Models for Metasearch*

Javed A. Aslam
Department of Computer Science
Dartmouth College
6211 Sudikoff Laboratory
Hanover, NH 03755
jaa@cs.dartmouth.edu

Mark Montague
Department of Computer Science
Dartmouth College
6211 Sudikoff Laboratory
Hanover, NH 03755
montague@cs.dartmouth.edu

ABSTRACT

Given the ranked lists of documents returned by multiple search engines in response to a given query, the problem of *metasearch* is to combine these lists in a way which optimizes the performance of the combination. This paper makes three contributions to the problem of metasearch: (1) We describe and investigate a metasearch model based on an optimal democratic voting procedure, the Borda Count; (2) we describe and investigate a metasearch model based on Bayesian inference; and (3) we describe and investigate a model for obtaining upper bounds on the performance of metasearch algorithms. Our experimental results show that metasearch algorithms based on the Borda and Bayesian models usually outperform the best input system and are competitive with, and often outperform, existing metasearch strategies. Finally, our initial upper bounds demonstrate that there is much to learn about the limits of the performance of metasearch.

1. INTRODUCTION

Numerous search systems have been developed both in academia [33] and in industry (Google, Alta Vista, Lycos, HotBot, etc.). In practice, no one system performs "better" than each of the others under all circumstances, and the "best" system for a particular task may not be known *a priori*. This being the case, metasearch engines (such as MetaCrawler, ProFusion, SavvySearch, MetaFerret, InFind, etc.) have been introduced which query a number of search engines, merge the returned lists of pages, and present the resulting ranked list to the user.

This after-the-fact combining of "complete" search engines can be called *external* metasearch. Alternatively, metasearch offers a systematic way of *internally* incorporating all of the various types of evidence available within a single search engine. For example, in the context of web page retrieval, many sources of information exist: each page has text, in-links, out-links, images, tags, keywords, and structural information. For each of these elements, numerous indexing and retrieval algorithms may exist. Metasearch can be used to easily and automatically take advantage of the information provided by these disparate retrieval components.

Potential benefits of metasearch include:

Improved recall: *Recall* is the ratio of retrieved relevant documents to total relevant documents. Fusion can improve recall as long as the input systems are retrieving different relevant documents [17].

Improved Precision: *Precision* is the ratio of retrieved relevant documents to retrieved documents. Lee [13] argues that an "unequal overlap property" holds in ranked list fusion: different retrieval algorithms retrieve many of the same relevant documents, but different irrelevant documents. Ng and Kantor [17] conclude with Lee that if this is true, any fusion technique that more heavily weights common documents should improve precision. Vogt [28] calls this the "chorus effect."

Consistency: Selberg and Etzioni [22] show that current Web search engines often respond to the same query very differently over time. Inasmuch as fusing is an averaging procedure, we can expect the idiosyncrasies of any one system to be smoothed out, providing more reliable behavior [15].

Modular Architecture: From the system design perspective, metasearch techniques allow a large, monolithic search engine to be decomposed into smaller, more specialized modules which can be queried in parallel and then fused.

In short, metasearch holds the promise of obtaining better results than the best underlying retrieval system.

Current metasearch techniques can be characterized by the data they require: whether they need relevance scores or only ranks, and whether they require training data or not. (See Figure 1.) A number of researchers have attempted to characterize when metasearch will yield good performance. The work of Ng and Kantor [17, 16] in this area, as well as Vogt et al. [31, 28], essentially support the conclusion of Croft [6]: The systems being combined should (1) have compatible outputs (e.g., on the same scale), (2) each produce accurate estimates of relevance, and (3) be independent of each other.

In this paper, we describe and investigate two models for the problem of metasearch, one based on democratic election strategies and another based on Bayesian inference. Unlike most existing metasearch algorithms, our techniques require rank information alone; the relevance scores assigned to re-

*This work generously supported by NSF grants EIA-9802068, BCS-9978116, and Career Award CCR 00-93131.

SIGIR'01, September 9-12, 2001, New Orleans, Louisiana, USA..

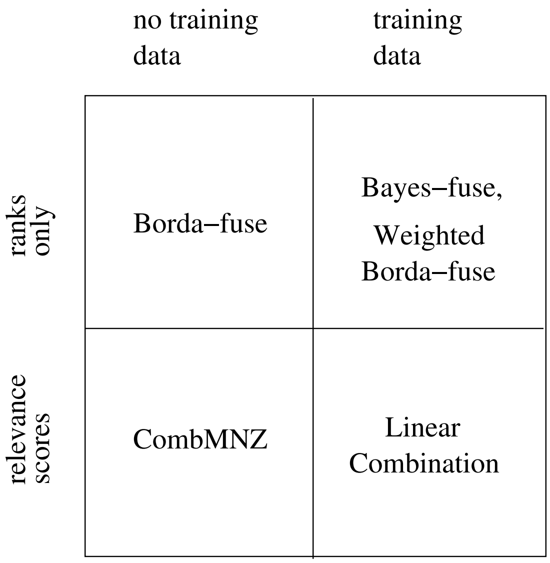

Figure 1: Metasearch techniques may or may not require relevance scores, and may or may not require training. We discuss each of the given methods in the paper.

trieved documents by the underlying retrieval systems are not required. In many real-world situations, these relevance scores are not available (consider most search engines on the World Wide Web). We also describe and investigate a model for obtaining upper bounds on the performance of metasearch strategies.

