computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

A METHODOLOGY FOR DATABASE AND DOCUMENT SELECTION

Raj Gaurang Tiwari [1)], Mohd. Husain [1)], Anil Agrawal [2)]

[1)] Azad Institute of Engineering and Technology Lucknow (UP), India
rajgaurang@gmail.com, mohd.husain90@gmail.com
[2)] Ambalika Institute of Management and Technology, Lucknow (UP), India
anil19974@gmail.com

Abstract: *As web users are facing the problems of information overload and drowning due to the significant and rapid growth in the amount of information and the number of users so there is need to provide Web users the more exactly needed information which is becoming a critical issue in web-based information retrieval and Web applications. In this work, we aspire to improve the performance of Web information retrieval and Web presentation through developing and employing Web data mining paradigms.*

Every search engine has a corresponding database that defines the set of documents that can be searched by the search engine. Generally, 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 several databases naturally. Proficient retrieval of preferred data can be attained if we can exactly predict 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 new approaches for database selection and documents selection. We also implement these algorithms using .net framework. Our experimental results indicate that these methods can yield substantial improvements over existing techniques.

Keywords: *Metasearch Engine, Distributed query processing, Document selection.*

1. INTRODUCTION

Internet has grown as a vast information source in recent years. To help ordinary users in finding desired data in the Internet, several search engines have been created. Every 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 engeine. For each term which represents a content word or a combination of several 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, HotBot, Lycos as well as Alta Vista are some 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. 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. 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 2.

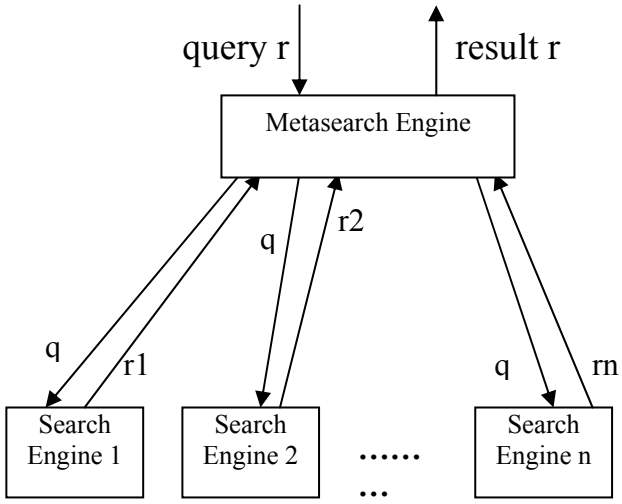

Fig. 1 – Two-Level Search Engine Organization

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.

2. RELATED WORK

Learning-based retrival approaches determine the number of documents to retrieve from a local database based on past retrieval experiences with the database. Several learning-based algorithms in [12, 13] are based on the use of training queries.

The guaranteed retrieval approach aims at guaranteeing such a property. The algorithm in [11] while guaranteeing that all potentially useful documents are retrieved may unnecessarily retrieve many non-similar documents. The approach in [14] is also a guaranteed retrieval approach but has a second goal of minimizing the retrieval of non similar documents. The document retrieval algorithm we propose in this thesis has the property that, when it is used together with any of our database selection methods, all the n most relevant documents for any query will be retrieved. Two solutions were proposed by W. Meng [14] for document selection. The first solution is to transform the threshold T0 for the global database (i.e., the global threshold) to a tight local threshold T_i for each local database D_i so that all documents in D_i having global similarities $\geq T_0$ are contained in the set of documents in D_i having local similarities $\geq$ Ti. This ensures that the former set of documents is retrieved. The second solution is that the metasearch engine modifies the user query before submitting it

to a local search engine such that the local similarity of a document in that local database with the modified query is the same as the global similarity of that document with the original user query.

3. DISTRIBUTED INFORMATION SYSTEM

Information is the critical ingredient for the operation and management of any organization. Information system (IS) is a coordinated collection of information subsystems that are rationally integrated to collect, store, process, retrieve, disseminate, and communicate information for the support of operations, management, and decision-making functions in business and other organizations. The objective of IS is to enhance productivity by improving the efficiency and effectiveness of business processes.

The field of information systems is unique in that it blends organizational and managerial concerns with the study of information technologies. The IS program is designed to provide students with (1) the technical background required to be able to function credibly in business and industry, and (2) organizational and managerial skills necessary to plan for and manage organizational information systems and to advance into leadership positions, particularly within the IS functional area of the firm.

