On Selective Result Merging in a Metasearch Environment

Elizabeth D. Diaz, Arijit De, Vijay V. Raghavan
Center for Advanced Computer Studies,
University of Louisiana, Lafayette.
{edd8747, axd9142, raghavan}@louisiana.edu

Abstract

When a query is passed to multiple search engines, each search engine returns a ranked list of documents. The problem of metasearch is to fuse these ranked lists such that optimal performance is achieved because of the combination. There are two parts of the metasearch process. The first part is to select which search engine results are to be merged. The second part is the actual process of merging. In this paper we propose (1) a strategy for selective merging of results from a metasearch engine and (2) a heuristic for handling missing documents in result sets to be merged to improve the result-merging process Our experimental results will show that our proposed strategy for selection before merging coupled by incorporating it into the BORDA method improves the performance of merging.

1. Introduction

A metasearch engine is a system that supports unified access to multiple existing search engines. When a user submits a query to the metasearch engine, it selects a few promising search engines, from a larger set of underlying search engines, to which it will dispatch the query.

The search engines return results in the form of ranked lists. The metasearch engine extracts and selects results from the returned ranked lists, and merges the selected results into a single ranked list. In short, the metasearch engine needs to select search engines whose results (represented as ranked lists) with respect to a certain query need to be merged.

In this paper, we provide a strategy to select search engines whose results need to be merged. Our proposed strategy for selecting search engine results revolves around distances between ranked lists. Given a finite set of ranked lists, we perform computations such as distance among rankings and clustering of rankings. The results of these computations are clusters of rankings with respect to a given query. We utilize distance functions introduced in [6] for the clustering of search engine results in order to perform selective merging of rankings. The motivation for selecting search engines based on distances, is to ensure that we merge results that rank documents differently. In this way we are able to combine the diverse opinions of various search engines with respect to the way in which they rank documents. Another motivation is to remove redundancy between search engines at the time of merging.

A missing document is a document that has been retrieved by some search engines but not by all. A document might be missing from a ranked list if the search engine does not retrieve it, if the search engine does not index it or if the search engine does not cover the document. Our research focuses on the need to come up with one or more heuristic by which we can compute the position of each missing document in the ranked list where it is missing. By doing so we can insert missing documents into the ranked list and thereby obtain a more homogenous environment for merging.

We also focus our attention on comparing three new heuristics for handling missing documents in ranked lists that are returned by a search engine in response to a given query. In this paper we propose three heuristics (H1, H2 and H3) to handle missing documents.

The data sets used in our experiments were the TREC datasets, TREC 3, TREC 5, TREC 9 and Vogt. We used recall-based precision as the measure for comparing the effects that the three heuristics and the selection strategy had. Our strategy for selection and heuristics for handling missing documents were used in conjunction with the BORDA method.

As part of our experiments, we compared the performance when merging pre-selected search engines (based on our proposed selection strategies) using the method where missing documents were handled based on our proposed heuristics to the simple BORDA method based on the model proposed by Aslam and Montague [1]. BORDA with selection and missing document heuristics perform significantly better than the simple BORDA. Figure 1 is a block diagram of the representation of the metasearch process as envisioned by us.

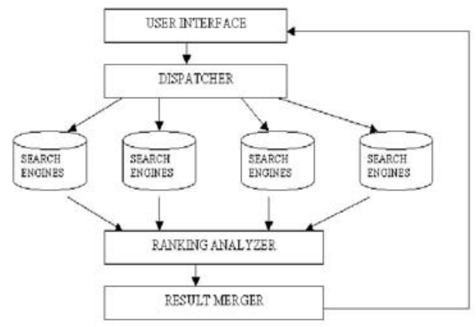


Figure 1: Block Diagram of the Metasearch process

The user interface captures the query from the user and the dispatcher sends the query to a series of search engines. Each search engine returns the results of the query in the form of ranked lists. These ranked lists are passed to the ranking analyzer. The ranking analyzer employs the selection strategy proposed by us to select the search engines whose results need to be merged. The ranked lists from these search engines are passed into the result merger. Missing document heuristics are applied in the result merger while the ranked lists are merged into a single final ranked list that is returned back to the user.

