT

The Internet has become a cosmic information source in recent years and can be considered as the world's largest digital library. To aid ordinary users in finding desired data in this library, numerous search engines have been created. Each search engine has a corresponding database that defines the set of documents that can be searched by the search engine. Typically, an index for all documents in the database is created and stored in the search engine. Text data in the Internet can be partitioned into numerous databases naturally. Proficient retrieval of desired data can be realized if we can accurately envisage the usefulness of each database, because with such information, we only need to retrieve potentially useful documents from useful databases. For a given query 'q' the usefulness of a text database is defined to be the no. of documents in the database that are sufficiently relevant to the query 'q'.

In this paper, we propose innovative approaches for database selection and documents selection.

General Terms

Measurement, Performance, Design, Experimentation

Keywords

Metasearch Engine, Distributed query processing, Document selection

1. INTRODUCTION

Information retrieval (IR) is the discipline of searching for documents, for information within documents, and for metadata about documents, as well as that of searching relational databases and the World Wide Web. There is overlap in the treatment of the terms data retrieval, document retrieval, information retrieval, and text retrieval, but each also has its individual body of literature, theory, praxis, and technologies. IR is interdisciplinary, rooted in computer science, mathematics, library science, information science, information architecture, cognitive psychology, linguistics, and statistics. Automated information retrieval systems are used to diminish what has been called "information overload". Many universities as well as public libraries exploit IR systems to provide access to books, journals and other documents. Web search engines are the most noticeable IR applications.

An information retrieval process instigates when a user enters a query into the system. Queries are formal statements of information needs, for instance search strings in web search engines. In information retrieval a query does not inimitably recognize a single object in the collection. Instead, numerous objects may match the query, possibly with different degrees of relevancy[6]. An object is an entity that is characterized by information in a database. User queries are matched against the database information. Depending on the application the data objects may be text documents, images, or videos. Often the documents themselves are not kept or stored directly in the IR system, but are instead represented in the system by document surrogates or metadata. Most IR systems compute a numeric score on how well each object in the database match the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

Over the years, Information Retrieval has experienced an enormous Change due to the surge in World Wide Web and the advent of modern, inexpensive user interfaces and large storage systems. Evaluation is at the crux of Information Retrieval. It investigates the various attributes like user satisfaction, system effectiveness (observed by the ordering of the retrieved list of documents), time efficiency etc.

Information retrieval systems are everywhere: Web search engines, library catalogs, store catalogs, cookbook indexes, and so on. Information retrieval (IR), also called information storage and retrieval (ISR or ISAR) or information organization and retrieval, is the art and science of retrieving from a collection of items a subset that serves the user's purpose.

As he Internet has become a vast information source in recent years, to help ordinary users find desired data in the Internet, many search engines have been created. Each search engine has a corresponding database that defines the set of documents that can be searched by the search engine. Usually, an index for all documents in the database is created and stored in the search engine. For each term which represents a content word or a combination of several (usually adjacent) content words, this index can identify the documents that contain the term quickly. The pre-existence of this is critical for the search engine to answer user queries efficiently.

Two types or search engines exist. General-purpose search engines attempt to provide searching capabilities for all documents in the Internet or on the Web. WebCrawler[10], HotBot, Lycos and Alta Vista are a few of such well-known search engines. Special-purpose search engines, on the hand, focus on documents in confined domains such as documents in an organization or of a specific interest. Tens of thousands of special-purpose search engines are currently running in the Internet.

The amount of data in the Internet is huge (it is believed that by the end of 2010, there were more than 30000 million web pages and is increasing at a very high rate. Many believe that employing a single general-purpose search engine for all data in the Internet is unrealistic [12]. First, its processing power and storage capability may not scale to the fast increasing and virtually unlimited amount of data. Second, gathering all data in the Internet and keeping them reasonably up-to-data are extremely difficult if not impossible. Programs (i.e. Robots) used by search engines to gather data automatically may slow down local servers and are increasingly unpopular.

