Tadpole: A Meta search engine
Evaluation of Meta Search ranking strategies

www.stanford.edu/~pavan/tadpole.html

Mahathi S Mahabhashyam
mmahathi@stanford.edu

Pavan Singitham
pavan@stanford.edu

Abstract

In this write up, we explain the design of Tadpole, a Meta search engine which obtains results from various search engines and aggregates them. We discuss three meta-search ranking strategies – two positional methods and a scaled foot rule optimization method and study the response-time/result quality trade-offs involved.

1.Introduction

A Meta search engine transmits user's search simultaneously to several individual search engines and their databases of web pages and gets results from all the search engines queried. We could thus save a lot of time by initiating the search at a single point and sparing the need to use and learn several separate search engines. This can be even more helpful, if we are looking for a broad range of results.

In our project, we have implemented a Meta search engine, which queries Google, Altavista and MSN databases. We have provided an interface for searching these search engines along with several advanced options for phrase search, conjunction, disjunction and negation of the key words. In order to rank the results obtained, we have made use of three rank aggregation strategies and evaluated the results obtained. Out of these, two are positional methods, which make use of the result's rank in each of the separate search engine to obtain a new rank by simple aggregation. The third one is a scaled foot rule optimization technique.

2.Motivation

There are primarily two motivating factors behind our developing a meta-search engine. Firstly, 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. 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.

Meta Search engines help us achieve the afore-mentioned objectives. However, we need good ranking strategies 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.

Our primary motivation was to develop a simple meta-search engine and study the response-time and performance trade-offs involved.

3.Previous Work

There are quite a few Meta search engines available on the Internet, which can be categorized as follows

1. Meta search engines for serious deep digging Ex: Surfwax, Copernic Basic

2. Meta Search engines which aggregate the results obtained from various search engines Ex: Vivisimo, Ixquick

3. Meta Search engines which present results without aggregating them Ex: Dogpile

Meta-search engines of the first kind are not available as free-software. So, their benefits are not reaped by most users. Some of the other issues involved and drawbacks of meta-search engines are provided in [3].

An aggregation of the results obtained would be more useful than just dumping the normal results. For such an aggregation, Ravi Kumar et al [1] have suggested several Rank aggregation methods for the web, broadly categorized as Borda's positional methods, Foot rule /Scaled Foot rule Optimization methods, Markov Chain methods for rank aggregation. They also suggest a local Kemenization technique, which brings the results that are ranked higher by the majority of the search engines to the top of the Meta search-ranking list. This is effective in avoiding spam.

4.Organization

The organization for the report is as follows:Section 5 discusses the architecture and design of Tadpole, the meta-search engine developed by us. Section 6 gives a study of the tradeoffs involved. In Section 7. we describe a few problems we encountered during the project. Section 8 gives the conclusion and future work.

5.Architecture of Tadpole

When a user issues a search request, multiple threads are created in order to fetch the results from various search engines. Each of these threads is given a time limit of 3 seconds to return the results, failing which a time out occurs and the thread is terminated.

Each process converts the given query to the format specific to the search engine it is

Error!.