2. Related Work

Researchers and scientists working in the field of metasearch and distributed information retrieval have explored data fusion techniques for result merging. Thus a number of data fusion based models have been developed. To test the effects of our heuristics for handling missing document and strategy for selection of search engines before merging we use the basic BORDA model. In this section, we describe the BORDA model as proposed by Aslam and Montague [1].

2.1. Borda-Fuse Model & Weighted Borda-Fuse

Aslam and Montague proposed two models [1]. The first one is called Borda-Fuse model and it is based on a political election strategy named Borda Count. The Borda-Fuse works by assigning points to each document in each one of the lists to be merged. The number of points per documents depends on the rank position of the document, i.e.,, for a list of n ranked documents, the top document receives n points, the next document receives n – 1 and so on. The points assigned for a given document by different search engines are added up and the documents are ranked from highest to lowest according to the sum. The Borda count for document d_i is $\sum_k (n - r_{ik})$ where n is the number of documents and r_{ik} is the rank position of document d_i under search engine k. This model does not require training data or the RSVs and its algorithm is simple and effective. It has been shown that the Borda Count is optimal [7, 8] when compared to standard voting methods. However, in [4], it had been demonstrated that the Borda has limitations with respect to the Condorcet Principle, Condorcet Order and the Increasing and Decreasing Principles.

Their second method, the Weighted Borda-Fuse, is a weighted version of the Borda-Fuse. A weight w_i is assigned to the i^{th} search engine according to their performance. Weighted Borda-Fuse requires training to determine the best weights for the performance of the search engines. This method was shown to perform better than the Borda-Fuse.

3. Handling Missing documents

In this section, we describe a heuristic for handling missing documents. First we define the concept of positional values.

3.1. Positional Values

Positional Value: The positional value (PV) of a document d_i in the resulting list l_k returned by a search engine s_k is defined as $(n - r_{ik} + 1)$ where r_{ik} is the rank of d_i in search engine s_k and n is the total number of documents in the result.

3.2. Case of Missing Documents

Let PV_i be the positional values for a document d in the i^{th} search engine. Let m be the total number of search engines. Let r be the number of search engines in which d appears. Let j denote a search engine not among the r search engines where d appears. Our heuristics are:

H1: For all j, $PV_j = \frac{\sum_{i=1}^{r} PVi}{r}$, i.e., average of the positional values of the document in the r search engines.

H2: For all j, $PV_j = \frac{\sum_{i=1}^{m} PVi}{m}$, i.e., the PV_j is the average of the positional values of the document in the m search engines where d appears.

H3: For all j, $PV_j = \min\{PV_i\}$ where $1 \le i \le r$, i.e., the minimum of the positional values of the document among the r search engines where d appears.

4. Proposed selection strategy

In the previous sections, we stated the heuristic for missing documents. Handling missing documents is an important part of the metasearch environment. However to improve the effectiveness of metasearch we can pre-select the search engines whose results we need to merge based on some strategies. In this section, we discuss our proposed approaches for selecting search engines.

4.1. Strategy of merging without selection

As the title suggests in this strategy we select search engines randomly. There is no specific strategy for selection.

4.2. Strategy of selective merging

In this section, we propose a method for selecting search engines based on the distances between the ranked lists obtained for a specific query. Distance computing measures are discussed in the next section. After distances are computed, we propose merging the search engine pair that has the maximum distance first. We call this "Farthest SE-Pair First" strategy. The rational behind, selecting the Search Engines that are farthest apart first, is to ensure that we merge results that rank documents differently. Thereby we are able to add variation to the results that are being merged.

4.3. Distance measures

To employ the "Farthest SE-Pair First" strategy we need to compute distances between the ranked lists returned for a given query by various search engines. By doing so we can measure the distanced between search engines in the context of a particular query. We use the distance function proposed in [6] with some slight modification. Let R_1 and R_2 be two rankings. Let Δ_1 be the document set for ranking R_1. Let Δ_2 be the documents set for ranking R_2. Let $\Delta_3 = \Delta_1 \cap \Delta_2$. Suppose that Ψ_1 and Ψ_2 are rankings of Δ_3.

