A Unified Model for Metasearch, Pooling, and System Evaluation*

Javed A. Aslam†, Virgiliu Pavlu, Robert Savell
Department of Computer Science
Dartmouth College
6211 Sudikoff Laboratory
Hanover, NH 03755
{jaa,virgilpavlu,rsavell}@cs.dartmouth.edu

ABSTRACT

We present a unified model which, given the ranked lists of documents returned by multiple retrieval systems in response to a given query, simultaneously solves the problems of (1) fusing the ranked lists of documents in order to obtain a high-quality combined list (metasearch); (2) generating document collections likely to contain large fractions of relevant documents (pooling); and (3) accurately evaluating the underlying retrieval systems with small numbers of relevance judgments (efficient system assessment). Our approach is based on the Hedge algorithm for on-line learning. In effect, our proposed system "learns" which documents are likely to be relevant from a sequence of on-line relevance judgments. In experiments using TREC data, our methodology is shown to outperform standard methods for metasearch, pooling, and system evaluation, often remarkably so.

1. INTRODUCTION

We consider the problems of metasearch, pooling, and system evaluation, and we show that all three problems can be efficiently and effectively solved with a single technique based on the Hedge algorithm for on-line learning. Our results from experiments with TREC data demonstrate that: (1) As an algorithm for metasearch, our technique combines ranked lists of documents in a manner whose performance equals or exceeds that of benchmark algorithms such as CombMNZ and Condorcet, and it generalizes these algorithms by seemlessly incorporating user feedback in order to obtain dramatically improved performance. (2) As an algorithm for pooling, our technique generates sets of documents containing far more relevant documents than standard techniques such as TREC-style depth pooling. (3) These pools, when used to evaluate retrieval systems, estimate the performance of retrieval systems and rank these systems in a manner superior to TREC-style depth pools of an equivalent size.

Our unified model for solving these three problems is based on the Hedge algorithm for on-line learning. In the context of these problems, Hedge effectively learns which systems are "better" than others and which documents are "more likely relevant" than others, given on-line relevance feedback. Thus Hedge (1) learns to rank documents in order of relvance (metasearch), (2) learns how to generate document sets likely to contain large fractions of relevant documents (pooling), and (3) efficiently and effectively evaluates the underlying retreival systems using these pools.

In the sections that follow, we describe the problems of metasearch, pooling, and system evaluation in more detail and discuss our results.

1.1 Metasearch

Metasearch is the well-studied process of fusing the ranked lists of documents returned by a collection of systems in response to a given user query in order to obtain a combined list whose quality equals or exceeds that of any of the underlying lists. Many metasearch techniques have been proposed and studied [3, 9, 14, 2, 12]. In this work, we consider two benchmark techniques: the first is based on combining the normalized scores given to each document by the underlying systems (CombMNZ [6, 10]), and the second is based on viewing the metasearch problem as a multi-candidate election where the documents are candidates and the systems are voters expressing preferential rankings among the candidates (Condorcet [13]).

CombMNZ and Condorcet produce quality ranked lists of documents by fusing the ranked lists provided by a collection of underlying systems. Given ranked lists produced by good but sufficiently different underlying systems, these metasearch techniques can produce fused lists whose quality exceeds that of any of the underlying lists. Given ranked lists produced by possibly correlated systems of varying performance, these metasearch techniques will most often produce fused lists whose performance exceeds that of the "average" underlying list but which rarely exceeds that of the best underyling list.

In the context of a metasearch engine, the fused list produced by CombMNZ or Condorcet would be presented to the user who would naturally begin processing the documents in rank order to satisfy the desired information need. While the user could naturally and easily provide relevance feed-

†Author for correspondence. Phone: (603) 646-1613. Fax: (603) 646-1672.
*This work partially supported by NSF Career Grant CCR-0093131 and NSF Grant 5-36955.

back to the metasearch algorithm, these techniques are not easily or naturally amenable to incorporating such feedback.

By contrast, our technique based on the Hedge algorithm for on-line learning quite naturally incorporates relevance feedback and performs impressively even in the absence of feedback.

In the absence of feedback, the metasearch performance of our technique most often equals or exceeds that of benchmark techniques such as CombMNZ and Condorcet. In experiments using TREC data, Hedge effectively equaled the performance of CombMNZ in four out of five competitions tested (TRECs 3, 6, 7, and 8), and Hedge outperformed CombMNZ in one competition (TREC 5); Hedge consistently outperformed Condorcet in each of the competitions tested, significantly so in two competition (TRECs 6 and 7).

In the presence of relevance feedback, our technique rapidly and effectively "learns" how to fuse the underlying ranked lists, significantly outperforming CombMNZ and Condorcet, and often outperforming the best underlying system after only a handful of relevance judgments.

