Bayes Optimal Metasearch: A Probabilistic Model for Combining the Results of Multiple Retrieval Systems*

Javed A. Aslam Mark Montague
Department of Computer Science
Dartmouth College
6211 Sudikoff Laboratory
Hanover, NH 03755

{jaa, montague}@cs.dartmouth.edu

Abstract

We introduce a new, probabilistic model for combining the outputs of an arbitrary number of query retrieval systems. By gathering simple statistics on the average performance of a given set of query retrieval systems, we construct a Bayes optimal mechanism for combining the outputs of these systems. Our construction yields a metasearch strategy whose empirical performance nearly always *exceeds* the performance of any of the constituent systems. Our construction is also *robust* in the sense that if "good" and "bad" systems are combined, the performance of the composite is still on par with, or exceeds, that of the best constituent system. Finally, our model and theory provide theoretical and empirical avenues for the improvement of this metasearch strategy.

1 Introduction

Numerous query retrieval systems have been developed both in academia [4] and in industry (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*. As such, *metasearch engines* (such as Dogpile, ProFusion, SavvySearch, etc.) have been introduced which query a number of search engines, merge the lists of pages returned, and present a resulting ranked list to the user. Our work concerns itself with precisely how to best merge the lists of ranked documents returned by different sub-engines, sometimes called the problem of "data fusion."

Fox and Shaw [1] experimented with a group of ranked-list combination rules in the TREC competitions. Their intuitively motivated rules are based on the unweighted min, max, or sum of each document's similarity estimates over the constituent systems.

Lee [2] 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 documents." He further argues that Fox's best combination rule, known as CombMNZ, appropriately takes advantage of this feature of variant systems' ranked lists.

Vogt, et al. [3] develop a system that learns weights to linearly combine ranked lists based on a number of properties of the lists. A document's final score (by which it is ranked) is a weighted sum of its scores in the constituent systems. Within the framework of linear combinations, they provide a formula for optimally combining two systems.

We propose a new, probabilistic model for combining the ranked lists of documents obtained by any number of query retrieval systems in response to a given query. Unlike most existing combination strategies, ours makes use of some knowledge of the average performance of the constituent systems. Our strategy most often yields a combination system whose performance *exceeds* that of any of its constituent systems. Furthermore, our strategy is *robust* in the sense that the resulting system's performance does not appreciably degrade even when some constituent systems are quite poor. Finally, our model and theory provide avenues for the possible improvement of our proposed system.

2 A Probabilistic Model for Metasearch

We assume that our metasearch system has access to the ranked lists of documents produced by a given set of retrieval systems in response to a given query. (Unlike many metasearch strategies, our system does *not* require access to the actual similarities used to produce these ranked lists, if available.) We also assume access to some simple statistics about the average performance of the constituent systems. Given this information, we develop our probabilistic model and derive a Bayes optimal strategy for metasearch as follows.

*This work generously supported by NSF grants EIA-9802068 and BCS-9978116.

Given the ranked lists of documents returned by a set of 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

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

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 $P_{\text{rel}} > P_{\text{irr}}$ and irrelevant otherwise. Since we are interested in *ranking* the documents, we shall compute the *odds* of relevance

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

and rank documents according to this measure. Applying Bayes rule, we obtain

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

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

$$O_{\text{rel}} = \frac{\Pr[r_1, r_2, \ldots, r_n|\text{rel}] \cdot \Pr[\text{rel}]}{\Pr[r_1, r_2, \ldots, r_n|\text{irr}] \cdot \Pr[\text{irr}]}. \tag{1}$$

By making the common *naive Bayes* independence assumptions,[1] we may rewrite this formula in a particularly simple form

$$O_{\text{rel}} = \frac{\Pr[\text{rel}] \cdot \prod_i \Pr[r_i|\text{rel}]}{\Pr[\text{irr}] \cdot \prod_i \Pr[r_i|\text{irr}]}. \tag{2}$$

Finally, since we are solely concerned with *ranking* the documents, we may drop the common $\Pr[\text{rel}]/\Pr[\text{irr}]$ term and take logs, obtaining our relevance formula

$$\sum_i \log \frac{\Pr[r_i|\text{rel}]}{\Pr[r_i|\text{irr}]}. \tag{3}$$

Note that $\Pr[r_i|\text{rel}]$ is the probability that a relevant document would be ranked at level r_i by system i. Similarly, $\Pr[r_i|\text{irr}]$ 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.

[1]While these independence assumptions are almost certainly not 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. A more sophisticated approach to evaluating Equation 1 will almost certainly yield better performance; we discuss such possibilities in Section 4.