If D and D' are in Δ_3 then the function is defined as

$$\delta_{\Psi_1,\Psi_2}(D,D') = \begin{cases} 0 & \rightarrow agree \\ 1 & \rightarrow onerankshigher, othertie \\ 2 & \rightarrow inverted \end{cases}$$

The ranking distance between Ψ_1 and Ψ_2, $d(\Psi_1, \Psi_2)$ can be defined by the expression

$$\frac{1}{|\Delta_3|(|\Delta_3|-1)} \sum_{(D,D')\in\Delta_3} \delta_{\psi 1,\psi 2}(D,D')$$

Once the distances have been calculated, the distance matrix can be defined.

Example: Calculating distance among rankings. Suppose we have 2 ranking list Ψ_1, Ψ_2 , from two different search engines. These two rankings have been pretreated with some kind of heuristic (H1, H2, H3).

$$\psi_1 = \begin{bmatrix} d_2 \\ d_1 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix}, \psi_2 = \begin{bmatrix} d_2 \\ d_5 \\ d_3 \\ d_1 \\ d_4 \end{bmatrix}$$

δ $(d_2, d_1) = 0$, δ $(d_2, d_3) = 0$,

$\delta(d_2, d_4) = 0$, $\delta(d_2, d_5) = 0$,
$\delta(d_1, d_3) = 2$, $\delta(d_1, d_4) = 2$,
$\delta(d_1, d_5) = 0$, $\delta(d_3, d_4) = 0$,
$\delta(d_3, d_5) = 2$, $\delta(d_4, d_5) = 2$,

$$d(\Psi_1, \Psi_2) = \frac{8}{5*4} = 0.4$$

4.4. Selection based on maximum distances: The "Farthest SE-Pair First" strategy

In this section, we provide an algorithm for the selection strategy mentioned in section 4.2.
Input: A set of search engines S. $S = \{SE_i \mid \forall i;\ 1 \le i \le n$ and n is the number of search engines underlying the meta search engine$\}$.
Two sets PICKED and NPICKED (non picked). PICKED = $\{ SE_i \mid SE_i\ \varepsilon\ S\ \&\ SE_i$ has been selected for merging $\}$. NPICKED = $\{ SE_i \mid SE_i\ \varepsilon\ S\ \&\ SE_i$ has not been selected for merging $\}$.
Distance matrix: DM is a matrix that contains the distance between the search engines based on the ranked lists obtained by querying it.
DM = { DM(i, j) | DM(i,j) is the distance between SE_i and SE_j }.
Number of search engines to be merged: k
Output: The set PICKED.
Stopping Condition: When the | PICKED | = k where | PICKED | is the size of the set PICKED.
Algorithm: Initially PICKED = ϕ. NPICKED = S. Select the element DM(i,j) of DM where DM(i,j) is maximum.
Select SE_i and SE_j for merging and place them in PICKED.
For each search engine SE_i in set PICKED, access its distance to every search engine SE_p that is in set NPICKED by referring to distance matrix, DM, and pick the DM(i,p) that has the maximum value over all i and p values. Remove SE_p from NPICKED and add it to picked. Ties are broken arbitrarily.
Repeat step 4 until the size of set PICKED is k.

5. Experiments

In this section, we describe the various aspects of our experiments.

5.1. Objectives

The objectives of our experiments were to (1) study the effect of our selection strategy on the BORDA method and (2) to explore how the heuristics for handling missing documents affected the performance of the BORDA method when used in conjunction with or without the selection strategy.

5.2. Procedure

In this section we describe experimental procedures for merging when no selection strategy is employed and when our proposed selection strategy is employed.

5.2.1. Experimental setup for "Randomly Select Search Engines"

Input: (1)A set of queries Q numbered 1 through n where n is the number of queries. (2) A set of search engines S. (3) Missing document heuristic to be applied. (4) Dataset to be used e.g., TREC3, TREC5 and TREC9.
Output: Average precision obtained by merging ranked list using BORDA.
Procedure: (A) For each query q in Q do the following procedure.
(1)Initialize a two dimensional matrix A[11]. Each element represents RB-precision values for a ranked list obtained by merging a certain number of search engines. Thus each element holds the value of RB-precision for a ranked list obtained when using a specific method to merge a specific number of search engines.
(2)Varying m from 2 through 12 do the following (a) pick m search engines randomly (b)pass the query q to the m search engines picked randomly and obtain results in the form of ranked lists.(c) Merge these ranked lists into one list using each of the methods BORDA, obtaining a single merged list called RBORDA. (d) Compute RB-precision for each of the merged lists for Recall values of 0.25, 0.5, 0.75 and 1.00. Average the RB-precision values thus obtained to consolidate them into a single average value. Thus we obtain a single value of precision, PRBORDA, for the list RBORDA . Let A[m-1]= PRBORDA. (e) Accumulate over m. Repeat steps a through d 50 times and average out the results.
(B) Accumulate over queries and then average by the number of queries.

