A Novel Approach to Automatically Combine Search and Ranking Results

Mohd. Husain
Department of Computer Sc. & Engineering.
MGIMT, Lucknow, INDIA

Ayushi Prakash
Department of I.T.
Azad IET, Lucknow
INDIA

Afsaruddin Khan
Department of I.T.
Azad IET, Lucknow
INDIA

ABSTRACT
In the World Wide Web there are innumerable information sources containing very useful information that cannot be indexed by general-purpose search engines and hence cannot be visited by most common users. Of course, users can search a source through its query interface if they know where the source can be found. The idea of querying and collating results from multiple databases is not new. Internet meta-search engines, online catalogues, multi-databases and other kinds of information integration systems have attracted a lot of attention since the advent of the network.
In the web's early days, it used to be that a search engine either presented crawler-based results or human-powered listings. Today, it is extremely common for both types of results to be presented. Usually, a hybrid search engine will favor one type of listings over another. Often the user is interested in items that are both visually and semantically similar. With a view to supporting such functionality, the hybrid search engine provides a novel retrieval method, in which both visual and ontology search is employed for the same query. This novel method automatically combines different types of search results, and complements content-based search with ontology-based search and vice versa. In this paper, we study the rank aggregation problem in the context of the web, i.e. the problem of ranking result from various sources. There are various ranking aggregation methods available. We design an algorithm, based on which we propose a new rank aggregation method. It is observed that our proposed method is more effective and efficient than other well-known methods.

Keywords
Crawling, Multi-criteria selection, Meta Search Engines, Rank Aggregation, and Word Association.

1. INTRODUCTION
Document Retrieval is the computerized process of producing a list of documents that are relevant to an inquirer’s request by comparing the user’s request to an automatically produced index of the textual content of documents in the system. These documents can then be accessed for use within the same system. Nearly everyone today uses Document Retrieval systems, although they may not refer to them as such, but rather as Web-based search engines.
Searching on Internet has never been an easy task, even though it is one of the most common tasks performed on the Web. This task is becoming even more difficult with the continued growth of the amount of information posted on the World Wide Web. Web is quickly gaining grounds as a viable source of information over the past year or so with the appearance of many traditional and novel information services on the Web. Due to the sheer size of the Web and the rapidity with which new information gets added and existing information changes, finding relevant documents could sometimes be worse than searching for a needle in a haystack [1].
The increasing availability of machine readable texts led to rapid, widespread growth in the usage of Information Retrieval systems as the collection against which users could search increased dramatically. Not surprisingly, this also led to the flourishing of the early commercial Information Retrieval systems. These systems were sufficiently complex and non-intuitive that end users needed to have trained search intermediaries do their searches, because these intermediaries could understand the intricacies of the data records in the Information Retrieval System and were well trained in constructing queries in Boolean logic. These early systems were not based on free text searching, but rather, required the intermediary to know the exact wording and syntax to use in searching, both in proper names and subject based descriptions – referred to respectively as authority files and controlled vocabulary.
Measuring relative performance of information retrieval (IR) systems such as Web search engines is essential for research and development and for monitoring search quality in dynamic environments. However, due to the size and dynamic nature of document collections and users, evaluating or comparing the retrieval performance of search engines in regular intervals is difficult. Automatic evaluation of retrieval systems is the ultimate solution to this problem. Assessing IR effectiveness normally requires a test collection, a set of queries, and relevance information about each document with respect to each query. However, for very large databases creating relevance judgment is a difficult and extremely time-consuming task, since all documents need to be judged for relevance to each query [2].
Searching for relevant information is a difficult and sometimes very time-consuming procedure because of an enormous amount of information and the lack of structure. Traditional Web search engines return the user query results by displaying a long list of documents without any process of data classification or clustering. Considering the variations among search engines, the efficient integration of the results from several different search engines for the same query, is an important but difficult technique that can dramatically improve Web search technology. To exhaust a Web search, one often has to use several search tools and has to be familiar with the different interfaces and searching rules. It would be desirable to have a central place with a uniform interface, where a query can be entered and the search conducted simultaneously in as many search tools and directories as necessary. The search results can be brought back and displayed in a consistent format [5].
Once a good ranking function has been engineered, query throughput often becomes a critical issue. Large search engines need to answer thousands of queries per second on collections of several billion pages. Even with the construction of optimized index structures, each user query

requires a significant amount of data processing on average. To deal with this workload, search engines are typically implemented on large clusters of hundreds or thousands of servers, and techniques such as index compression, caching, and result presorting and query pruning are used to increase throughput and decrease overall cost.