A more practical approach to providing search services to the entire Internet is the following multi-level approach. At the bottom level are the local search engines. These search engines can be grouped, say based on the relatedness of their database, to form next level search engines (called metasearch engines). Lower, level metasearch engines can themselves be grouped to form higher level metasearch engines[5]. This process can be repeated until there is only one metasearch engine at the top. A metasearch engine is essentially an interface and it does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain information about the contents of the (meta) search engines at a lower level to provide better service. When a metasearch engine receives a user query, it first passes to the appropriate (meta) search engines at the next level recursively until real search engines are encountered, and then collects (sometimes, reorganizes) the results from real search engines, possible going through metasearch engines at lower levels. A two-level search engine organization is illustrated in Figure 1.

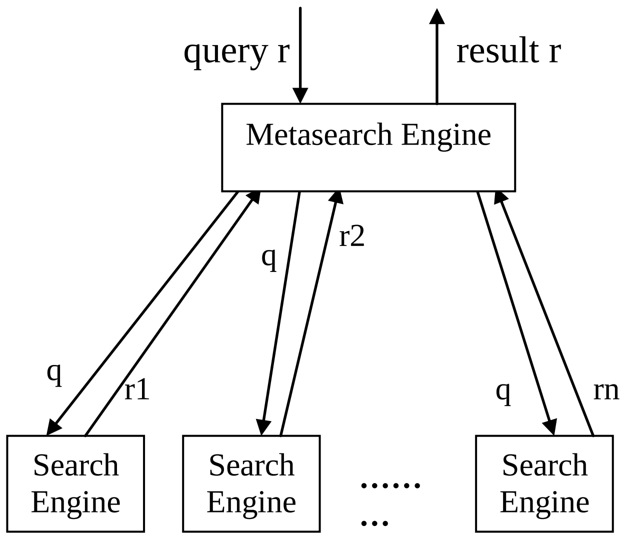


Figure 1: Two-Level Search Engine Organization

The advantages of this approach are

(a) User queries can (eventually) be evaluated against smaller databases in parallel, resulting in reduced response time;
(b) updates to indexes can be localized, i.e., the index of a local search engine is updated only when documents in its database are modified; (Although local updates may need to be propagated to upper level metadata that represent the contents of local databases, the propagation can be done infrequently as the metadata are typically statistical in nature and can tolerate certain degree of inaccuracy.)
(c) Local information can be gathered more easily and in amore timely manner;
(d) The demand on storage space and processing power at each local search engine is more manageable. In other words, many problems associated with employing a single super search engine can be overcome or greatly alleviated when this multi-level approach is used.

When the number of search engines that cane be invoked by a metasearch engine is large, a serious inefficiency may arise. Typically, for a given query, only a small fraction of all search engines may contain useful documents to the query. As a result, if every search engine is blindly invoked for each user query, then substantial unnecessary network traffic will be created when the query is sent to useless search engines. In addition, local resources will be wasted when useless database are searched. A better approach is to first identify those search engines that are most likely to provide useful results to a given query and then pass the query to only these search engines for desired documents. A challenging problem with this approach is how to identify potentially useful search engines. The current solution to this problem is to rank all underlying databases in decreasing order of usefulness for each query using some metadata that describe the contents of each database. Often, the ranking is based on some measure which ordinary users may not be able to utilize to fit their needs. For a given query, the current approach can tell the user, to some degree of accuracy, which search engine is likely to be the most useful, the second most useful, etc. While such a ranking can be helpful, it cannot tell the user how useful any particular search engine is.

The main contribution of this paper is a set of novel algorithms aimed at improving the overall effectiveness of the web search process through the database and document selection processes.

In the first part of our work we present an algorithm DBSEL for database selection. This algorithm selects those databases from no. of databases which contain query ‘q’. This algorithm test each database with its documents stored in it. If any document of database contains the query ‘q’ at least one time then we select that database. If all the documents of database does not contains the query ‘q’ then that database will not be selected.