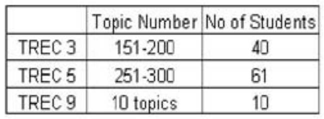
5.2.2 Experimental setup for "Farthest SE-Pair First"

The experimental procedure followed was the same. However instead of selecting search engines randomly (as in A(2).a), in this case we select search engines based on the strategy proposed in section 4.4. Thus, for a given m, only single search engines are considered.

5.3. Implementation

Our experiments were done using programs written in Visual Basic programming language that queried a Microsoft Access Database that held the data exported from TREC 3, TREC 5 and TREC 9 datasets.

For each experiment, we need to input the method name (in our case we have only one method), the strategy for selection and heuristic for selection.



	Topic Number	No of Students
TREC 3	151-200	40
TREC 5	251-300	61
TREC 9	10 topics	10

Table 1(a): Description of Data Sets.

5.4. Data Sets

Table 1 shows the particulars of the data sets TREC 3, TREC 5 TREC 9 and Vogt that are used. Each of the data sets has a specified number of systems that return up to 1000 documents when queried with a certain topic. There are 50 topics in each of the data sets. Each topic is analogous to a query and each system is analogous to a search engine. Thus, topics (queries) are passed onto a system (search engine). The search engines then return a set of documents in the form of a ranked list. Each document is either relevant (represented by 1), highly relevant (represented by 2) or irrelevant (represented by 3).

The comparative results of various experiments were tabulated. Each column in the represents a set of results obtained for a specific experimental case. Table 1(b) shows the symbols used in column headings in the tables to describe the experiments.

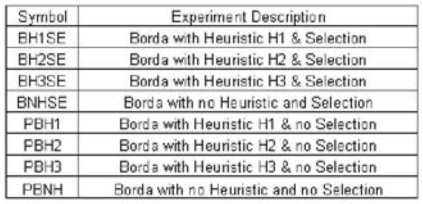
5.5. Performance Metrics

Our metric for measuring performance is Recall Based Precision. The detailed theory of Recall Based (RB) precision can be found in [5]. Recall based precision is used in case when there are a series of documents ranked in partial order and we need to find out the precision for various levels of recall.

The formula is shown below

$$\frac{x*n}{x*n+j+s*\frac{i}{r}}$$

where
(1) x is one of the standardized recall values i.e., 0.25, 0.5, 0.75, etc; (2) n is the number of relevant documents in the collection; (3) s is the number of relevant document wanted; (4) i is the number of irrelevant documents in the final rank; (5) r is the number of relevant documents in the final rank; (6) j is the number of irrelevant documents to get to s documents.
In this context, final rank is defined as the rank containing or completing the number of relevant documents as specified by s.



Symbol	Experiment Description
BH1SE	Borda with Heuristic H1 & Selection
BH2SE	Borda with Heuristic H2 & Selection
BH3SE	Borda with Heuristic H3 & Selection
BNHSE	Borda with no Heuristic and Selection
PBH1	Borda with Heuristic H1 & no Selection
PBH2	Borda with Heuristic H2 & no Selection
PBH3	Borda with Heuristic H3 & no Selection
PBNH	Borda with no Heuristic and no Selection

Table 1(b): Symbols describing experiments.

5.6. Comparison of missing document heuristics.

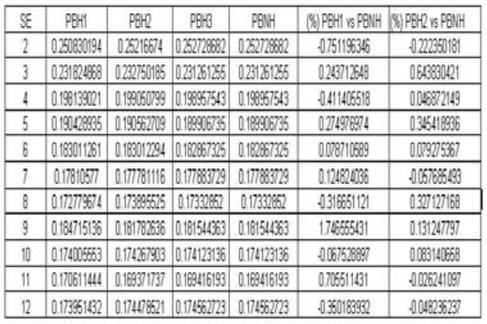
In this set of experiments, we have two cases
In case 1, we compare the performance of the BORDA algorithm, when each of the three heuristics for missing documents (H1, H2 and H3) proposed are applied in conjunction with the selection strategy.