There is a growing need for formal methods that guarantee the reliability, correctness, and efficiency of computerized systems. Document Retrieval is more commonly referred to as Information Retrieval. It is the computerized process of producing a list of documents that are relevant to an inquirer's request by comparing the user's request to an automatically produced index of the textual content of documents in the system. These documents can then be accessed for use within the same system. Nearly everyone today uses Document Retrieval systems, although they may not refer to them as such, but rather as Web-based search engines. In Document Retrieval, some processes take place dynamically when the user inputs their query, while other processes take place off-line in advance and in batch mode and do not involve individual users. These static processes are run on the documents that will be made available in the retrieval system [3].

To provide users a certain degree of robustness of search in the face of various shortcoming and bases of individual search engines, we can rank the database with respect to several small subsets of the queries, and aggregate these rankings. This is commonly known as rank aggregation. Rank aggregation can be used in situations where the user preference includes a variety of criteria, and the logic of classifying a document as acceptable or not is too complex such as multi-criteria selection or word association queries. Multi-criteria selection can be when a user tries to choose a product from its database and word association queries can be when the user tries to search for a good document on a topic, knowing a list of keywords that collectively describe the topic, but not sure that the best document on the topic necessarily contains all of them. Ranking a list of several alternatives based on one or more criteria is encountered in many situations like in identifying the best alternatives [1]. In case of single criteria for ranking, the task is easy and is simply a reflection of the judges (search engine in the case of meta-search, individual criterion for multi-criteria selection, and subsets of queries in the case of word association queries) opinions. In contrast, there can be another case when individual ranking preferences of several judges is given.

2. META SEARCH ENGINES

Many search engines not only log query submissions but also record details each time a user clicks on a search result. In order to rank the results obtained, we have made use of rank aggregation strategies. A meta search engine can be use to transmit user's search simultaneously to several individual search engines and their database of web pages and get results from all the search engines queried [2]. A lot of time can be saved if the search is initiated at a single point sparing the need to learn and use several separate search engines.

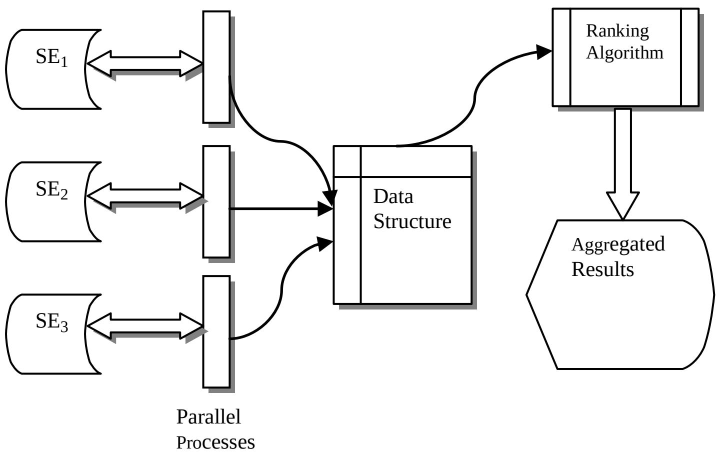

Figure 1. Architecture of a meta-search engine

The idea of querying and collating results from multiple databases is not new. Internet meta-search engines, online catalogues, multi-databases and other kinds of information integration systems have attracted a lot of attention since the advent of the network. There are many meta-search and information integration systems are available.

Most current meta-search engines only use a simplest user interface. Some systems only list all user interfaces of different sources separately on a page or several hierarchically organized pages. In order to avoid losing important functions of search engines, both their generality and particularity should be considered when constructing the user interface of a meta-search engine.