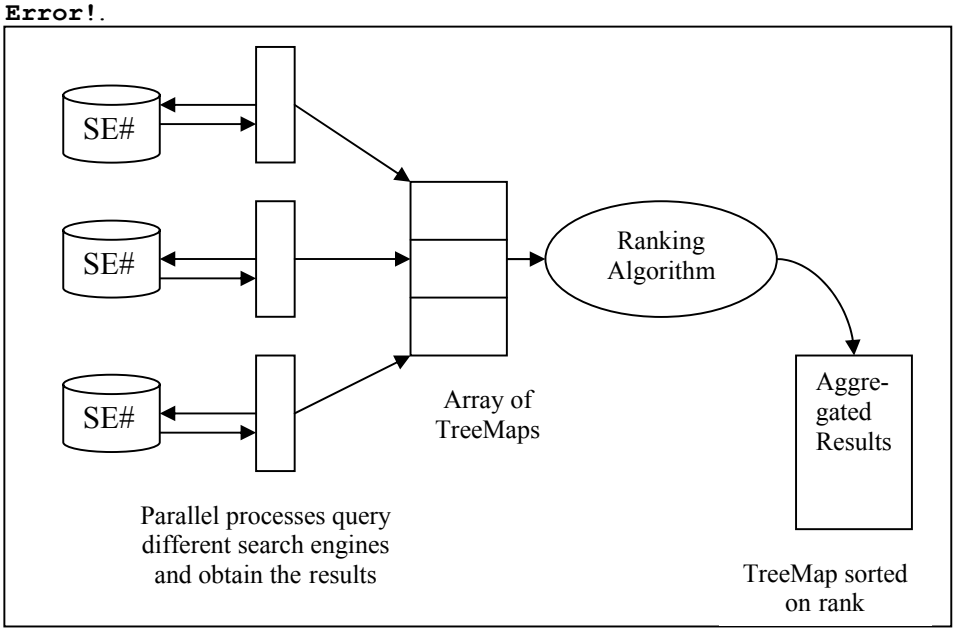


Figure 1

dealing with. This request is sent to the search engine via the java URL object and the results are obtained in the form of a HTML page. This HTML results page is parsed by the process and for each result, the URL, Title, Description, Rank and SearchSource are stored, creating a Result object. These results are entered into a **TreeMap** data structure with the key as the **url** and the item as the **Result** object.

The GUI also provides for advanced search options for entering Boolean queries, Phrase searches, selecting the number of results per search engine and the selection of search engines to be queried.

5.1 Design Decisions

During the design of Tadpole, we various design decisions were taken. Some of them are listed below:

Why TreeMap?

TreeMap data structure combines the nice features of a tree (low search and retrieval time) and Map (easy association) data structures. By storing the results with the URL as the key, we can retrieve a result in (log n) time while removing the duplicates and merging them in the ranking algorithm. This helps in a considerable speed up when we have hundreds of results from each search engine.

The TreeMaps thus obtained from each of the threads are then inserted in an array and passed on to the Ranking algorithm. The Ranking algorithm then returns a tree map sorted on rank.

Why these three ranking strategies?

The positional methods are computationally more efficient. They give a good precision when compared to just aggregation of results without using any ranking. The scaled-footrule method is computationally more complex, but is proven to have given much better performance. It is also useful in the reduction of spam to an extent. As the basic idea of this project was to study the trade-offs involved, we wanted to get a gradation in the level of computational complexity and performance and so we chose these three rank aggregation methods.

5.2 Ranking Aggregation Methods Implemented

Take the Best Rank

In this algorithm, we try to place a URL at the best rank it gets in any of the search engine rankings.

That is,

MetaRank (x) =

Min(Rank1(x),Rank2(x),…. , Rankn(x));

Clashes are avoided by an ordering of the search engines based on popularity. That means, if two results claim the same position in the meta-rank list, the result from a more popular search engine, (say Google) is preferred to the result from a less popular one.

Borda's Positional Method

In this algorithm, the MetaRank of a url is obtained by computing the Lp-Norm of the ranks in different search engines.

MetaRank(x)=

$[\Sigma(Rank1(x)^p, Rank2(x)^p, \ldots , Rankn(x)^p)]^{1/p}$

In our algorithm, we have considered the L1-Norm which is the sum of all the ranks in different search engine result lists. Clashes are again avoided by search engine popularity.

The search source for a URL, which is displayed in the meta search results, is set as the search engine in which the URL is ranked the best.

Scaled Footrule Optimization Method

In this algorithm, the scaled footrule distances are used to rank the various results. Let T1, T2 , … Tn be partial lists obtained from various search engines. Let their union be S. A weighted bipartite graph for scaled footrule optimization (C,P,W) is defined as

C = set of nodes to be ranked

P = set of positions available

W(c,p) = is the scaled- footrule distance (from the Ti's) of a ranking that places element 'c' at position 'p', given by

$W(c,p) = \sum_{I=1}^{k} | Ti(c)/|Ti| - p/n|$

Where n = number of results to be ranked and |Ti| gives the cardinality of Ti.
Computation of foot-rule aggregation for partial lists is NP-hard [1]. Hence the use of scaled foot-rule distance measure. This problem can be converted to a minimum cost perfect matching in bipartite graphs described above. There are various algorithms for finding the minimum cost perfect matching in bipartite graphs. We have used the Hungarian method for doing it.
The Hungarian method proceeds as follows:

- Obtain 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.
- Try to cover all the zeroes with the minimum number of horizontal and vertical lines.
- If the number of lines equals the size of the matrix, find the solution.
- If you have covered all of the zeroes with fewer lines than the size of the matrix, find the minimum number that is uncovered.
- Subtract it from all uncovered values and add it to any value(s) at the intersections of your lines.
- Repeat until a solution is obtained.