In case 2, we also compare the performance of the BORDA algorithm, when each of the three heuristics for missing documents (H1, H2 and H3) proposed are applied without any the selection strategy.

5.6.1. Case 1

Tables 2(a), 2(b), 2(c) shows the performance of the BORDA algorithm for the data sets TREC 9, TREC 5, and TREC 3. The first column shows the number of search engines being merged. The second, third and fourth columns named BH1SE, BH2SE, BH3SE and show results when heuristic H1, H2 and H3 are applied. The fourth column named BHNSE represents results of experiments in which the selection strategy was employed but no heuristics was applied. Column 9,10,11 shows the improvement effects of heuristic H1, H2 and H3, respectively, in comparison to the case when no heuristic was applied.

<u>TREC 9:</u> Table 2(a) shows the results for TREC 9. From the table 2(a) the following observations can be drawn up: (1) Heuristic H1 improved upon the case of no heuristic by up to 18% in some cases. (2) Heuristic H2 improved by 2.5% in some cases. (3) Notice that Heuristic H3 did not effect the merging performance. The results are almost identical for the cases BNHSE and BH2SE (4). For each of the cases in which a heuristic is used and the case in which no heuristic is used the performance measure seem to go down as we vary the number of search engines from 2 to 5. Then the performance improves as we vary the number of search engines from 5 to 6. Beyond that if the number of search engines is increased the performance goes down.



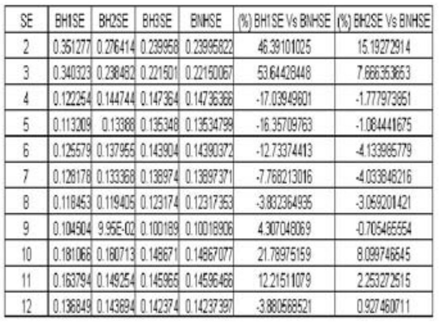
SE	PBH1	PBH2	PBH3	PBNH	(%) PBH1 vs PBNH	(%) PBH2 vs PBNH
2	0.250830194	0.25216674	0.252728682	0.252728682	-0.751196346	-0.222350181
3	0.231824868	0.232750185	0.231261255	0.231261255	0.243712648	0.643830421
4	0.198139021	0.199050799	0.198957543	0.198957543	-0.411405518	0.046872149
5	0.190428935	0.190562709	0.189906735	0.189906735	0.274976974	0.345418936
6	0.183011261	0.183012294	0.182867325	0.182867325	0.078710589	0.079275367
7	0.17810577	0.177781116	0.177883729	0.177883729	0.124824036	-0.057685493
8	0.172779674	0.173895525	0.17332852	0.17332852	-0.316651121	0.327127168
9	0.184715136	0.181782636	0.181544363	0.181544363	1.746555431	0.131247797
10	0.174005653	0.174267903	0.174123136	0.174123136	-0.067528897	0.083140658
11	0.170611444	0.169371737	0.169416193	0.169416193	0.705511431	-0.026241097
12	0.173951432	0.174478521	0.174562723	0.174562723	-0.350183932	-0.048236237

Table 2(a): Comparing performance of Heuristics when selection strategy is employed for TREC 9

<u>TREC 5:</u> Table 2(b) shows the results for TREC 5. From the table 2(b) the following observations can be drawn up: (1) Heuristic H1 performs best effecting the performance of metasearch by about 46% in some cases. (2) The effect of Heuristic H2 once again is somewhat limited at about 7-8% (3) As in case of TREC 9 applying heuristic H3 has the same effect as applying no heuristic at all. The results are almost identical for the cases BNHSE and BH2SE (4) Overall performance tends to decrease with the increase in number of search engine results being merged till about 8 search engines after which the performance improves.