In the second part of our work we present an algorithm HighRelDoc for documents selection. This algorithm search all the selected databases and select only those documents from each database in which the query ‘q’ occurs at least one time. After that this algorithm ranks all the selected documents according to the no. of occurrence of query ‘q’ in descending order. Finally this algorithm returns the top ‘n’ most relevant documents from the sorted list of documents for any positive integer ‘n’.

2. DATABASE SELECTION AND DOCUMENT SELECTION PROBLEM

To help ordinary users find desired data from the Web, many search engines have been created. Each search engine has a text database that is defined by the set of documents that can be searched by the search engine[8]. In this paper, search engine and database will be used interchangeably. Usually, an inverted file index for all documents in the database is created and stored in the search engine. For each term which can represent a significant word or a combination of several (usually adjacent) significant words, this index can identify the documents that contain the term quickly.

Frequently, the information needed by a user is stored in multiple databases. As an example, consider the case when a user wants to find research papers in some subject area. It is likely that the desired papers are scattered in a number of publishers’ databases. Substantial effort would be needed for the user to search each database and identify useful papers from the retrieved papers. A solution to this problem is to implement a metasearch engine on top of many local search engines. A metasearch engine is a system that supports unified access to multiple existing search engines. It does not maintain its own index on documents. However, a sophisticated metasearch engine may maintain information about the contents of its underlying search engines to provide better service. When a metasearch engine receives a user query, it first passes the query to the appropriate local search engines, and then collects (sometimes reorganizes) the results from its local search engines. With such a metasearch engine, only one query is needed from the above user to invoke multiple search engines.

Building a metasearch engine is also an effective way to increase the search coverage of the Web. As more and more data are put on the Web at faster paces, the coverage of the Web by individual search engines has been steadily decreasing. By combining the coverages of multiple search engines, a metasearch engine can have a much larger coverage of the Web.

A closer examination of the metasearch approach reveals the following problems.

1. If the number of local search engines in a metasearch engine is large, then, it is likely that for a given query, only a small percentage of all search engines may contain sufficiently useful documents to the query. In order to avoid or reduce the possibility of invoking useless search engines for a query, we should first identify those search engines that are most likely to provide useful results to the query and then pass the query to only the identified search engines. Examples of systems that employ this approach include gGlOSS [1], Savvy Search [2], D-WISE [3], CORI Net [4]. The problem of identifying potentially useful databases to search is known as the database selection problem.
2. If a user only wants the n most similar documents across all local databases, for some positive integer n, then the n documents to be retrieved from the identified databases need to be carefully specified and retrieved. This is the document selection problem.

Both the problems are described in figure 2.

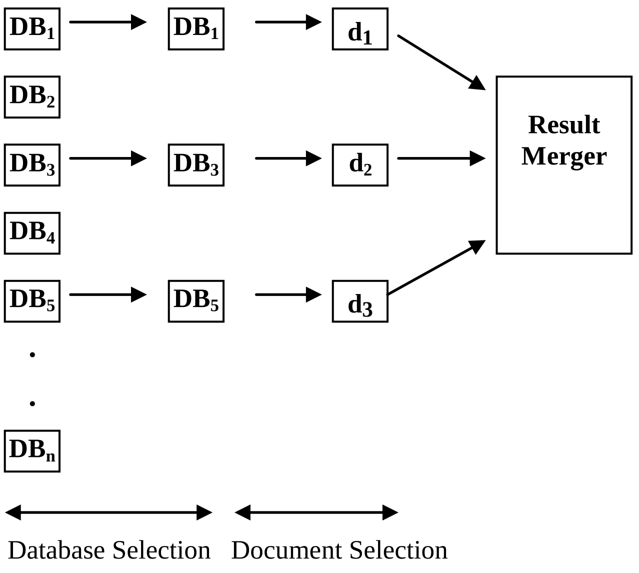


Figure 2: Database and document selection

The methodology that we propose to retrieve the n most relevant documents across multiple databases for a given query consists of the following two steps:

1. By using algorithm DBSEL we select those databases from number of databases which contain our query 'q'.
2. After databases selection we retrieve 'n' most relevant documents from the selected databases by using algorithm HighRelDoc.

