International Journal of Information Sciences and Application.
ISSN 0974-2255 Volume 3, Number 2 (2011), pp. 119-125

http://www.irphouse.com

Improved Search Quality Using Rank Aggregation

N. Sathish Kumar[1], G. Sampath Kumar[2], N. Srinivas Rao[3] and N. Satyanarayana[4]

[1]Associate Professor, Department of CSE, SVS Institute of Technology, Bheemaram, Hanamkonda, Warangal-50600l, A.P., India
E-mail: satish4info@gmail.com
[2]Assistant Professor, Department of CSE, Adam's Engineering College Seetarampatnam, Paloncha-507115, Khammam, Dt. A.P., India
E-mail: kasyapsam@hotmail.com
[3]Associate Professor, Department of CSE, Adam's Engg. College, Seetarampatnam, Paloncha-507115, Khammam Dt., A.P., India
E-mail: srinivas.nune@gmail.com
[4]Professor, Department of CSE, Nagole Institute of Technology & Sciences Hyderabad, A.P., India
E-mail: nsn1208@gmail.com

Abstract

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.

Introduction

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.

Meta Search Engines

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.

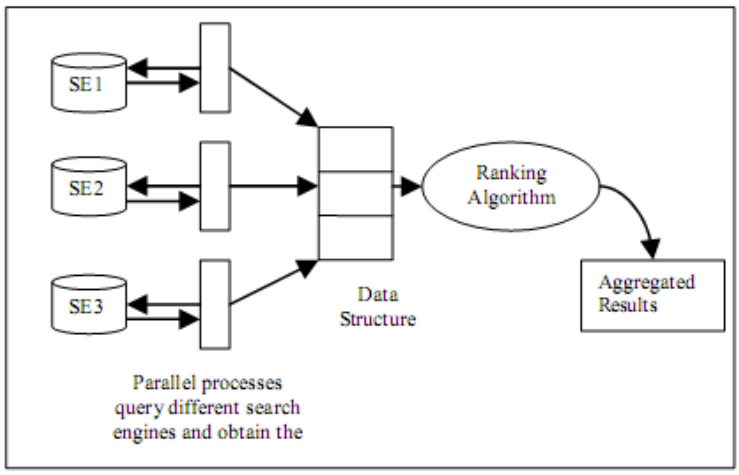

Fig 1: Architecture of a Meta Search Engine

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 [3]. 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

i. Meta search engines for serious deep digging.
ii. Meta Search engines which aggregate the results obtained from various search engines.
iii. 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 [4] 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, here 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.

Ranking

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 = [x1 > x2 > ... > xd], with each xi 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 [1].
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 lists is the following – 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. To measure the distance between two full lists with respect to a set S, distance measures are:

1. The distance (D1) 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 (D1) is given by-

$$D1\ (L, M) = \Sigma i\ |L(i) - M(i)| \quad (1)$$

After dividing this number by the maximum value (1/2)|S|2, one can obtain a normalized value of the distance (D1), which is always between 0 and 1. The distance (D1) between two lists can be computed in linear time.

2. The second distance (D2) counts the number of pair wise disagreements between two lists; that is, the distance between two full lists L and M is

$$D2\ (L, M) = |\{(i, j) : i < j, L(i) < L(j) \text{ but } M(i) > M(j)| \quad (2)$$

Dividing this number by the maximum possible value (1/2)S(S - 1) we obtain a normalized version of the distance (D2). The distance (D2) 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 (D2) 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, M1, ..., Mk, for instance, the normalized distance (D1) of L to M1, ..., Mk is given by-

$$D1\ (L, M1, \ldots, Mk) = (1/k)\ \Sigma i\ D1(L, Mi) \quad (3)$$

One can define generalizations of these distance measures to partial lists. If M1, ..., Mk are partial lists, let U denote the union of elements in M1, ..., Mk, and let L be a full list with respect to U. (3) Given one full list and a partial list, the distance (D1) 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:

$$SD1\ (L, M) = \Sigma i \text{ in } M\ |(L(i)/|L|) - (M(i)/|M|)| \quad (4)$$

We will normalize SD1 by dividing by |M|/2.

Our Proposed Work

In our proposed algorithm, the distances are used to rank the various results. Let P1, P2,……., Pn be partial lists obtained from various search engines. Let their union be S. A weighted bipartite graph for distance (D1) optimization (N, SP, D1) is defined as-

N = set of nodes to be ranked

SP = set of positions available

D1(e,p) = is the distance (from the Pi's) of a ranking that places element 'e' at position 'p', given by-

$$D1(e,p) = \Sigma I=1\ k \mid Pi(e)/|Pi| - p/n| \quad (5)$$

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

Computation of aggregation for partial lists is NPhard. Hence we have used distance measure (D1). 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 Rak Aggregation methods at a give Recall.

Naive's Method							
Precision	0.1	0.9	0.7	0.5	0.2	0.5	0.1
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

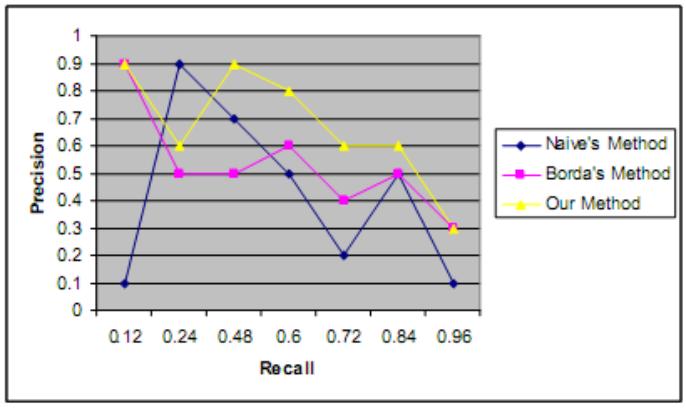

Fig 2: Graphical Representation of Precision and Recall

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 and effective and provides robustness of search in the context of web.

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, 2001.
[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, 2001.
[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 2003,pp.379–3