The first model we shall discuss, *Borda-fuse*, is based on an optimal voting procedure, the Borda Count. Metasearch algorithms based on the Borda Count have a number of advantages: (1) They require no training data; (2) they do not require relevance scores; and (3) they are surprisingly simple and efficient. Our experiments with *Borda-fuse* show that its performance is competitive with even the best existing metasearch strategies which typically require access to relevance scores which are not often available. We also describe a weighted version of this algorithm which requires minimal training but whose performance improves.

The second model we shall discuss, *Bayes-fuse*, is based on a Bayesian analysis of the odds of document relevance. Metasearch algorithms based on this model require training data, but they have the following advantages: (1) They do not require relevance scores; (2) they are based on a rigorous, probabilistic model of the problem; and (3) they can be implemented in both a simple, naive manner that assumes search engine independence and in a sophisticated manner that accounts for dependencies. Our experiments with *Bayes-fuse* show that its performance is competitive with even the best existing metasearch strategies which typically require access to relevance scores and that its performance can significantly exceed the best existing strategies when combining the results of disparate systems.

Finally, we propose a model for obtaining upper bounds on the performance of metasearch algorithms based on the idea of a *constrained oracle*. Given a set of ranked lists to merge, we assess the performance of an omniscient metasearch oracle that knows which documents are relevant and which are not but is constrained in how it may rank these documents. The constraints are chosen so as to capture natural limitations of any reasonable metasearch algorithm. Our experiments reveal a sizable gap between the oracle's performance and the performance of current metasearch algorithms.

In the remainder of this work, we first review the relevant literature, noting especially the various metasearch algorithms that have been proposed in the past. We then present the Borda-fuse, Bayes-fuse, and upper bound models, describe a number of implementation issues, and detail the results of a number of experiments on TREC data. Finally, we conclude with directions for future research.

2. RELATED WORK

The use of data fusion to combine document retrieval results has received considerable attention in the past few years: it has been the subject of a number of doctoral dissertations [2, 16, 28, 23], journal articles [25, 10, 30], and conference papers [8, 3, 12, 13, 17, 29, 14], being especially used in the TREC competitions [9, 24, 18]. In this section we review the results of these publications as they relate to our work.

2.1 CEO Model

Thompson [25, 26] proposes a Bayesian model that he calls the Combination of Expert Opinion (CEO) model. It bears some resemblance to our Bayesian model, but also differs in a number of ways. CEO is not really a general-purpose metasearch algorithm, but is a complete search system; it has two basic search components (each driven by Bayesian updating of probabilities) that are fused with a Bayesian formula. The fusion algorithm is not based on odds. And, it requires a full probability distribution to be specified for each document (not simply a probability of relevance). As far as we know, the CEO system has never been implemented.

2.2 Min, Max, and Sum Models

Fox and Shaw [9] fuse based on the unweighted min, max, median, or sum of each document's normalized[1] relevance scores over the constituent systems. Fox and Shaw also try to more or less heavily weight the number of systems that returned a given document (n_d) by using the formula

$$rel(d) = n_d^\gamma \sum_i rel_i(d)$$

for $\gamma \in \{-1, 0, 1\}$. The sum is over the constituent systems. When $\gamma = -1$, the system is equivalent to the average similarity *over systems that returned* d, "CombANZ". When $\gamma = 0$, the result is the sum of the similarities, "CombSUM" (assuming, as they do, a similarity of zero for documents not returned, this is equivalent to the average similarity over *all* systems—even those that did not return d). And when $\gamma = 1$, the result is the formula they call "CombMNZ" (Multiply-by-number-Non-Zero). They found CombSUM to be slightly more effective than CombMNZ.

Lee [13] performed experiments with Fox and Shaw's algorithms, arguing that "different runs retrieve similar sets of relevant documents but retrieve different sets of non-relevant

[1] In an attempt to make relevance scores from different systems directly comparable, it is common to map the scores into the range [0,1].

documents." He further argues that the CombMNZ combination rule appropriately takes advantage of this observation by heavily weighting common documents.

In our own work, we have found that CombMNZ is the most effective of these schemes, and as it has become something of a standard, we shall use it as a baseline for experimental comparison.

2.3 Averaging Models

In the context of the filtering problem, Hull et al. [12] try averaging. The four systems they fuse each output estimates for the probability of relevance of each document, so they try both directly averaging these probabilities as well as averaging the log-odds ratios, $\log \frac{p}{1-p}$, a technique related to our own log-odds formulation. They find that averaging the log-odds does yield improvement and in fact works better than more complicated regression and weighting techniques.

Manmatha et al. [14] also average probabilities of relevance, pointing out that this minimizes the Bayes' error if the search engines are viewed as independent classifiers. Without training data they transform arbitrary relevance scores into probabilities by fitting a negative exponential and Gaussian mixture model to the scores, from which they infer probabilities of relevance.

2.4 Logistic Regression Model

Savoy et al. [21] train a logistic regression model and find that over one TREC collection (WSJ2), it achieves performance slightly superior to a linear combination fusion model, improving on the best input system's performance by almost 11%.

2.5 Linear Combination Model

Bartell [2], Vogt [31, 30, 28, 29], and others experiment with linearly combining the normalized relevance scores given to each document. That is,

$$rel(d) = \sum_i \alpha_i rel_i(d)$$

where the sum is over the constituent systems. Note that this fusion model requires both relevance scores and training data to determine the weight α_i given to each input system.

Although good results are achieved in specific cases, this technique has not yet been shown to produce reliable improvement.

3. BORDA-FUSE

In this section, we describe our first model for the metasearch problem which is based on election strategies. We begin by describing the corresponding problem of voting.