Information system may be of distributed type. A distributed information system has been defined as "a combination of information processing facilities, data communication facilities, and endpoint facilities. Together, these support the movement and processing of files, programs, data, messages, and transactions".

Due to the advances in computer network technology and the steadily decreasing cost of hardware, distributed information systems have become an attractive alternative to centralized information systems. While many organizations still prefer the services of centralized systems, we are witnessing an increasing number of systems in which information processing and storage functions are distributed among several computers.

A distributed system is a collection of autonomous computers which cooperate in order to achieve a common goal. They do so without sharing memory or clock, and communicate by passing messages over a communication network. Ideally, the person using such a system is not aware of the different computers, their location, storage replication, load balancing, reliability or functionality. Instead the system should appear as though it runs on a single computer.

Documents may be full-text, bibliographic, sound, image, video or mixed-media records. A document server is set up by some individual or organisation wishing to publish a set of electronic documents. The publisher is referred to loosely as a document source. A person views such documents using a document client, for example a simple Web browser. To view a document the client sends a request containing a document identifier, such as an Internet URL, and the document server returns the document in question if available. This document-pull process is familiar to any person who has used the Web (Fig 2). In this figure the client request contains a document identifier and the server's response contains the full text of the document in question, if available. In this case the Netscape Navigator client is requesting the document with http://pastime.anu.edu.au/nick/work.shtml as its Uniform Resource Locator (URL). The client sends an HTTP request to http://pastime.anu.edu.au for the document identified (within the server). The server's response contains the full text of the document in question (which is in the HyperText Markup Language (HTML).

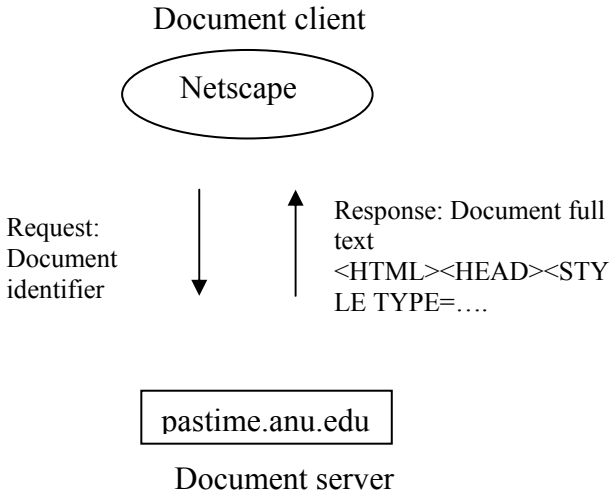

Fig. 2 – Document Request

An information retrieval problem arises when a person has access to many documents and requires some systematic organisation or search facility to find relevant information. A common form of information retrieval system is one which takes a query from the person who wishes to find information, and returns a list of documents which are estimated likely to be relevant. Retrieval of relevant information may also be aided by browsing amongst document hyper-links or some category/directory hierarchy.

A distributed information retrieval problem arises when the documents are spread across many document servers. In such a situation it may be possible for a single information retrieval system to request every document from every document server, and perform its search task over the combined document set. Alternatively, various search servers

may be set up on the network each covering documents from one or more document servers. In any case such networked information retrieval systems usually provide their search service to clients across the network (as opposed to restricting their service to a single machine). An information retrieval system available across the network is called a search server (Figure 3), and it is accessed sing a search client.

Systems which return search results, such as search servers and other information retrieval systems, usually return to the user a ranked results list R. The minimal content of R = <D, O> is a set of document identifiers D and some ordering O over D. A system is more effective if its results document set D contains more relevant documents, or the same number of relevant documents ranked more highly (O). A system is more efficient if it has reduced the costs involved in finding R. The cost of search includes several factors. Computation or storage resources may be expended at client or server. Network resources such as bandwidth may be expended in their communication. Monetary network usage or per-search charges might also apply. Users want a system which is both effective and efficient, in the latter case particularly minimizing the costs which apply to the user.