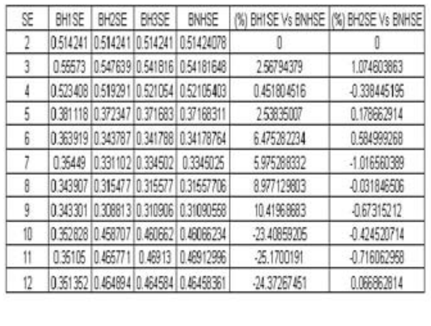
Heuristic H1 is most effective in improving performance of the merging algorithm. Heuristic H2 is less effective and H3 has no effect on the merging algorithm at all.

<u>TREC 3:</u> Table 2(c) shows the results for TREC 3. Our observations were similar to TREC 5. (1) Heuristic H1 performs fairly well when the number of search engines being merges is less that 10. (2) Heuristic H2 effects the performance nominally. In certain cases the effect is adverse and in some case the effect is positive. (3) Applying heuristic H3 has the same effect as applying no heuristic at all. (5) Overall performance tends to decrease with the increase in number of search engine results being merged until about 8 search engines after which the performance improves.



SE	BH1SE	BH2SE	BH3SE	BNHSE	(%) BH1SE Vs BNHSE	(%) BH2SE Vs BNHSE
2	0.351277	0.276414	0.239958	0.23995822	46.39101025	15.19272914
3	0.340323	0.238482	0.221501	0.22150067	53.64428448	7.666353663
4	0.122254	0.144744	0.147364	0.14736366	-17.03949601	-1.777973851
5	0.113209	0.13388	0.135348	0.13534798	-16.35709763	-1.084441675
6	0.125579	0.137955	0.143904	0.14390372	-12.73374413	-4.133985779
7	0.128178	0.133368	0.138974	0.13897371	-7.768213016	-4.033848216
8	0.118453	0.119405	0.123174	0.12317353	-3.832364935	-3.059201421
9	0.104504	9.95E-02	0.100189	0.10018906	4.307048069	-0.705465554
10	0.181066	0.180713	0.148671	0.14867077	21.78975159	8.099746645
11	0.163794	0.149254	0.145965	0.14596466	12.21511079	2.253272515
12	0.136849	0.143894	0.142374	0.14237397	-3.880568521	0.927460711

Table 2(b): Comparing performance of Heuristics when selection strategy is employed for TREC 5



SE	BH1SE	BH2SE	BH3SE	BNHSE	(%) BH1SE Vs BNHSE	(%) BH2SE Vs BNHSE
2	0.514241	0.514241	0.514241	0.51424078	0	0
3	0.55573	0.547639	0.541816	0.54181648	2.56794379	1.074603863
4	0.523408	0.519291	0.521054	0.52105403	0.451804516	-0.338445195
5	0.381118	0.372347	0.371683	0.37168311	2.53835007	0.178862914
6	0.363919	0.343787	0.341788	0.34178764	6.475282234	0.584999268
7	0.35449	0.331102	0.334502	0.3345025	5.975288332	-1.016560389
8	0.343907	0.315477	0.315577	0.31557706	8.977129803	-0.031846506
9	0.343301	0.308813	0.310906	0.31090558	10.41968683	-0.67315212
10	0.352828	0.458707	0.460662	0.46066234	-23.40859205	-0.424520714
11	0.35105	0.465771	0.46913	0.46912996	-25.1700191	-0.716062958
12	0.351352	0.464894	0.464584	0.46458361	-24.37267451	0.066862814

Table 2(c): Comparing performance of Heuristics when selection strategy is employed for TREC 3

5.6.2. Case 2

Table 3(a), 3(b), 3(c) shows the performance of the BORDA algorithm for the data sets TREC 9, TREC 5, and TREC 3. The first column shows the number of search engines being merged The second, third and fourth columns named PBH1, PBH2, PBH3 and show results when heuristic H1, H2 and H3 are applied. The fourth column named PBNH represents results of experiments in which the selection strategy was employed but no heuristics was applied. Column 9,10,11 shows the improvement effects of heuristic H1, H2 and H3, respectively, in comparison to the case when no heuristic was applied.