3.1 A Voting Model

Voting procedures can be considered data fusion algorithms since they combine the preferences of multiple "experts". Many voting procedures, including plurality voting, instant run-off, and approval voting are not directly applicable to the problem of metasearch: they assume a few-candidates, many-voters scenario. In metasearch we have the opposite: many candidates and relatively few voters. The Borda Count voting algorithm, however, is applicable even when the number of voters is small. Interestingly, it has recently been shown [20, 7] that the Borda Count is optimal in the sense that only the Borda Count satisfies all of the symmetry properties that one would expect of any reasonable election strategy.

The Borda Count works as follows. Each voter ranks a fixed set of c candidates in order of preference. For each voter, the top ranked candidate is given c points, the second ranked candidate is given $c-1$ points, and so on. If there are some candidates left unranked by the voter, the remaining points are divided evenly among the unranked candidates. The candidates are ranked in order of total points, and the candidate with the most points wins the election.

There is a direct analogy between multi-candidate election strategies and the problem of metasearch: the set of retrieved documents are the "candidates", and the input retrieval systems are "voters" expressing preferential rankings among these candidates. Applied in this fashion, the Borda Count can be used to combine the ranked lists returned by multiple retrieval systems. No relevance scores are required, no training data is required, and the combination algorithm is simple and efficient. Furthermore, our experiments detailed below demonstrate that the performance of the Borda Count is quite good.

Researchers outside the field of Information Retrieval have also used the Borda Count to combine ranked lists. Van Erp and Schomaker [27] experiment with the Borda Count and two variants to combine simulated classifier ranked data for the field of Handwriting Recognition.

3.1.1 Weighted Borda-fuse

In the case of a democratic election, it is certainly desirable that each voter has equal weight. In metasearch, however, the best performance is often achieved by unequally weighting the input systems. A simple weighting scheme is to multiply the points assigned to a candidate by system S_i by a system weight α_i. Reasonable weights might include an overall assessment of the performance of the system such as its average precision, and we experiment with just such weights. We know that this need not be the optimal weighting scheme, and in fact Bartell and Vogt report results to the contrary for their linear combination experiments. We shall call this strategy *Weighted Borda-fuse* and leave more sophisticated weighting techniques for future work. Obtaining performance assessments for an underlying system is a form of weak training for our metasearch algorithm. In order to test fairly, in all of our experiments (described below) we train on even TREC topics and test on the odd topics and then train on odd topics and test on evens, averaging the two results.

We now report on some preliminary experiments using Borda-fuse and Weighted Borda-fuse.

3.2 Experiments

3.2.1 Data Sets

We use systems submitted to the annual Text REtrieval Conference (TREC) as input to our fusion algorithms. TREC offers large, standard data sets with many ranked lists for each query, ready to be fused. Also, each system submits 50 queries, enough to allow training and testing. Table 1 shows information about the data sets. Note that in TREC, each system is allowed to return up to 1000 documents for each query. For the TREC 3 and TREC 5 data, we used the data submitted to the TREC "adhoc" task. For TREC 9, the adhoc task had been replaced by the "web"

Data Set Name	TREC Topics	No. Sys	Avg Precision Min	Max	Avg	St Dev
TREC 3	151–200	40	0.029	0.423	0.257	0.0859
TREC 5	251–300	61	0.004	0.317	0.190	0.0683
Vogt	(10 topics)	10	0.225	0.395	0.288	0.0543
TREC 9	451–500	105	0.000	0.352	0.144	0.0779

Table 1: The data sets used in our experiments.

track. Therefore, over that data set we are fusing the results of World Wide Web search engines. In TREC terminology, a "topic" is a query; they are numbered consecutively. In the table, the column labeled "No. Sys" contains the number of search systems that submitted results to TREC that year—this is the number of systems available for us to fuse.

The Vogt data set consists of a subset of the TREC 5 data set as defined by Vogt [29]. In particular, it contains only 10 of the 61 TREC 5 systems, and only 10 of the 50 TREC 5 queries. This subset was chosen by Vogt to highlight the strengths of the metasearch technique: it contains retrieval systems chosen to maximize diversity, as measured by nine similarity criteria. The systems are: CLTHES, DCU961, anu5aut1, anu5man6, brkly17, colm1, fsclt4, gm96ma1, mds002, and uwgcx0. The queries were chosen for their large number of relevant documents: queries 257, 259, 261, 272, 274, 280, 282, 285, 288, and 289. We include this data set because of its diverse inputs: we expect that it more closely models the environment of "internal" metasearch than the other data sets.

The last four columns in the table show performance information about each data set: the minimum average precision obtained by a system in a given data set, the maximum, the mean, and the standard deviation. Note that the Vogt data set and the TREC 9 data set are very different: not only is Vogt small while TREC 9 is large, but Vogt has the highest average performance, while TREC 9 has the lowest (only half of Vogt's 0.288). And, Vogt has low deviation, especially given its high performance, while TREC 9 has high deviation, especially given its low performance. Thus the performances of systems in Vogt are similar to each other (a situation advantageous for metasearch) while the performances of systems in TREC 9 vary widely.

3.2.2 *Experimental Setup*

We examine the performance of each metasearch strategy when combining random groups of retrieval systems. Each data point represents the average value obtained over 200 trials (or as many as are combinatorially possible) performed as follows. Randomly select a set of n (for $n \in \{2, 4, ...12\}$) input systems, apply the metasearch algorithm to these systems, and record the average precision of the metasearch algorithm's output. (Additionally, we record the average precision of the best underlying system in order to meaningfully assess the improvement gained, if any.) This experiment is designed to test how well a fusion algorithm performs on average, and to see how the algorithm improves when more input systems are available. A successful system will consistently improve on the *best* of its inputs, no matter how many input systems are available.