If the system is a search server, its effectiveness depends on the documents it indexes and its retrieval system. A retrieval system implements several retrieval algorithms, for ranking, stemming, case folding, relevance feedback and other functions. One type of search client is a simple client, such as Netscape Navigator in Figure 3. Users of a simple client face a number of problems. First, they may be have difficulty finding new servers and selecting which to search, particularly in an environment such as theWeb where there is no exhaustive list of servers and servers do not export descriptions of their documents. Further, if useful results are spread across multiple search servers, the user must query each in turn after learning the query language and interface conventions of each. This process of learning and querying sequentially is time consuming. The simple client also fails in terms of transparency, because the user is aware of search server heterogeneity, delays and down time. Finally, a simple client does not provide a unified view of results from different servers. The user has no indication of how results from one list compare to those of another, or even how each document matches their query. For example, one server given the query "david hawking" might return only documents containing the phrase, while others might return documents containing one word or the other, or even documents containing words with the same stem such as "hawk" and "hawker".

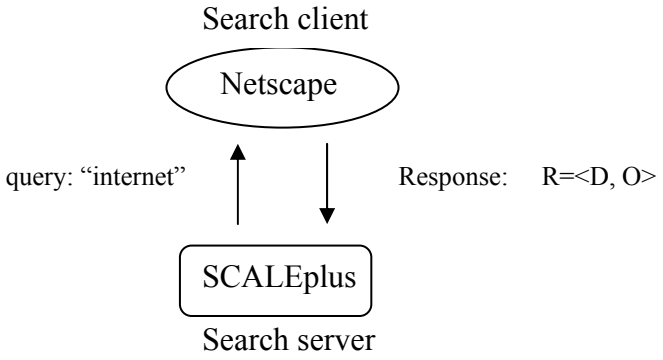

Fig. 3 – Simple Search

A search broker is a more sophisticated search client. Given a query and a set of search servers, it selects a set of servers likely to return relevant documents, queries them concurrently and produces a single ranked results list (Figures 4 and 5)

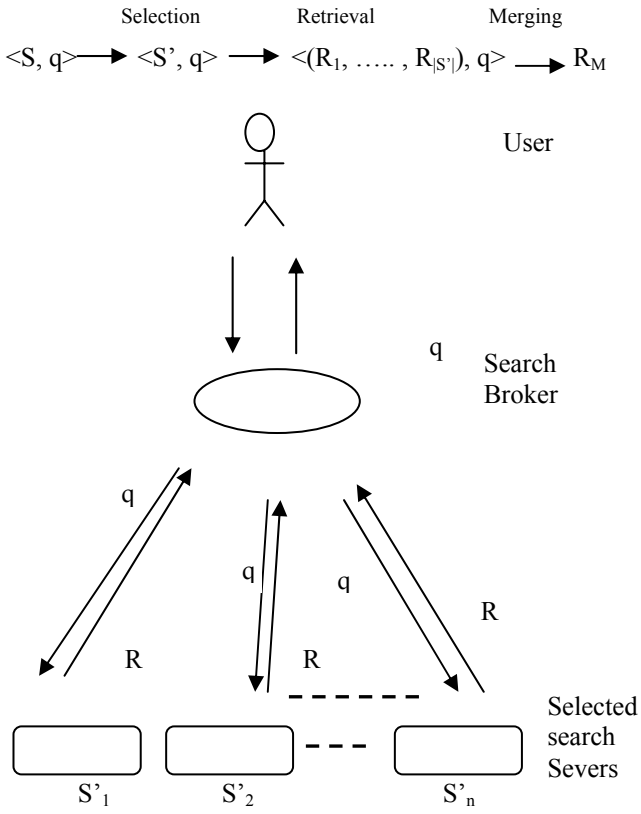

Fig. 4 – Search Broker Network Communication

The broker's task begins with a set of search servers S and a query q. A broker is set up to address servers S, analogously to a search server set up to search some document set. Identification of servers S is usually performed manually, as noted by Hawking and Thistlewaite [15] who calls it the problem of database detection.