SE	PBH1	PBH2	PBH3	PBNH	(%) PBH1 vs PBNH	(%) PBH2 vs PBNH
2	0.250830194	0.25216674	0.252728682	0.252728682	-0.751196346	-0.222350181
3	0.231824868	0.232750185	0.231261255	0.231261255	0.243712648	0.643830421
4	0.198139021	0.199050799	0.198957543	0.198957543	-0.411405518	0.046872149
5	0.190428936	0.190562709	0.189906735	0.189906735	0.274976974	0.345418936
6	0.183011261	0.183012294	0.182867325	0.182867325	0.078710689	0.079275367
7	0.17810577	0.177781116	0.177883729	0.177883729	0.124824036	-0.057685493
8	0.172779674	0.173895525	0.17332852	0.17332852	-0.316651121	0.327127168
9	0.184715136	0.181782636	0.181544363	0.181544363	1.746555431	0.131247797
10	0.174005553	0.174267903	0.174123136	0.174123136	-0.067528897	0.083140658
11	0.170611444	0.169371737	0.169416193	0.169416193	0.705511431	-0.026241097
12	0.173951432	0.17447821	0.174562723	0.174562723	-0.350183932	-0.048236237

Table 3(a): Comparing performance of Heuristics when no selection strategy (random selection) is employed for TREC9

<u>TREC 9:</u> Table 3(a) shows the results for TREC 9. Table 3(a) show how each missing document heuristic effects the performance when no selection strategy is employed before merging. From the table, we clearly observe that heuristic H1 and H2 have only slight positive effect on the process of merging. Heuristic H3 has virtually no effect on performance.

SE	PBH1	PBH2	PBH3	PBNH	(%) PBH1 vs PBNH	(%) PBH2 vs PBNH
2	0.15188	0.149433	0.155284	0.155284	-2.192073741	-3.767948896
3	0.130171	0.127782	0.127883	0.127883	1.789251664	-0.079053778
4	0.119623	0.124124	0.126352	0.126352	-5.325427438	-1.763278582
5	0.118852	0.125291	0.125984	0.125984	-5.661753233	-0.55052975
6	0.121126	0.124459	0.124119	0.124119	-2.411382981	0.273845621
7	0.110652	0.112071	0.111827	0.111827	-1.050176349	0.218512739
8	0.115969	0.120143	0.119273	0.119273	-2.769940823	0.729358966
9	0.109459	0.110626	0.111613	0.111613	-1.929548309	-0.884412391
10	0.117286	0.12102	0.120953	0.120953	-3.03175941	0.055123905
11	0.104387	0.108657	0.109984	0.109984	-5.08820152	-1.206209406
12	0.107937	0.110942	0.112675	0.112675	-4.205373756	-1.538045406

Table 3(b): Comparing performance of Heuristics when no selection strategy (random selection) is employed for TREC 5

<u>TREC 5:</u> Table 3(b) shows the results for TREC 5. Table 3(b) show how each missing document heuristic effects the performance when no selection strategy is employed before merging. In case of TREC 5 the performance is adversely effected when missing documents are handled using heuristic H1 and H2. In case of H1 the effect is as significant as 5% in some cases. In case of heuristic H2 the effect is almost negligible.

SE	PBH1	PBH2	PBH3	PBNH	(%) PBH1 vs PBNH	(%) PBH2 vs PBNH
2	0.52116	0.524217	0.51935	0.51935	0.348380454	0.936982441
3	0.560275	0.557998	0.551531	0.551531	1.58547768	1.172711833
4	0.490884	0.499751	0.50276	0.50276	-2.362287882	-0.598467262
5	0.439323	0.451641	0.453906	0.453906	-3.212896091	-0.499153294
6	0.448269	0.465727	0.469344	0.469344	-4.490318535	-0.770690564
7	0.441351	0.458343	0.456938	0.456938	-3.411257774	0.307521135
8	0.436542	0.455136	0.45802	0.45802	-4.689170498	-0.629561983
9	0.455396	0.468804	0.469341	0.469341	-2.971170544	-0.11444257
10	0.410279	0.421591	0.422523	0.422523	-2.897760049	-0.220453796
11	0.416299	0.424384	0.424478	0.424478	-1.926864653	-0.022126705
12	0.396	0.416228	0.418124	0.418124	-5.291401045	-0.453516105

Table 3(c): Comparing performance of Heuristics when no selection strategy (random selection) is employed for TREC 3

<u>TREC 3:</u> Table 3(c) shows the results for TREC 3. Table 3(c) show how each missing document heuristic effects the performance when no selection strategy is employed before merging. Results are almost identical to that of TREC 5.