3.2.3 *Experimental Results*

In the experiments, we compare the performance of Borda-fuse and Weighted Borda-fuse with CombMNZ. Note that each of the these algorithms requires slightly different information (see Figure 1): CombMNZ needs relevance scores, Weighted Borda-fuse needs training data (albeit of a very minimal kind), and Borda-fuse needs neither. The experimental results are shown in Figure 2.

From the graphs, first note that for all of the plots, performance usually increases as more systems are added to the combination. Even when 10 or 12 systems are being combined, performance gains are usually still available.

Also note that on the Vogt data set, all of the fusion algorithms yield considerable performance increases, while the TREC 9 data set is much more difficult. We expect this is due to the already-noted differences in the data sets: Vogt contains systems that rank documents very differently yet perform similarly. Thus each input system provides new information, but also reliable information (relative to the other input systems). TREC 9 contains systems that perform very differently—some systems provide relatively unreliable information. We have not studied how similarly TREC 9 systems rank the documents.

Considering the curves individually, we see that for three out of the four data sets, Weighted Borda-fuse outperforms the best input system. Also, Weighted Borda-fuse is always at least as good as the unweighted Borda-fuse, and sometimes (see TREC 9) dramatically better. This is probably due to the large differences in performance between input systems in TREC 9: weights allow the algorithm to concentrate on the good advice it receives and somewhat ignore the bad advice of poor performers. Finally, note that Weighted Borda-fuse has performance comparable to CombMNZ on each data set.

4. BAYES-FUSE

In this section, we describe our second model for the metasearch problem which is based on Bayesian inference.

4.1 The Probabilistic Model

Given the ranked lists of documents returned by n retrieval systems, let $r_i(d)$ be the *rank* assigned to document d by retrieval system i (a rank of ∞ may be used if document d is not retrieved by system i). This constitutes the *evidence of relevance* provided to the metasearch strategy concerning document d. For a given document, let

$$\begin{aligned} P_{\rm rel} &= \Pr[\mathrm{rel}|r_1, r_2, \ldots, r_n] \quad \text{and} \\ P_{\rm irr} &= \Pr[\mathrm{irr}|r_1, r_2, \ldots, r_n] \end{aligned}$$

be the respective probabilities that the given document is *relevant* and *irrelevant* given the rank evidence $r_1, r_2, \ldots, r_n$. The Bayes optimal decision rule for determining the relevance of a document dictates that a document should be assumed relevant if and only if $P_{\rm rel}/P_{\rm irr} \geq \tau$ for some threshold τ chosen so as to optimize the expected loss suffered if incorrect. Since we are interested in *ranking* the documents, we shall compute this *odds* of relevance