Meta search engines help us in achieving the following objectives- as the World Wide Web is a huge unstructured corpus of information, various search engines crawl the WWW from time to time and index the web pages [9]. However, it is virtually impossible for any search engine to have the entire web indexed. Most of the time a search engine can index only a small portion of the vast set of web pages existing on the Internet. Each search engine crawls the web separately and creates its own database of the content. Therefore, searching more than one search engine at a time enables us to cover a larger portion of the World Wide Web. Secondly, crawling the web is a long process, which can take more than a month whereas the content of many web pages keep changing more frequently and therefore, it is important to have the latest updated information, which could be present in any of the search engines. However, good ranking strategies are needed in order to aggregate the results obtained from the various search engines. Quite often, many web sites successfully spam some of the search engines and obtain an unfair rank. By using appropriate rank aggregation strategies, we can prevent such results from appearing in the top results of a meta-search.

Meta search engines can be categorized as-

- Meta search engines for serious deep digging.
- Meta Search engines which aggregate the results obtained from various search engines.
- Meta Search engines which present results without aggregating them.

Meta search engine of the second type i.e. which aggregate the results obtained is more useful. We have proposed an aggregation method for such an aggregation. Any method for rank aggregation [9] for Web applications must be capable of dealing with the fact that only the top few hundred entries of each ranking are available. Of course, if there is absolutely no overlap among these entries, there isn't much any algorithm can do; the challenge is to design rank aggregation algorithms that work when there is limited but non-trivial overlap among the top few hundreds or thousands of entries in each ranking. Finally, in light of the amount of data, it is implicit that any rank aggregation method has to be computationally efficient.

There are several applications of rank aggregation methods in the context of searching and retrieval [5] such as – Meta-Search, Aggregating Ranking Functions, Spam Reduction, Word Association Techniques and Search Engine Comparison.

3. RANKING

In every query formulation technique there is a human in the loop. From very simple queries to extremely complex queries and there must be a person to define the information need in the form of a query. One of the system performance measures that are often ignored is the level of effort required for query construction. In many cases of the information need, the required query is quite simple. Specifically, simple queries perform well in the case where the information density is high. For example, if the analyst wants to know the score of the Lakers game last night, there are many sources that can provide that information and a simple query will suffice. In other cases, particularly where the information density low, the query must be complex and broad so that relevant data is not missed.

An online information seeker often fails to find what is wanted because the words used in the request are different from the words used in the relevant material. Moreover, the searcher usually spends a significant amount of time reading retrieved material in order to determine whether it contains the information sought.

The conceptual indexing and retrieval system used for these experiments automatically extracts words and phrases from unrestricted text and organizes them into a semantic network that integrates syntactic, semantic, and morphological relationships. The resulting conceptual taxonomy is used by a specific passage-retrieval algorithm to deal with many paraphrase relationships and to find specific passages of text where the information sought is likely to occur. The database systems support a simple Boolean query retrieval model, where a selection query on a SQL database returns all tuples that satisfy the conditions in the query. This often leads to the Many-Answers Problem: when the query is not very selective, too many tuples may be in the answer [5].

Document surrogates containing both anchor text and query associations have been found to improve retrieval effectiveness. Indeed, Web search engines have long made use of anchor text to improve result quality. For retrieval purposes, a text document may be supplemented with additional terms derived from external sources such as metadata, anchor text and so on. In the case of document surrogates, the additional terms form their own document which is used instead of the original. Retrieval may be based on scoring the surrogate collection or those scores may be combined with scores from the original collection. The following are examples of the use of surrogate or supplemented documents [5].

Given a universe U, an ordered list (or simply, a list) L with respect to U is an ordering of a subset S of U, i.e. -

$L = [x_1 > x_2 > ... > x_d]$, with each x_i in S, and > is some ordering relation on S. Also, if i in U is present in L, let L(i) denote the position or rank of i (a highly ranked or preferred element has a low-numbered position in the list). For a list L, let |L| denote the number of elements. By assigning a unique identifier to each element in U, we may assume without loss of generality that $U = \{1, 2, ..., |U|\}$.

Depending on the kind of information present in L, three situations arise -

1. If L contains all the elements in U, then it is said to be a full list. Full lists are, in fact, total orderings of U. For instance, if U is the set of all pages indexed by a search engine, it is easy to see that a full list emerges when we rank pages with respect to a query according to a fixed algorithm [10].