A detailed description of the algorithm is provided in [3]

6.Evaluation of Ranking Strategies

6.1 Algorithmic Complexity

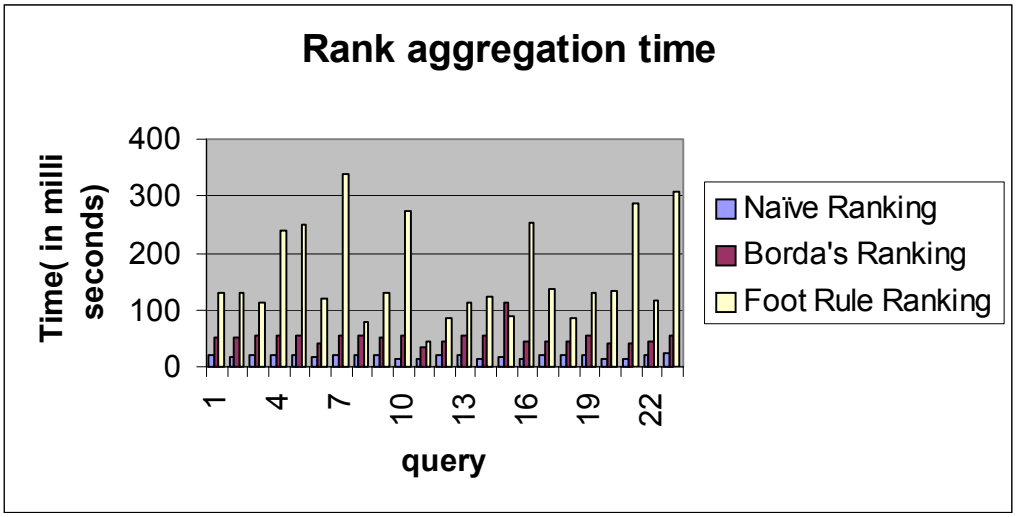
The first parameter for testing the three ranking strategies is the time complexity of the algorithms. The positional methods – MinRanker and Borda's positional method take linear time, that means they have a complexity of O(n).
Scaled Footrule optimization can be solved using the Hungarian algorithm for Bipartite-matching.

6.2 Rank Aggregation Time

The aggregation times of various ranking strategies were measured with respect to each other and with normal search engines. The evaluation was carried out with respect to the following set of 38 queries, which were previously used in other studies [1,4,5]

affirmative action, alcoholism, amusement parks, architecture, bicycling, blues, cheese, citrus groves, classical guitar, computer vision, cruises, Death Valley, field hockey, gardening, graphic design, Gulf war, HIV, java, Lipari, lyme disease, mutual funds, National parks, parallel architecture, Penelope Fitzgerald, recycling cans, rock climbing, San Francisco, Shakespeare, stamp collecting, sushi, table tennis, telecommuting, Thailand tourism, vintage cars, volcano, zen buddhism, and Zener.

The results are summarized below:



Average Rank Aggregation Times
Naïve Ranking - 18.6 msec
Borda's Ranking - 51.2 msec
FootRule Ranking - 161.5 msec

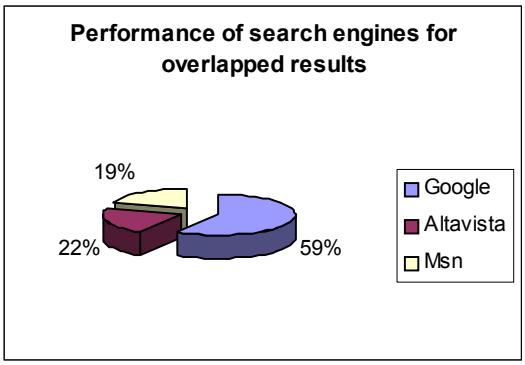
We observe that the rank aggregation times for the foot rule ranking are on an average thrice those for the Borda's positional ranking.

6.3 Overlap across search engines – Relative Search Engine Performance

Among the top 10 results obtained for each query , we found the results that overlap across multiple search engines. An interesting observation would be to find which search engines rank the overlapping results better. An intuition behind such a measure is that a search engine, which ranks the overlapping results, better can be regarded as a better search engine considering that the overlapping results are more relevant.