$$O_{\rm rel} = P_{\rm rel}/P_{\rm irr}$$

and rank documents according to this measure. Applying

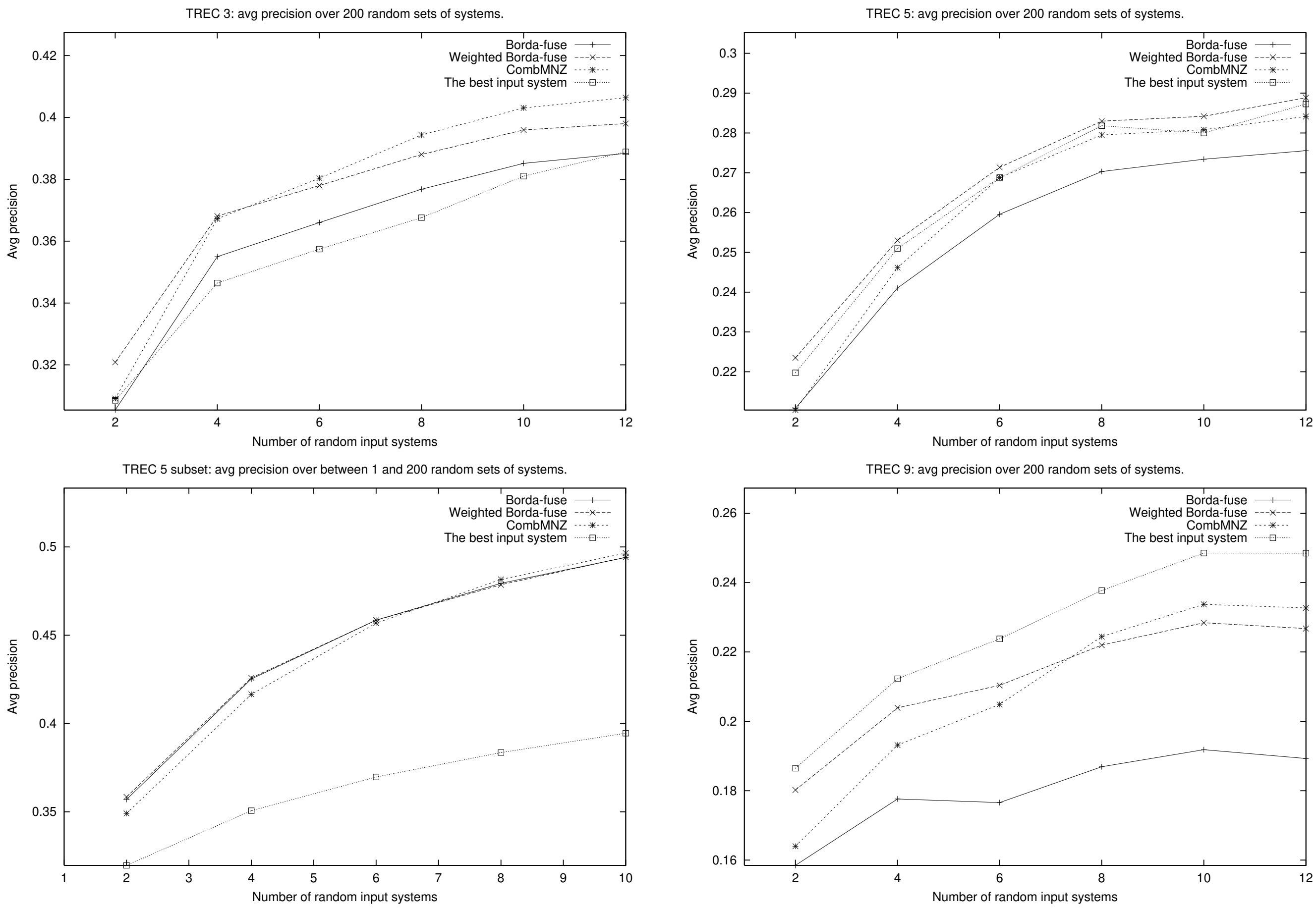

Figure 2: Borda-fuse experiments over the four data sets. Without using relevance scores, Weighted Borda-fuse performs comparably to CombMNZ.

Bayes rule, we obtain

$$P_{\mathrm{rel}} = \frac{\Pr[r_1, r_2, \ldots, r_n|\mathrm{rel}] \cdot \Pr[\mathrm{rel}]}{\Pr[r_1, r_2, \ldots, r_n]} \quad \text{and}$$

$$P_{\mathrm{irr}} = \frac{\Pr[r_1, r_2, \ldots, r_n|\mathrm{irr}] \cdot \Pr[\mathrm{irr}]}{\Pr[r_1, r_2, \ldots, r_n]}.$$

While the $\Pr[r_1, r_2, \ldots, r_n]$ term would be difficult to assess in practice, it is eliminated in the odds formulation

$$O_{\mathrm{rel}} = \frac{\Pr[r_1, r_2, \ldots, r_n|\mathrm{rel}] \cdot \Pr[\mathrm{rel}]}{\Pr[r_1, r_2, \ldots, r_n|\mathrm{irr}] \cdot \Pr[\mathrm{irr}]}. \quad (1)$$

By making the common *naive Bayes* independence assumptions[2] and taking logs, we arrive at a simple log odds formulation:

$$O_{\mathrm{rel}} = \frac{\prod_i \Pr[r_i|\mathrm{rel}] \cdot \Pr[\mathrm{rel}]}{\prod_i \Pr[r_i|\mathrm{irr}] \cdot \Pr[\mathrm{irr}]},$$

$$\log O_{\mathrm{rel}} = \sum_i \log \frac{\Pr[r_i|\mathrm{rel}]}{\Pr[r_i|\mathrm{irr}]} + \log \frac{\Pr[\mathrm{rel}]}{\Pr[\mathrm{irr}]}.$$

Finally, since we are solely concerned with *ranking* the documents, we may drop the common $\log(\Pr[\mathrm{rel}]/\Pr[\mathrm{irr}])$ term, obtaining our relevance formula

$$rel(d) = \sum_i \log \frac{\Pr[r_i(d)|\mathrm{rel}]}{\Pr[r_i(d)|\mathrm{irr}]}. \quad (2)$$

Note that $\Pr[r_i|\mathrm{rel}]$ is the probability that a relevant document would be ranked at level r_i by system i. Similarly, $\Pr[r_i|\mathrm{irr}]$ is the probability that an irrelevant document would be ranked at level r_i by system i. Thus, to obtain the relevance of a document for ranking purposes, we simply sum the log of the ratio of these probabilities over all systems.

We note that this derivation is similar to other probabilistic models that have been proposed for different problems in IR (e.g., [4, 5]). Indeed, the combination of evidence via Bayesian techniques can be applied at the level of system combination (as in our case) as well as at the level of individual system construction (as in the combination of evidence from various features, etc.).

4.2 Implementation

A number of details are left unspecified in this model; in particular, how should the estimation of $Pr[r_i(d)|rel]$ be computed?

[2]While these independence assumptions may not be true (for example, we are assuming that the ranks assigned to a given relevant document by two systems are independent), they are often made in practice.

```
Queryid (Num):          50
Total number of documents over all queries
    Retrieved:      50000
    Relevant:        9805
    Rel_ret:         7305
Interpolated Recall - Precision Averages:
    at 0.00        0.8992
    at 0.10        0.7514
    at 0.20        0.6584
    at 0.30        0.5724
    at 0.40        0.4982
    at 0.50        0.4272
    at 0.60        0.3521
    at 0.70        0.2915
    at 0.80        0.2173
    at 0.90        0.1336
    at 1.00        0.0115
Average precision (non-interpolated)
for all rel docs(averaged over queries)
                   0.4226
Precision:
  At    5 docs:   0.7440
  At   10 docs:   0.7220
  At   15 docs:   0.6867
  At   20 docs:   0.6740
  At   30 docs:   0.6267
  At  100 docs:   0.4902
  At  200 docs:   0.3848
  At  500 docs:   0.2401
  At 1000 docs:   0.1461
R-Precision (precision after R
(= num_rel for a query) docs retrieved):
    Exact:         0.4524
```

Figure 3: `trec_eval` statistics for `inq102`.