3. AN ALGORITHM FOR DATABASE SELECTION

We want to select those databases from number of databases which contain our query 'q'. For this we proposed an Algorithm **DBSEL.** The Basic idea of this algorithm is that we test databases in the order DB_1, DB_2, DB_3, DB_4, B_5,………., DB_N, until we get the databases which contain the query 'q'. This algorithm works as follows:

1. Test each database with its documents stored in it. If any document of database contains the query 'q' at least one time then we select that database.
2. If all the documents of database does not contains the query 'q' then that database will not be selected.

DBSEL Algorithm

```
1.  Let  the'qlen' is the  length of query 'q';
2.  i = 1;
3.  while (i < = No. of Databases)
    {
        j=1, s=0;
        while (j < = No. of Documents in DB_i)
        {
(a) Let no. of occurrences of query 'q' in  j^th document   noc = 0;
(b) k=1;
(c) Obtain the length 'dlen' of j^th document;
        while (k < = dlen)
        {
          i.   Take the 'qlen' characters from j^th document starting from k^th position;
          ii.  Compare the query 'q' with these 'qlen' characters;
          iii. If both are equal then noc =noc + 1;
          iv.     k = k + 1;

        }
(d) Take the no. of occurrences of query 'q' in j^th document of i^th database
        dnoc [ i, j ] = noc;
        s = s + noc;
        j = j + 1;
        }
                if (s > 0) then
                {   Select i^th database SD[i] = DB_i;   }
        else
        {    i^th database will not be selected;   }
                i=i+1;
    }
```

4. AN ALGORITHM FOR DOCUMENTS SELECTION

After database selection we retrieve documents from the databases in the order DB_1, DB_2, DB_3, DB_4, DB_5, DB_N, until 'n' most relevant documents contained in the selected databases are obtained. For this we proposed an algorithm

HighRelDoc to retrieve documents from the selected databases. This algorithm works as follows:

1. We search all the selected databases in the order DB_1, DB_2, DB_3, DB_4, DB_5,………., DB_N. We select only those documents from each database in which the query ‘q’ occurs at least one time.
2. Rank all the selected documents according to the no. of occurrence of query ‘q’ in descending order.
3. Return the top ‘n’ most relevant documents from the sorted list of documents for any positive integer ‘n’.

HighRelDoc Algorithm

```
1.  i = 1,
2.  Let the total no. of selected documents t = 0;
3.  while( i < = No. of selected Databases)
    {
          j=1;
          while (j < = No. of documents in selected DB_i)
          {
          if (dnoc [i, j] > 0)
                {
          (a) Select the j^th document of i^th database
              Sdoc[ i , j ] = DB[i , j];

          (b) Take the no. of occurrences of query ‘q’ in selected j^th document of i^th database
              Sdnoc [i, j] = dnoc [i, j];

          (c) t=t+1;
                }
                j = j + 1;
          }
          i = i + 1;
    }
4.  Rank all the selected documents according to the no. of occurrence of query ‘q’ in descending order.
5.  Return the top ‘n’ most relevant documents from the sorted list of documents for any positive integer ‘n’.
```