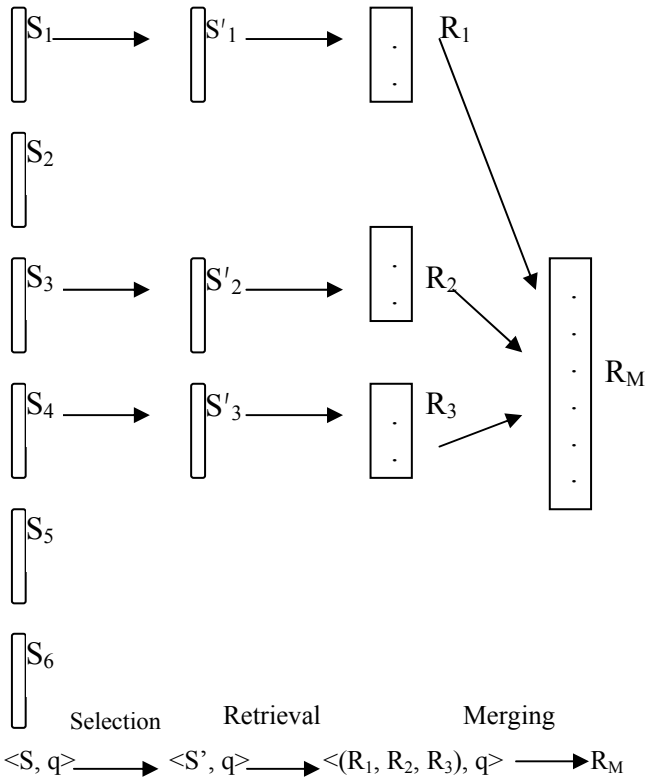

Fig. 5 – Search broker information flow

During server selection the broker selects a subset S'of servers S which are best for answering the user's query q. Choice of best servers might depend on both effectiveness and efficiency considerations.

During retrieval the broker applies the query q at servers S' to obtain results lists R_1, ... , $R_{|S'|}$. As described previously, each results list $R_i = <D_i, O_i>$ consists of a document set D_i and an ordering O_i. The broker must employ the appropriate retrieval methods – communication protocol, query language and results parser – to retrieve each list R_i. However, for a given set of servers S`, these methods have little influence over final broker effectiveness. Rather, the retrieval system and document set at server s'_i determines the quality of R_i. In an environment such as theWeb the broker designer usually has no control over server effectiveness. Instead the broker's retrieval methods either succeed or fail in retrieving R_i.

During results merging the broker combines results R_1, ... , $R_{|S'|}$ into a merged results list $R_M = <D_M, O_M>$, such that $D_M = D_1 \cup \cdots \cup D_{|S'|}$ and O_M is an effective ranking. Merging may be based on properties of R_1, ... ,$R_{|S'|}$, downloaded documents D_M or information provided by cooperating servers.

A broker may apply very simple methods for selection and merging. For example, it may select S' = S for every query as does MetaCrawler [9]. It may also merge results lists by simply concatenating the incoming lists. Such selection and merging is likely to be ineffective in an environment of many search servers, some of which return no relevant documents. Selecting all servers is also inefficient, again because it may lead to querying servers which contribute no useful information.

4. 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. 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 6.

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.

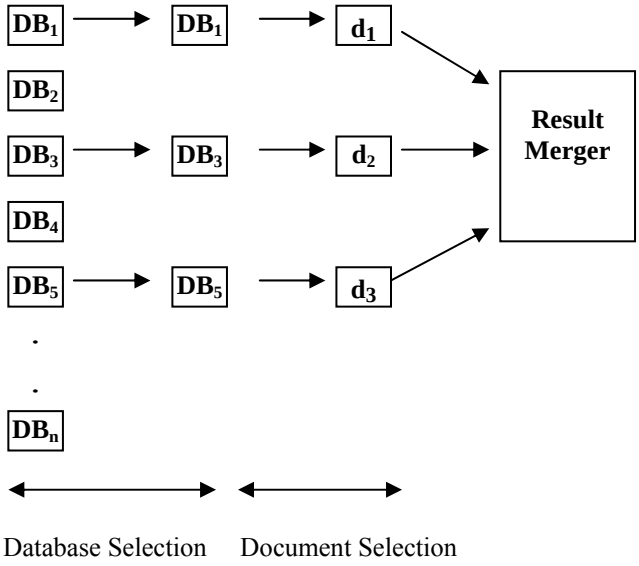

Fig. 6 – Database and document selection

4.1. 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, DB_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;
   }

4.2. 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'.

5. 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 [10] 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 [10] 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.

According to experimental results when the query word "INTEL" is searched in 9 databases containing many files as shown in fig 7, the five files having highest similarity with the query are selected from the databases. (Shown in figure 8)