We use the results of the `trec_eval` program (available from the TREC website) to obtain the data necessary to infer the probabilities used in Equation 2. An example of the output of this program for the best system in the TREC 3 competition, `inq102`, is given in Figure 3. In particular, we use the average precisions at the various document levels together with the average number of relevant documents per query to estimate the probability that a relevant document would be ranked, for example, in the range $\{31, 100\}$. From this we obtain $\Pr[31 \leq r_i(d) \leq 100|\text{rel}]$. We repeated this procedure for each rank bucket $\{\{1, 5\}, \{6, 10\}, \ldots, \{1001, R\}\}$, where R is the total number of relevant documents. Repeating for irrelevant documents gives us all of the basic probabilities we need for Equation 2. We have experimented with more sophisticated methods of estimating these basic probabilities, including smoothing and interpolating, but obtained no performance gains over this simple method.

The relevance judgments used by the `trec_eval` program were made by human assessors. This technique could also be used to assess the performance of real-world systems; automatic techniques could also be employed.

Gathering statistics about a retrieval system is a type of training for our combination strategy. Therefore, as for Weighted Borda-fuse, we train on even topics and test on odd, then train on odd topics and test on even, and average the two results.

4.3 Experiments

In this section, we describe experimental results comparing the performance of CombMNZ and Bayes. Results are shown in Figure 4.

Perhaps most impressive is the performance of Bayes-fuse on the Vogt data set. Here, over a diverse pool of inputs, the naive Bayes independence assumptions are more valid, and the performance of Bayes-fuse exceeds that of CombMNZ (and both Borda-fuse algorithms—compare with Figure 2).

On all of the data sets except TREC 9, Bayes-fuse outperforms the best input system. On half of the data sets, Bayes-fuse outperforms CombMNZ.

5. UPPER BOUNDS

We have seen performance curves for different metasearch algorithms, in some cases showing substantial improvements over the best input system. But questions remain: how much improvement can be gained by metasearch? Are existing techniques approaching a natural limit? To address these questions, we must find upper bounds on the performance of any metasearch algorithm.

5.1 Models

Assume we have an omniscient oracle that is going to act as a metasearch algorithm. Like other metasearch algorithms, the oracle takes ranked lists as input and must produce a single ranked list as output. Unlike other metasearch algorithms, the oracle knows which documents are relevant. Without any constraints, the oracle's job is easy: for each fusion problem, the oracle simply ignores its inputs and then outputs the relevant documents for the given query (ranked arbitrarily)—it scores perfectly every time. In order to obtain a meaningful upper bound on the performance of any real metasearch strategy, we *constrain* the oracle in any way which is consistent with a natural constraint imposed on any real metasearch strategy. We discuss two such constraints below.

5.1.1 Naive Bound

The first natural constraint is that the oracle may *only output documents that appear in one of the input ranked lists*. Although the oracle can return all of the relevant documents in the input and none of the irrelevant, it no longer scores perfectly since it is not allowed to include *all* relevant documents for each query; it is dependent on its input. We call the performance of this oracle the *Naive Bound*.

5.1.2 Ordered Pairs Bound

We can further constrain the oracle in a natural way by insisting that it effectively heed the advice of its inputs. In particular, *if document A is ranked above document B by all of the input systems, then A must be ranked above B in the output.*[3] We call the performance of this oracle the *Ordered-Pairs Bound*.

5.2 Implementations

Implementing the oracle for the Naive Bound is straightforward. However, implementing the oracle for the Ordered Pairs Bound is more difficult, due to the computational complexity of finding all consistently ranked pairs and optimally ranking the documents subject to these constraints. Thus we have implemented a somewhat weaker constraint as follows: All *relevant* documents are assigned a score equal to the *minimum rank* given to it by an input system; all *irrelevant* documents are scored according to their *maximum rank*. The documents are then sorted in order of increasing score with ties broken in favor of relevant documents.

[3]Of course, not all fusion algorithms need obey this rule, and for some input systems it is not even a good idea. For instance, a perverse input system might always invert it's ranked list before giving to the fusion routine. In this case, a smart fusion routine could notice the inversion and correct for it, thereby even profiting from the perverse input system and perhaps outperforming the constrained oracle. Even so, *reasonable* fusion algorithms acting on *reasonable* input systems will agree with the unanimous advice from their inputs.