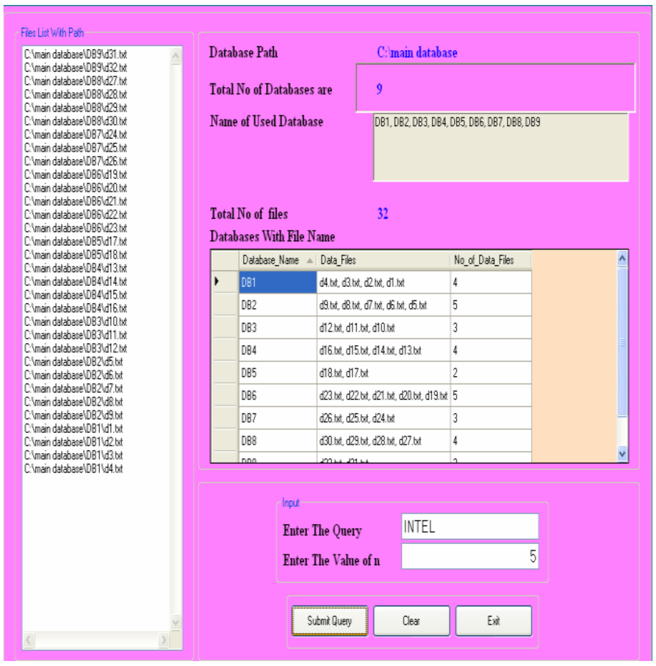
4. EXPERIMENTAL EVALUATION

Here we compare previous high-correlation method and OptDocRetrv algorithm with our DBSEL and HighRelDoc algorithms. Here, we compare the performance of the following estimation methods in retrieving the n most relevant documents for n = 5, 10 from the 9 databases.

1. The high-correlation method does not provide any detail on how a cutoff in database selection is chosen nor which documents are picked from each chosen database.
2. The previous OptDocRetrv algorithm retrieves documents from the databases, after the databases have been ranked.
3. Our DBSEL algorithm gives the cut off value while selecting the databases. Thus overhead incurred in processing the databases that are not related to query is minimized.
4. Our HighRelDoc algorithm selects the documents when all the documents of all selected databases have been ranked. That gives more correct results in comparison with the OptDocRetrv algorithm which retrieve documents from the databases, after the databases have been ranked.

5.1 Experimental Results

Our DBSEL and HighRelDoc algorithms were implemented in .Net Framework. The snapshots of our work are given below.



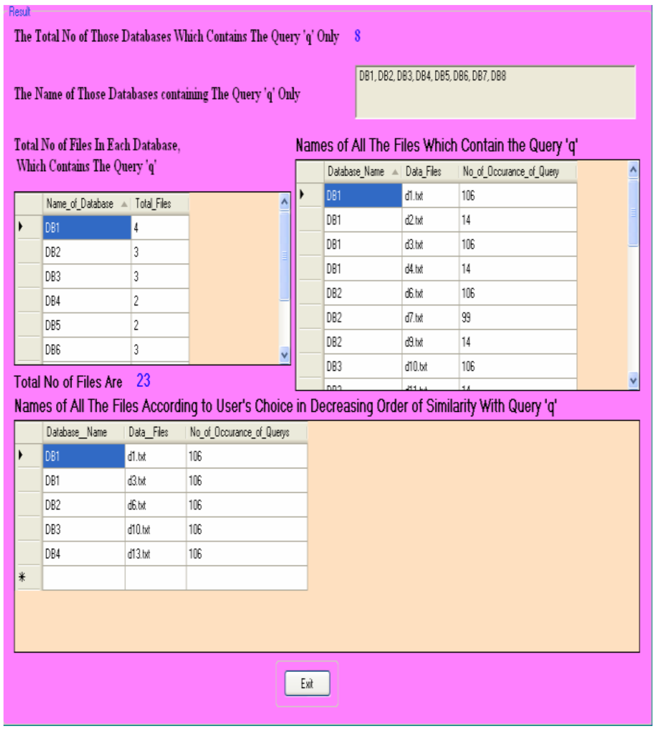
C:\main database\DB9\d31.txt
C:\main database\DB9\d32.txt
C:\main database\DB8\d27.txt
C:\main database\DB8\d28.txt
C:\main database\DB8\d29.txt
C:\main database\DB8\d30.txt
C:\main database\DB7\d24.txt
C:\main database\DB7\d25.txt
C:\main database\DB7\d26.txt
C:\main database\DB6\d19.txt
C:\main database\DB6\d20.txt
C:\main database\DB6\d21.txt
C:\main database\DB6\d22.txt
C:\main database\DB6\d23.txt
C:\main database\DB5\d17.txt
C:\main database\DB5\d18.txt
C:\main database\DB4\d13.txt
C:\main database\DB4\d14.txt
C:\main database\DB4\d15.txt
C:\main database\DB4\d16.txt
C:\main database\DB3\d10.txt
C:\main database\DB3\d11.txt
C:\main database\DB3\d12.txt
C:\main database\DB2\d5.txt
C:\main database\DB2\d6.txt
C:\main database\DB2\d7.txt
C:\main database\DB2\d8.txt
C:\main database\DB2\d9.txt
C:\main database\DB1\d1.txt
C:\main database\DB1\d2.txt
C:\main database\DB1\d3.txt
C:\main database\DB1\d4.txt