2. There are situations where full lists are not convenient or even possible. For instance, let U denote the set of all Web pages in the world. Let L denote the results of a search engine in response to some fixed query. Even though the query might induce a total ordering of the pages indexed by the search engine, since the index set of the search engine is almost surely only a subset of U, we have a strict inequality $|L| < |U|$. In other words, there are pages in the world which are unranked by this search engine with respect to the query. Such lists that rank only some of the elements in U are called partial lists.
   A special case of partial list is as follows –
   If S is the set of all the pages indexed by a particular search engine and if L corresponds to the top 100 results of the search engine with respect to a query, clearly the pages that are not present in list L can be assumed to be ranked below 100 by the search engine. Such lists that rank only a subset of S and where it is implicit that each ranked element is above all unranked elements, are called top d lists, where d is the size of the list [12].

To measure the distance between two full lists with respect to a set S, distance measures are -

(1) The distance (D_1) is the sum, over all elements i in S, of the absolute difference between the rank of i according to the two lists. Formally, given two full lists L and M, their distance (D_1) is given by-

$$D_1 (L, M) = \sum_i |L(i) - M(i)| \qquad (1)$$

After dividing this number by the maximum value (1/2)|S|2, one can obtain a normalized value of the distance (D_1), which is always between 0 and 1. The distance (D_1) between two lists can be computed in linear time.

(2) The second distance (D_2) counts the number of pair wise disagreements between two lists; that is, the distance between two full lists L and M is -

$$D_2 (L, M) = |\{(i, j) : i < j, L(i) < L(j) \text{ but } M(i) > M(j)| \qquad (2)$$

Dividing this number by the maximum possible value (1/2) S (S - 1) we obtain a normalized version of the distance (D_2). The distance (D_2) for full lists is the "bubble sort" distance, i.e., the number of pair wise adjacent transpositions needed to transform from one list to the other. The distance (D_2) between two lists of length n can be computed in n log n time using simple data structures. The above measures are metrics and extend in a natural way to several lists. Given several full lists L, M_1, ..., M_k, for instance, the normalized distance (D_1) of L to M_1, ..., M_k is given by-

$$D_1 (L, M_1,....., M_k) = (1/k) \sum_i D_1(L, M_i) \qquad (3)$$

One can define generalizations of these distance measures to partial lists. If M_1, ..., M_k are partial lists, let U denote the union of elements in M_1, ..., M_k, and let L be a full list with respect to U.

(3) Given one full list and a partial list, the distance (D_1) weights contributions of elements based on the length of the lists they are present in. More formally, if L is a full list and M is a partial list, then -

$SD_1(L, M) = \sum_{i \text{ in } M} |(L(i)/|L|) - (M(i)/|M|)|$ (4)

We will normalize SD_1 by dividing by $|M|/2$.

4. OUR PROPOSED WORK

In our proposed algorithm, the distances are used to rank the various results. Let $P_1, P_2, \ldots, P_n$ be partial lists obtained from various search engines. Let their union be S. A weighted bipartite graph for distance (D_1) optimization (N, SP, D_1) is defined as-

N = set of nodes to be ranked
SP = set of positions available

$D_1(e,p)$ = is the distance (from the Pi's) of a ranking that places element 'e' at position 'p', given by-

$D_1(e,p) = \sum i = 1k \mid Pi(e)/|Pi| - p/n|$ (5)

where n = number of results to be ranked and |Pi| gives the cardinality of Pi.

Computation of aggregation for partial lists is NP-hard. Hence we have used distance measure (D_1). This problem can be converted to a minimum cost perfect matching in bipartite graphs. There are various algorithms for finding the minimum cost perfect matching in bipartite graphs.

Our proposed algorithm works as follows–

Step1: Calculate the reduced cost matrix from the given cost matrix by subtracting the minimum of each row and each column from all the other elements of it.

Step2: Cover all the zeroes with the minimum number of horizontal and vertical lines.

Step3: If the number of lines equals the size of the matrix, find the result.

Step4: If all of the zeroes are covered with fewer lines than the size of the matrix, find the minimum number that is uncovered.

Step5: Subtract it from all uncovered values and add it to any value(s) at the intersections of the lines.

Step6: Repeat until result is obtained.