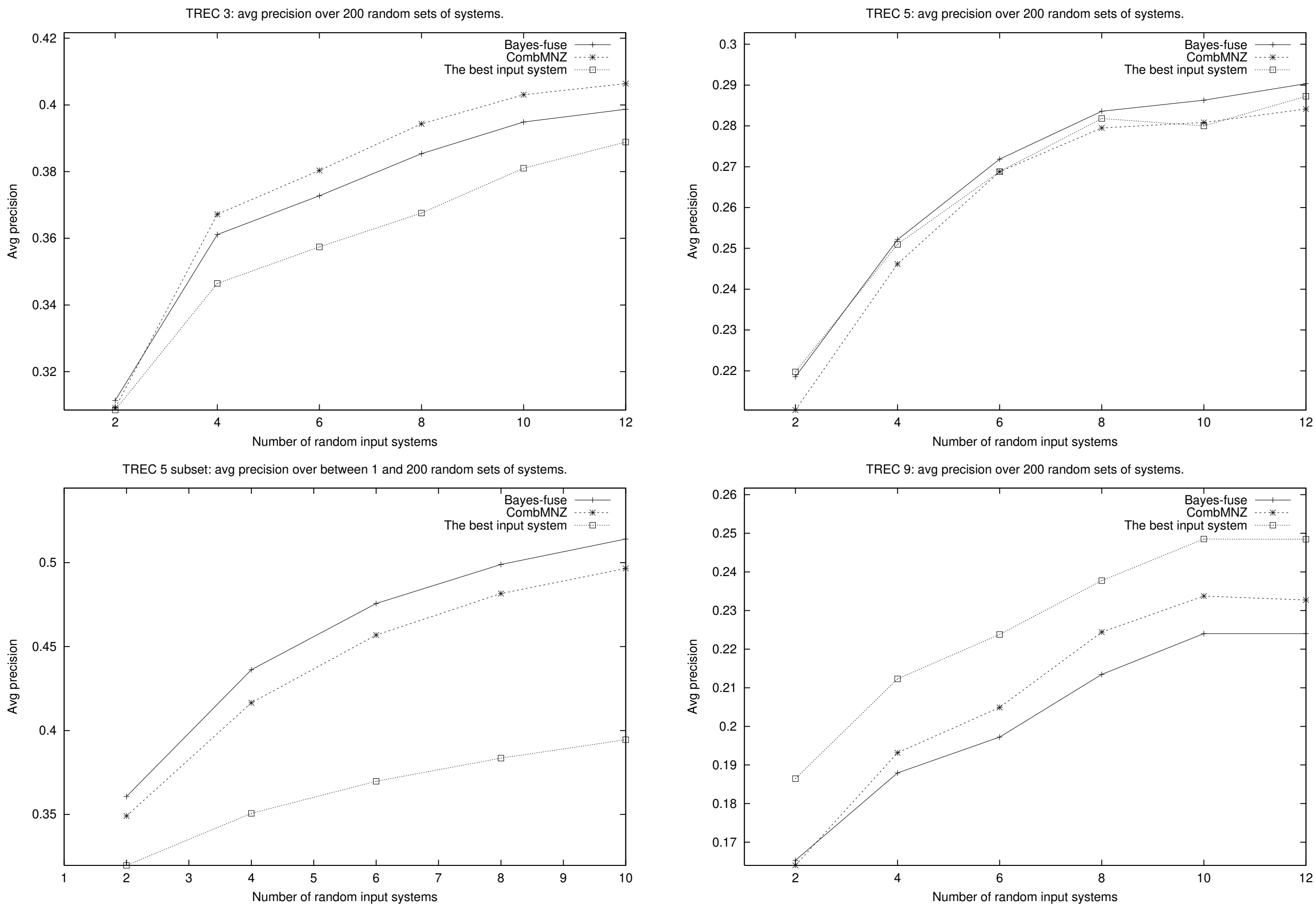

Figure 4: Bayes-fuse experiments over the four data sets. Using training data but no relevance scores, Bayes-fuse outperforms CombMNZ on half of the data sets. It performs especially well on the Vogt subset of TREC 5, where the diversity of the input systems suits the naive Bayes independence assumptions.

Note that this constraint captures many, though not all, of the ordered pair constraints between relevant and irrelevant documents. We call the performance of this oracle the *Min/Max Bound*.

5.3 Experiments

Results of the upper bounds experiments are shown in Figure 5. Note that in this case each data point represents only 50 trials.

When combining two systems, the Naive Bound achieves 75% of the performance of an unconstrained oracle (not shown), and the Min/Max Bound achieves half the performance; actual metasearch systems are closer to 25%. Furthermore, the upper bounds approach perfect scores relatively quickly as the number of systems being fused increases. Qualitatively similar results are obtained on other data sets.

6. CONCLUSIONS

We may summarize our work with Borda-fuse as follows: (1) The algorithm is *simple*; (2) it requires only *ranks*, not relevance scores; (3) it requires *no training*; (4) if even the simplest form of training data is available, its performance usually *exceeds* that of the best input system; and (5) with minimal training, its performance is competitive with that of standard techniques like CombMNZ which require relevance scores. The performance of Weighted Borda-fuse will almost certainly improve through the use of optimized weights.

We may summarize our work with Bayes-fuse as follows: (1) The probabilistic model is *simple* and *rigorous*; (2) Bayes-fuse requires only *ranks*, not relevance scores; (3) it requires *training*, but only a particularly simple kind (e.g., just the gathering of simple statistics about the number of relevant documents above different ranks); (4) its performance usually *exceeds* that of the best input system; (5) its performance is *robust* even when many poor systems are included in the fusion; (6) its performance is comparable to that of standard techniques such as CombMNZ; and (7) its performance is especially strong over diverse inputs. One may expect to find this kind of diversity when using metasearch to combine subsystems within a search engine.

One way in which Bayes-fuse may be improved is in the (full or partial) elimination of the independence assumptions used to transition from Equation 1 to Equation 2. These independence assumptions, while commonly made in practice, are almost certainly untrue. A more sophisticated evaluation of Equation 1 which accounts for this dependence may yield improvements in our strategy.

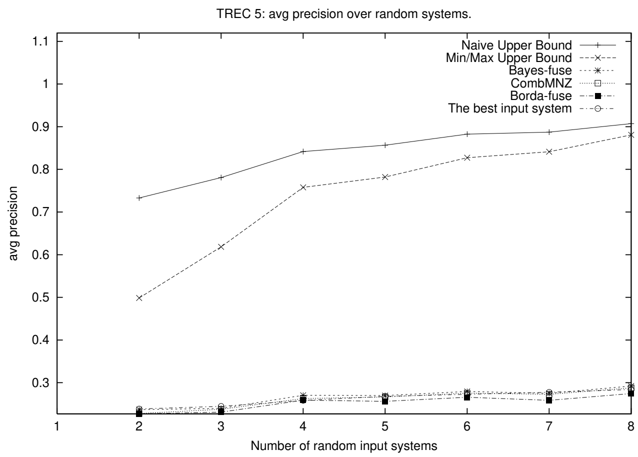

Figure 5: Upper bounds experiments over the TREC 5 data set.