Database Path: C:\main database
Total No of Databases are: 9
Name of Used Database: DB1, DB2, DB3, DB4, DB5, DB6, DB7, DB8, DB9
Total No of files: 32
Databases With File Name

Database_Name	Data_Files	No_of_Data_Files
DB1	d4.txt, d3.txt, d2.txt, d1.txt	4
DB2	d9.txt, d8.txt, d7.txt, d6.txt, d5.txt	5
DB3	d12.txt, d11.txt, d10.txt	3
DB4	d16.txt, d15.txt, d14.txt, d13.txt	4
DB5	d18.txt, d17.txt	2
DB6	d23.txt, d22.txt, d21.txt, d20.txt, d19.txt	5
DB7	d26.txt, d25.txt, d24.txt	3
DB8	d30.txt, d29.txt, d28.txt, d27.txt	4

Enter The Query: INTEL
Enter The Value of n: 5

Figure 3: Input Page For Query ‘q’



The Total No of Those Databases Which Contains The Query 'q' Only: 8
The Name of Those Databases containing The Query 'q' Only: DB1, DB2, DB3, DB4, DB5, DB6, DB7, DB8

Total No of Files In Each Database, Which Contains The Query 'q'

Name_of_Database	Total_Files
DB1	4
DB2	3
DB3	3
DB4	2
DB5	2
DB6	3

Total No of Files Are: 23

Names of All The Files Which Contain the Query 'q'

Database_Name	Data_Files	No_of_Occurance_of_Query
DB1	d1.txt	106
DB1	d2.txt	14
DB1	d3.txt	106
DB1	d4.txt	14
DB2	d6.txt	106
DB2	d7.txt	99
DB2	d9.txt	14
DB3	d10.txt	106

Names of All The Files According to User's Choice in Decreasing Order of Similarity With Query 'q'

Database_Name	Data_Files	No_of_Occurance_of_Querys
DB1	d1.txt	106
DB1	d3.txt	106
DB2	d6.txt	106
DB3	d10.txt	106
DB4	d13.txt	106

Figure 4: Result

5. CONCLUSION

With the increase of the number of search engines on the World Wide Web, providing easy, efficient and effective access to text information from multiple sources has increasingly become necessary. In this paper, we proposed two new methods for estimating the number of potentially useful databases and documents in selected databases. Our estimation methods are based upon established statistical theory and general database representation framework. Our experimental results indicate that these methods can yield substantial improvements over existing techniques. Our contributions consist of:

a. An algorithm DBSEL for selecting those databases from no. of databases which contain given query 'q'.
b. An algorithm HighRelDoc to return the top 'n' most relevant documents with respect to a given query from a collection of selected databases for any positive integer 'n'.

6. REFERENCES

[1] L. Gravano and H. Garcia-Molina, "Generalizing GlOSS to Vector-Space databases and Broker Hierarchies," Int'l Conf. Very Large Data Bases, p. 78-89, Sep. 1995.

[2] B. Jansen, A. Spink, J. Bateman, and T. Saracevic, "Real Life Information Retrieval: A Study of User Queries on the Web," Proc. ACM Special Interest Group on Information Retrieval Forum, vol. 32, no. 1, 1998.