6.4 Performance of the various rank aggregation methods

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.

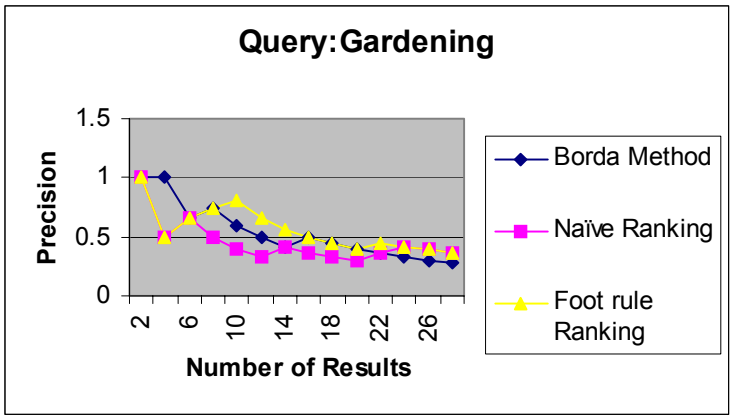


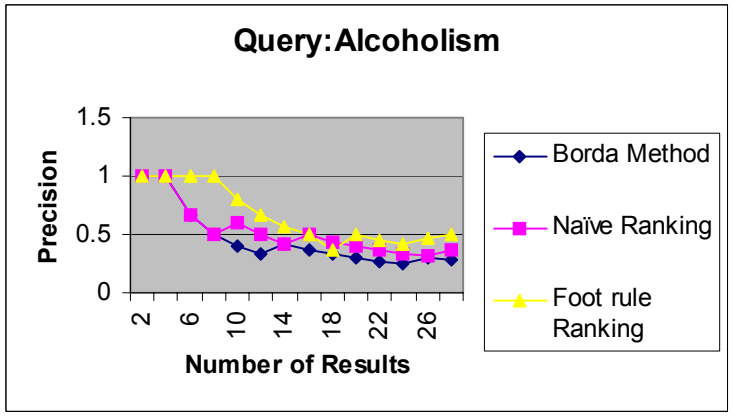
We have plotted the precision of the ranking strategies with respect to both the number of search results and the recall.

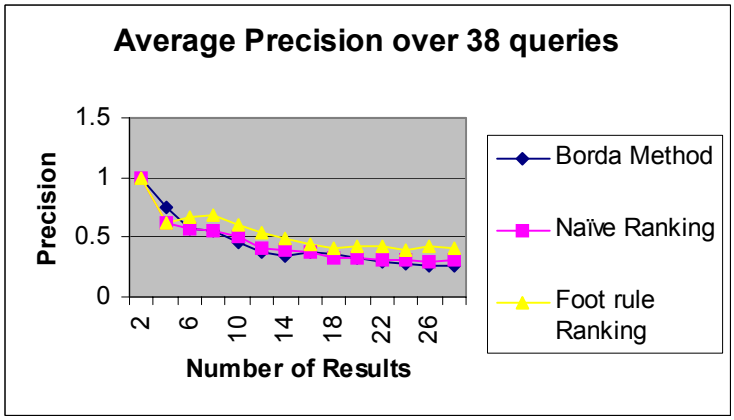
In considering the recall, we have taken the total number of relevant documents based on user evaluation of all the top 10 results retrieved by each search engine. The recall is calculated as the number of relevant documents retrieved/ total number of relevant results thus judged.

We have taken the relevance feedback from two different judges. The Kappa measure of this relevance feedback is 0.78. In the following graphs, we present the results for two out of the 38 queries run. We also present the average of the results obtained over the 38 queries.