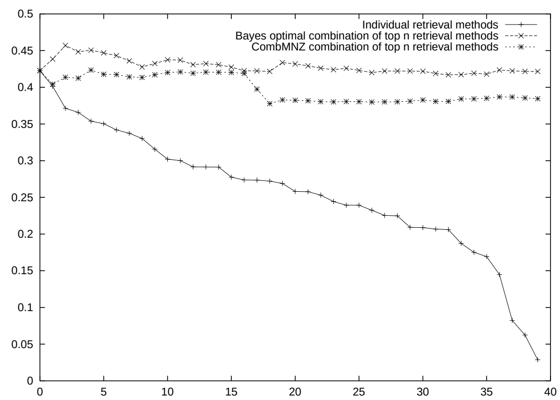

Figure 1: The 40 systems were first sorted according to performance. The x axis corresponds to system performance rank, and the y axis corresponds to 11 point average precision. In one plot, we show the performance of each of the 40 systems entered in TREC3. In another plot, we show the performance of our metasearch strategy in combining the top i ranked systems. In a third plot, we show the performance of the CombMNZ strategy in combining the top i ranked systems.

3 Experiments

We evaluated our system using data from the adhoc track of the TREC3 competition. Forty systems were entered in that year's competition, each submitting runs corresponding to 50 queries (TREC topics 151 to 200). The 11 point average precisions for each of these systems is given in Figure 1 (together with other results), sorted according to performance. We used the results of the supplied `trec_eval` program to obtain the data necessary to infer the probabilities used in Equation 3. An example of the output of this program for the best system in that year's competition, `inq102`, is given in Figure 2. In particular, we used the average precisions at various document levels together with available information on average numbers of relevant and irrelevant documents to estimate the probabilities that relevant and irrelevant documents would be ranked at any level i, as required in Equation 3. We note that the relevance judgements 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 might also be employed. Details are provided in the full paper.

Note that gathering statistics about a retrieval system can be viewed as a type of "training" for our

```
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 2: `trec_eval` statistics for `inq102`.

combination strategy. Therefore, in all our experiments, we first gathered all necessary statistics using the *odd* TREC topics and tested our system using the *even* topics. We then gathered all necessary statistics using the *even* TREC topics and tested our system using the *odd* topics. The performance recorded in all cases was the average of these two runs.[2]

In our first set of experiments, we used our proposed strategy to combine the best i retrieval systems, for all $i \in \{1, \ldots, 40\}$. In other words, we used our proposed strategy to first (trivially) "combine" the top system, then the top two systems, the top three systems, and so on, culminating in the final experiment in which we combined all 40 systems. Our results are shown in Figure 1. Note that the performance of our strategy always equalled or exceeded the performance of the *best* constituent system, even when extremely poor systems were part of the combination. The performance of our strategy peaked when combining the top three systems, at which point it yielded an 8.1% improvement over the best system and a 14.7% improvement over the average performance of the three systems it was combining. Note that the performance of the best system was quite good; in other experiments, we have seen far greater absolute and percentage gains when combining systems of more moderate performance. Finally, for the purposes of comparison, we ran identical combination experiments using the well studied CombMNZ strategy [1, 2]. The results of these experiments are also shown in Figure 1; in all cases, the performance of our strategy exceeded that of CombMNZ.[3]

We conducted another experiment to determine the "expected" improvement that combining two systems would generate. We generated 100 *pairs* of systems at random from among the 40 systems which participated in TREC3, and we combined each of these pairs using both our strategy and CombMNZ. For each combination, the percentage improvement over the *better* of the two constituent systems was calculated, and these percentage improvements were averaged over all 100 runs. Our results were as follows. CombMNZ yielded and average performance 12.8% *lower* than the better of its constituent systems, while our strategy yielded an average performance 1.5% *higher* than the better of its constituent systems.[4]

4 Discussion

We have proposed a probabilistic model for combining the outputs of an arbitrary number of query retrieval systems. Within this model, we have derived a Bayes optimal strategy for performing such combinations. Finally, using TREC data, we have demonstrated that our strategy is both powerful and robust.

One way in which our strategy may very likely 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. In fact, Lee [2] argues convincingly that the sets of relevant documents returned by retrieval systems are highly correlated, while the sets of irrelevant documents returned are far less so. A more sophisticated evaluation of Equation 1 which accounts for this dependence will almost certainly yield improvements in our strategy, and we are currently pursuing just such an improvement.

References

[1] E. A. Fox and J. A. Shaw. Combination of multiple searches. In *The Second Text REtrieval Conference (TREC-2)*, pages 243–249, 1994.

[2] J. H. Lee. Analyses of multiple evidence combination. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–275, 1997.

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

[4] E. Voorhees and D. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In *The Eighth Text REtrieval Conference (TREC-8)*, 2000.

[2] This is essentially two-way hold-out cross-validation.

[3] This is perhaps unsurprising given that our strategy makes use of statistics concerning the average performance of its constituent systems while CombMNZ does not. Note, however, that CombMNZ generally requires the actual *similarity scores* used to produce rankings, while our strategy needs only the rankings (and some performance statistics).

[4] Again, this is perhaps unsurprising given that our strategy "knows" something about the constituent systems; it does, however, emphasize the power that such knowledge can bring.