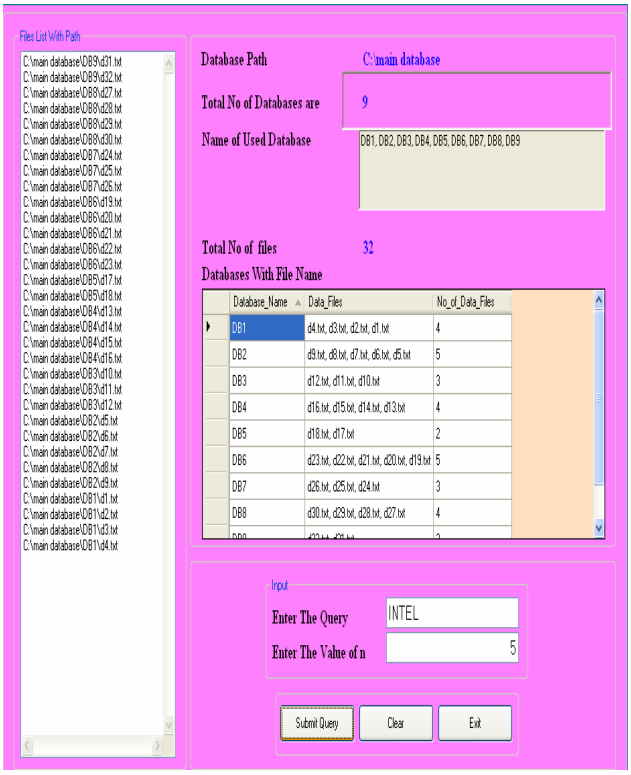

Fig. 7 – Input Page For Query 'q'

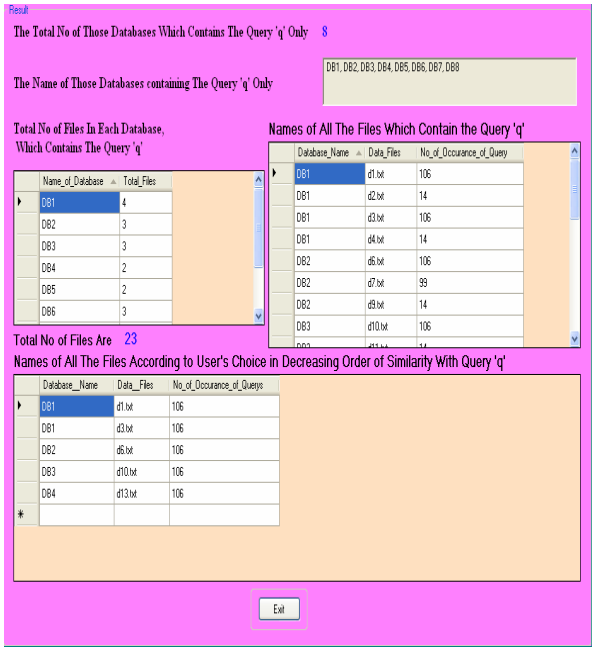

Fig. 8 – Result

6. 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’.

7. REFERENCES

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

[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*, (32) 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*, Apr. 1997, pp. 391-400.

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

[5] Patricia Correia Saraiva, Edleno Silva deMoura, Nivio Ziviani,Wagner Meira, 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. *In WWW Posters 2001*, 2001.

[8] N. Craswell, P. Bailey, and D. Hawking. Server Selection on the World Wide Web. *In Proceedings of the Fifth ACM Conference on Digital Libraries*, 2000, pp. 37-46.

[9] E. Selberg, and O. Etzioni. The MetaCrawler Architecture for Resource Aggregation on the Web. *IEEE Expert*, 1997.

[10] Wensheng Wu, Clement Yu, Weiyi Meng. Database Selection for Longer Queries, 2003.

[11] L. Gravano, and H. Garcia-Molina. Merging Ranks from Heterogeneous Internet sources. *International Conferences on Very Large Data Bases*, 1997.

[12] G. Towell, E. Voorhees, N. Gupta, and B. Johnson-Laird. Learning Collection Fusion Strategies for Information Retrieval. *12th Int'l Conf. on Machine Learning*, 1995.

[13] E. Voorhees, N. Gupta, and B. Johnson-Laird. Learning Collection Fusion Strategies. *ACM SI-GIR Conference*, Seattle, 1995.

[14] W. Meng, K.-L. Liu, C. Yu, X. Wang, Y. Chang, and N. Rishe. Determining Text Databases to Search in the Internet. *Proc. Int’l Conf. Very Large Data Bases*, Aug. 1998. pp. 14-25.

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.

***Prof.(Dr.) Mohd. Husain**: Presently working as Director, AZAD Institute of Engineering and Technology, Lucknow, India. He Received Ph.D. Degree from Integral University, Lucknow in 2008 and Master Degree (M.Tech.) from UP Technical University, Lucknow.*

He has about 21 years of experience in IT & Academics and 07 years research experience in the field of Data mining. He has published more than 110 International and National publications.

***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.*