Finally, we have proposed the first models for assessing the limits of the performance of metasearch. The performance gap between existing metasearch algorithms and our tightest upper bound suggests that there is much to learn about the limits of the performance of metasearch and/or the existence of better metasearch algorithms.

7. REFERENCES

[1] *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, Louisiana, USA, 2001. ACM Press, New York.

[2] B. T. Bartell. *Optimizing Ranking Functions: A Connectionist Approach to Adaptive Information Retrieval*. PhD thesis, University of California, San Diego, 1994.

[3] N. Belkin, P. Kantor, C. Cool, and R. Quatrain. Combining evidence for information retrieval. In Harman [11], pages 35–43.

[4] W. S. Cooper. Some inconsistencies and misnomers in probabilistic information retrieval. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, New York, 1991.

[5] W. S. Cooper, A. Chen, and F. C. Gey. Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. In Harman [11], pages 57–66.

[6] W. B. Croft. Combining approaches to information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, chapter 1. Kluwer Academic Publishers, 2000.

[7] The mathematics of voting: Democratic symmetry. *The Economist*, page 83, Mar. 2000.

[8] E. A. Fox, M. P. Koushik, J. Shaw, , R. Modlin, and D. Rao. Combining evidence from multiple searches. In D. Harman, editor, *The First Text REtrieval Conference (TREC-1)*, pages 319–328, Gaithersburg, MD, USA, Mar. 1993. U.S. Government Printing Office, Washington D.C.

[9] E. A. Fox and J. A. Shaw. Combination of multiple searches. In Harman [11], pages 243–249.

[10] K. L. Fox, O. Frieder, M. Knepper, and E. Snowberg. SENTINEL: A multiple engine information retrieval and visualization system. *Journal of the ASIS*, 50(7), May 1999.

[11] D. Harman, editor. *The Second Text REtrieval Conference (TREC-2)*, Gaithersburg, MD, USA, Mar. 1994. U.S. Government Printing Office, Washington D.C.

[12] D. A. Hull, J. O. Pedersen, and H. Schütze. Method combination for document filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 279–287, Zurich, Switzerland, 1996. ACM Press, New York.

[13] J. H. Lee. Analyses of multiple evidence combination. In N. J. Belkin, A. D. Narasimhalu, and P. Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–275, Philadelphia, Pennsylvania, USA, July 1997. ACM Press, New York.

[14] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1].

[15] M. Montague and J. Aslam. Metasearch consistency. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1].

[16] K. B. Ng. *An Investigation of the Conditions for Effective Data Fusion in Information Retrieval*. PhD thesis, School of Communication, Information, and Library Studies, Rutgers University, 1998.

[17] K. B. Ng and P. B. Kantor. An investigation of the preconditions for effective data fusion in ir: A pilot study. In *Proceedings of the 61th Annual Meeting of the American Society for Information Science*, 1998.

[18] K. B. Ng, D. Loewenstern, C. Basu, H. Hirsh, and P. B. Kantor. Data fusion of machine-learning methods for the TREC5 routing task (and other work). In Voorhees and Harman [32], pages 477–487.

[19] *Content-Based Multimedia Information Access (RIAO)*, Paris, France, Apr. 2000.

[20] D. G. Saari. Explaining all three-alternative voting outcomes. *Journal of Economic Theory*, 87(2):313–355, Aug. 1999.

[21] J. Savoy, A. L. Calvé, and D. Vrajitoru. Report on the TREC-5 experiment: Data fusion and collection fusion. In Voorhees and Harman [32], pages 489–502.

[22] E. Selberg and O. Etzioni. On the instability of web search engines. In RIAO [19], pages 223–235.

[23] E. W. Selberg. *Towards Comprehensive Web Search*. PhD thesis, University of Washington, 1999.

[24] J. A. Shaw and E. A. Fox. Combination of multiple searches. In D. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 105–108, Gaithersburg, MD, USA, Apr. 1995. U.S. Government Printing Office, Washington D.C.

[25] P. Thompson. A combination of expert opinion approach to probabilistic information retrieval, part 1: the conceptual model. *Information Processing and Management*, 26(3):371–382, 1990.

[26] P. Thompson. A combination of expert opinion approach to probabilistic information retrieval, part 2: mathematical treatment of CEO model 3. *Information Processing and Management*, 26(3):383–394, 1990.

[27] M. van Erp and L. Schomaker. Variants of the borda count method for combining ranked classifier hypotheses. In *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 443–452, Amsterdam, Sept. 2000. International Unipen Foundation.

[28] C. C. Vogt. *Adaptive Combination of Evidence for Information Retrieval*. PhD thesis, University of California, San Diego, 1999.

[29] C. C. Vogt. How much more is better? Characterizing the effects of adding more IR systems to a combination. In RIAO [19], pages 457–475.

[30] C. C. Vogt and G. W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, Oct. 1999.

[31] C. C. Vogt, G. W. Cottrell, R. K.Belew, and B. T. Bartell. Using relevance to train a linear mixture of experts. In Voorhees and Harman [32], pages 503–515.

[32] E. Voorhees and D. Harman, editors. *The Fifth Text REtrieval Conference (TREC-5)*, Gaithersburg, MD, USA, 1997. U.S. Government Printing Office, Washington D.C.

[33] E. Voorhees and D. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In D. Harman, editor, *The Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, USA, 2000. U.S. Government Printing Office, Washington D.C.