In evaluating the performance of the ranking strategies for all the queries, we have chosen precision as a good measure of relative performance because all the ranking strategies work on the same set of results and try to get the most relevant ones to the top. Hence, a strategy that has a higher precision at the top can be rated better from the user's perspective. We have plotted the precision of the ranking strategies with respect to the recall. The recall is calculated as the number of relevant documents retrieved/total number of relevant results thus judged. It can be observed that on an average, our proposed ranking aggregation method gives better precision for the given set of results.

TABLE 1. Precision of several Rank Aggregation methods at a given Recall

Condorcet Method							
Precision	0.8	0.7	0.8	0.6	0.4	0.5	0.3
Recall	0.12	0.24	0.48	0.6	0.72	0.84	0.96
Borda's Method							
Precision	0.9	0.5	0.5	0.6	0.4	0.5	0.3
Recall	0.12	0.24	0.48	0.6	0.72	0.84	0.96
Our Method							
Precision	0.9	0.6	0.9	0.8	0.6	0.6	0.3
Recall	0.12	0.24	0.48	0.6	0.72	0.84	0.96

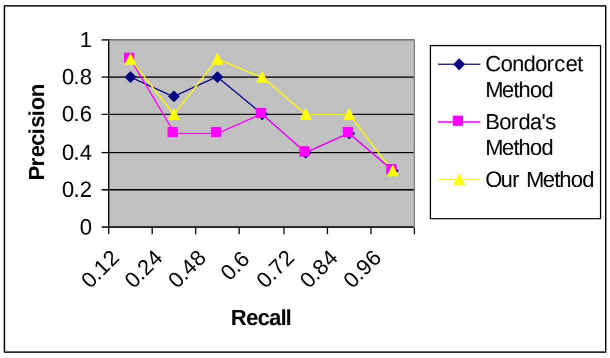

Figure 2. Graphical Representation of Precision and Recall

5. CONCLUSION

We have proposed a rank aggregation method which works on our designed algorithm. This method has the advantage of being applicable in a variety of contexts and tries to use as much information as available. Our method is simple for implementation and do not have any computational overhead as compared to other methods. It is efficient, effective and provides robustness of search in the context of web.

6. REFERENCES

[1]. J. I. Marden. Analyzing and Modeling Rank Data. Monographs on Statistics and Applied Probability, No 64, Chapman & Hall, 1995.

[2]. Meng, W., Yu, C., & Liu, K.-L., Building efficient and effective metasearch engines. ACMComputing Surveys, 2001, 34(1), 48–89.

[3]. Aslam, J. A., Montague, M., Models for metasearch. In: Proceedings of the 24th ACMSIGIR conference (pp. 276–284), 2001.

[4]. Cynthia Dwork, Ravi Kumar, Moni Naor, D Siva Kumar, Rank Aggregation Methods for the web. In proceedings of the Tenth World Wide Web Conference, 2001.

[5]. Baeza-Yates, R., & Ribeiro-Neto, B., Modern information retrieval. New York: ACM Press, 2010.

[6]. Amitay, E., Carmel, D., Lempel, R., & So.er, A., Scaling IR-system evaluation using term relevance sets. In Proceedings of the 27th ACMSIGIR conference, 2004, pp. 10–17.

[7]. Soboro., I., Nicholas, C., & Cahan, P. Ranking retrieval systems without relevance judgments. In Proceedings of the 24th ACM SIGIR conference, 2001, pp. 66–73.

[8]. Croft, W. B., Combining approaches to information retrieval. In W. B. Croft (Ed.), Advances in information retrieval: recent research from the center for intelligent information retrieval. Kluwer Academic Publishers, 2000.

[9]. Cynthia Dwork, Ravi Kumar, Moni Naor, D Siva Kumar, Rank Aggregation Methods for the web. In proceedings of the Tenth World Wide Web Conference, 2010.

[10]. Fan, W., Fox, E. A., Pathak, P., & Wu, H. The effects of fitness functions on generic programming-based ranking discovery for Web search. Journal of the American Society for Information Science and Technology, 55(7), 2004, 628–636.

[11]. Hawking, D., Craswel, N., Bailey, P., & Gri.ths, K., Measuring search engine quality. Information Retrieval, 4(1), 2001, 33–59.

[12]. Nuray, R., & Can, F., Automatic ranking of retrieval systems in imperfect environments. In Proceedings of the 26th ACM SIGIR conference 2009, pp. 379–380.