[3] B. Yuwono and D. Lee, "Server Ranking for Distributed Text Resource Systems on the Internet," Proc. Fifth Int'l Conf. Database Systems for Advanced Applications, pp. 391-400, Apr. 1997.

[4] 4. J. Callan, Z. Lu, and W. Bruce Croft, "Searching Distributed Collections with Inference Networks," Proc. ACM Special Interest Group on Information Retrieval Conf. pp. 21-28, July 1995.

[5] Patricia Correia Saraiva, Edleno Silva deMoura, Nivio Ziviani,WagnerMeira, Rodrigo Fonseca, and Berthier Ribeiro-Neto. Rank–Preserving Two–Level Caching for Scalable Search Engines. In ACM, editor, Proceedings of the SIGIR2001 conference, New Orleans, LA, September 2001. SIGIR.

[6] C. Badue, R. Baeza-Yates, B. Ribeiro-Neto, and N. Ziviani. Distributed query processing using partitioned inverted files. In Proc. of the 9th String Processing and Information Retrieval Symposium (SPIRE), September 2002.

[7] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Trovatore: Towards a Highly Scalable Distributed Web Crawler. InWWWPosters 2001, 2001.

[8] N. Craswell, P. Bailey, and D. Hawking. Server Selection on theWorldWideWeb. In Proceedings of the Fifth ACM Conference on Digital Libraries, pages 37–46, 2000.

[9] B. Yuwono and D. Lee, "Server Ranking for Distributed Text Resource Systems on the Internet," Proc. Fifth Int'l Conf. Database Systems for Advanced Applications, pp. 391-400, Apr. 1997.

[10] Charu C. Aggarwal, Fatima Al-Garawi, and Philip S. Yu. Intelligent Crawling on the World Wide Web with Arbitrary Predicates. In Proceedings of the World Wide Web 2001 (WWW10), pages 96–105, 2001.

[11] S. Mukherjea. WTMS: A System for Collecting and Analyzing Topic-SpecicWeb Information. Computer Networks, 33(1):457–471, 2000.

[12] Boris Chidlovskii, Claudia Roncancio, and Marie-Luise Schneider. Semantic Cache Mechanism for Heterogeneous Web Querying. In Proceedings of the WWW8 Conference / Searching and Querying, 1999.

[13] J. Cho and H. Garcia-Molina. Estimating Frequency of Change. Technical report, Stanford University, 2000.

[14] Junghoo Cho and Hector Garcia-Molina. Synchronizing a Database to Improve Freshness. pages 117–128, 2000.

AUTHOR BIOGRAPHIES

Mr. Anil Agrawal received his Masters degree in Computer Science from Allahabad Agricultural Institute- Deemed University, Allahabad in 2007. Currently he is working as Assistant Professor at Ambalika Institute of Management and Technology, Lucknow, India. His research interest includes Data Mining.

Prof (Dr.) Mohd. Husain is working as Director at AZAD Institute of Engineering and Technology, Lucknow, India. He got his Masters Degree from UP Technical University & Ph.D Degree from Integral University in 2007. He has more than 12 years teaching experience and 10 years research experience in the field of Data mining. He has published more than 100 International and National publications

Mr. Raj Gaurang Tiwari is pursuing Ph. D. in Computer Science from Dravidian University. He received his Masters degree in Computer Applications from Dr. B. R. Ambedkar University, Agra in 2002 and Masters degree in Computer Sc. and Engg. From Gautam Buddh Technical University, Lucknow in 2010. Currently he is working as Assistant Professor at AZAD Institute of Engineering and Technology, Lucknow, India. His research interests are Knowledge-Based Engineering and Web Engineering. He authored more than 35 International and national journal and conference papers.

Mr. Subodh Kumar is working as Senior Lecturer at Ambalika Institute of Management and Technology, Lucknow, India. His research interest includes Data Mining.