6.4.1 Precision with respect to Number of Results returned

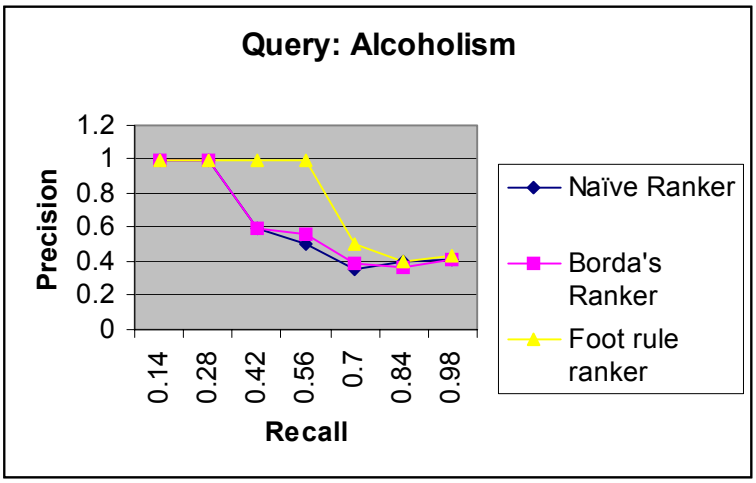


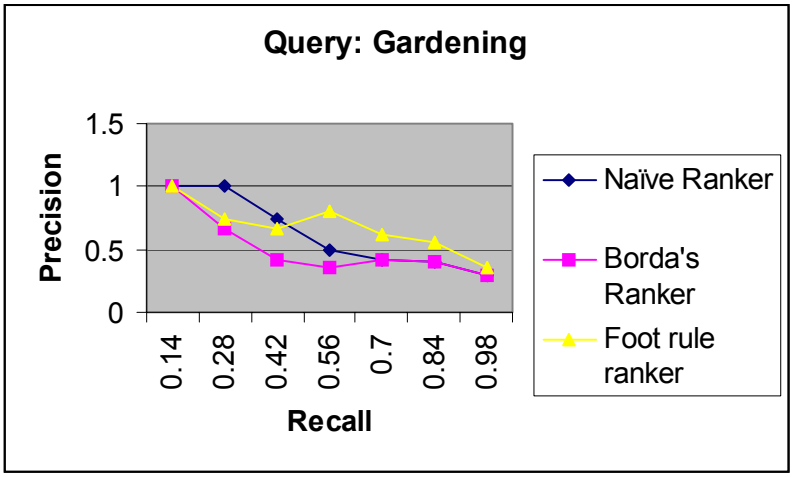




It can be observed that on an average, the footrule distance ranking aggregation method gives better precision for the given set of results. Also, easily computable Borda's method does a good job when compared to the Naïve ranking method.

6.4.2 Precision vs. Recall





A similar observation can be made with respect to the precision at a given recall for each of the ranking strategies.

7.Problems encountered

During the design of the advance search interface, we realized that all the options that normal search engines provide, could not be made available because, each search engine provides a different set of advanced options.

Some of the advanced search options implemented in the different search engines are tabulated below. There are other advanced search options like file format, language specific search which have not been explored as part of this project.

Another major issue we faced was finding an optimal algorithm for implementing minimum cost bipartite matching. We chose to implement the Hungarian method, but in retrospect we think other efficient algorithms would have been better.

Feature	Google	MSN	Altavista	Tadpole
Conjunction	Yes	Yes	Yes	Yes
Disjunction	Yes	Yes	Yes	Yes
Negation	Yes	Yes	Yes	Yes
Phrase Search	Yes	Yes	Yes	Yes
Number of results per page	No (for the API)	No	Yes	No

8.Conclusion and Future Work

In the context of our project, we have studied some trade-offs that are involved in the design of meta-search engines. We have observed that the computational complexity of ranking algorithms used and performance of the meta-search engine are conflicting parameters. A compromise must be achieved between these two, based on the perceived applications and environment in which the meta-search engine will be used.

Future work involves, incorporating more number of search engines in the study, studying the performance for the most popular queries published by the various search engines, incorporate local kemmenization to e spam, to incorporate methods for avoiding mirrored search results.

Bibliography

[1] 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.

[2]Hungarian Method
http://www.math.nus.edu.sg/~matcgh/MA3252/lecture_notes/Hungarian.pdf
http://www.cob.sjsu.edu/anaya_j/HungMeth.htm

[3]http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.html

[4]K. Bharat and M. Henzinger, Improved algorithms for topic distillation in a hyperlinked environment.*ACM SIGIR*, pages 104--111, 1998.

[5]S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins.
Experiments in topic distillation. *Proc. ACM SIGIR Workshop on Hypertext Information Retrieval on the Web*, 1998.

[6]H. P. Young. An axiomatization of Borda's rule. *Journal of Economic Theory*, 9:43--52, 1974.