5.7. Comparison of BORDA with and without selection strategy.

In this set of experiments, we compare the performance of the BORDA method when we employ a selection strategy before merging results to the performance of the BORDA method where no prior selection is done. In this case, no heuristics are employed for handling missing documents.

SE	BNHSE	PBNH	% Improvement
2	0.24208	0.252729	-4.396447454
3	0.245596	0.231261	5.836885023
4	0.245596	0.198958	18.990053
5	0.221579	0.189907	14.29388472
6	0.268051	0.182867	31.7788461
7	0.24447	0.177884	27.23702524
8	0.230523	0.173329	24.81070683
9	0.207476	0.181544	12.4986131
10	0.207014	0.174123	15.8880553
11	0.206902	0.169416	18.11783354
12	0.206729	0.174563	15.5595243

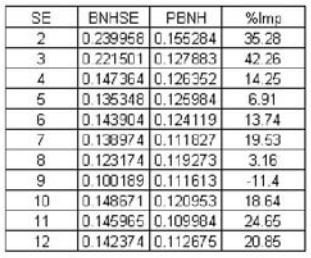
Table 4(a): Selection vs. no (random) selection for

TREC 9

<u>TREC 9:</u> Table 4(a) shows the results when merging with and without selection. In this comparison, we do not apply any heuristics for handling missing documents. The performance is significantly better when our selection strategy is employed. The improvements when 6 search engines are merged are about 31%. Table 4(a) shows the improvements.

<u>TREC 5:</u> Table 4(b) shows the results for TREC 5. In this comparison, we do not apply any heuristics for handling missing documents. The

performance is significantly better when our selection strategy is employed. In the best case, improvement is up to 35%. On the average 20% improvement is observed. Table 4(b) shows the improvements.



SE	BNHSE	PBNH	%Imp
2	0.239958	0.155284	35.28
3	0.221501	0.127883	42.26
4	0.147364	0.126352	14.25
5	0.135348	0.125984	6.91
6	0.143904	0.124119	13.74
7	0.138974	0.111827	19.53
8	0.123174	0.119273	3.16
9	0.100189	0.111613	-11.4
10	0.148671	0.120953	18.64
11	0.145965	0.109984	24.65
12	0.142374	0.112675	20.85

Table 4(b): Selection vs no (random) selection for TREC 5

6. Conclusions

In our paper we have dealt with two problems pertaining to merging of results in the context of metasearch. The first one pertained to missing documents and second one was pertaining to selection of search engines that need to be queried before merging. We proposed three heuristics for handling missing documents and a strategy for selecting search engines to be merged based on the distances between the ranked lists of results they returned for certain queries. When merging search engines at random, our heuristics for handling missing documents had no effect on the performance of the merged list. However when applied in conjunction with the selection strategy the average precision of the resulting merged list was greatly improved. Selection before merging applied independently of missing document heuristics also resulted in significant improvements.

7. References

[1] J. A. Aslam, M. Montague, Models for Metasearch, Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, United States, September 2001, pp. 276-284.

[2] W. Meng, C. Yu, K. Liu, Building Efficient and Effective Metasearch engines, ACM Computing Surveys, March 2002, pp. 48-84.

[3] W. Meng, C. Yu, K. Liu. A Highly Scalable and Effective Method for Metasearc*h,* ACM Transactions on Information Systems, July 2001 pp. 310-335.

[4] J. R. Parker, Multiple Sensors, Voting Methods and Target Value Analysis, Computer Science Technical Report, 1998, February 1, 1998, University of Calgary, Laboratory for Computer Vision, pp. 615-06.

[5] P. Bollmann, V. V. Raghavan, G. S. Jung, and L. C. Shu. On probabilistic notions of precision as a function of recall. *Information Processing and Management*, May-June 1992, Vol. 28:291--315.

[6] V. Raghavan, H. Sever, On The Reuse Of Past Optimal Queries, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, July 9-13, 1995, pp. 344-350,.

[7] F. Roberts, *Discrete Mathematical Models*, Prentice Hall, Inc., 1976.

[8] F. Zachary, Lansdowne, Outranking Methods for Multicriterion Decision Making: Arrow's and Raynaud's Conjecture"; Social Choice and Welfare; Vol. 14, No. 1; January, 1997; 125-128; #2431.