1.2 Pooling and System Evaluation

Collections of retrieval systems are traditionally evaluated by (1) constructing a test collection of documents (the "corpus"), (2) constructing a test collection of queries (the "topics"), (3) judging the relevance of the documents to each query (the "relevance judgments"), and (4) assessing the quality of the ranked lists of documents returned by each retrieval system for each topic using standard measures of performance such as mean average precision. Much thought and research has been devoted to each of these steps in, for example, the annual TREC competition [8].

For large collections of documents and/or topics, it is impractical to assess the relevance of each document to each topic. Instead, a small subset of the documents is chosen, and the relevance of these documents to the topics is assessed. When evaluating the performance of a collection of retrieval system, as in the annual TREC competition [8], this judged "pool" of documents is typically constructed by taking the union of the top k documents returned by each system in response to a given query; in the TREC competition, $k = 100$ has been shown to be an effective cutoff in evaluating the relative performance of retrieval systems [8]. Both shallower and deeper pools have been studied [16, 8], both for TREC and within the greater context of the generation of large test collections [4]. Pooling is an effective technique since many of the documents relevant to a topic will appear near the top of the lists returned by (quality) retrieval systems; thus, these relevant documents will be judged and used to effectively assess the performance of the collected systems.

While pooling is an effective technique for greatly reducing the number of relevance judgments required for effective system evaluation, it can still be quite expensive. Within the TREC competition, for example, upwards of 100 systems return lists of 1000 ranked documents in response to each of 50 topics. Traditional TREC-style pooling dictates that the top 100 documents returned by each system in response to each topic should be judged, and these relevance judgments should then be used to assess the relative performance of the systems. While many of the top documents are retrieved by multiple systems, thus reducing the overall size of the pool, the total number of relevance judgments is still substantial; for example, in TREC 8 [15] 86,830 relevance judgments were used to assess the quality of the retrieved lists submitted by 129 systems in response to 50 topics. Reducing the number of relevance judgments required would permit competitions such as TREC to scale well in the future, as well as more easily permit the assessment of large numbers of systems over vast, changing data collections such as the World Wide Web.

Pools are often used to evaluate retrieval systems in the following manner. The documents within a pool are judged to determine whether they are relevant or not relevant to the given user query or topic. Documents not contained within the pool are *assumed* to be non-relevant. The ranked lists returned by the retrieval systems are then evaluated using standard measures of performance (such as mean average precision) using this "complete" set of relevance judgments. Since documents not present in the pool are assumed non-relevant, the *quality* of the assessments produced by such a pool is often in direct proportion to the fraction of relevant documents found in the pool (its *recall*). On-line pooling techniques have been proposed which attempt to identify relevant documents as quickly as possible in order to exploit this phenomenon [4].

In the results that follow, we demonstrate that the Hedge algorithm for on-line learning is ideally suited to generating efficient pools which effectively evaluate retrieval systems. In effect, the Hedge algorithm learns which documents are likely to be relevant, these documents can be judged and added to the pool, and these relevance judgments can be used as feedback to improve the learning process, thus generating more relevant documents in subsequent rounds. The quality of the pools thus generated can be measured in two ways: (1) At what rate are relevant documents found (recall

	TREC				
Pool	3	5	6	7	8
Depth	$n = 40$	$n = 82$	$n = 79$	$n = 103$	$n = 129$
1	19	38	38	32	40
2	39	68	67	55	69
3	47	98	95	76	95
4	60	126	120	95	119
5	73	153	146	114	144
6	85	181	172	134	167
7	96	208	197	152	191
8	107	234	221	170	215
9	118	262	246	189	238
10	129	288	271	207	260
15	183	418	393	297	379
20	235	543	513	389	494
30	336	791	743	571	717
40	436	1034	969	754	939
50	531	1273	1191	936	1155
60	626	1509	1410	1114	1366
70	718	1745	1629	1299	1574
80	811	1978	1845	1486	1777
90	903	2206	2058	1675	1978
100	995	2434	2271	1860	2176

Table 1: The size of the pool (per query) for various pool depths if the pooling is performed TREC-style. Here n is the number of input systems in the given data set.

percentage as a function of total judgments)? (2) How well do these pools evaluate the retrieval systems (score or rank correlations vs. "ground truth")? In our experiments using TREC data, Hedge found relevant documents at rates *nearly double* that of benchmark techniques such as TREC-style depth pooling. These Hedge pools were found to evaluate the underlying retreival systems much better than TREC-style depth pools of an equivalent size (as measured by Kendall's τ rank correlation, for example). Finally, these Hedge pools seemed particular effective at properly evaluating the best underlying systems, a task difficult to achieve using small pools.

In the sections that follow, we first describe our methodology for simultaneously solving the metasearch, pooling and system evaluation problems using Hedge. We then describe the results of our methodology in experiments conducted on TREC data. Finally, we conclude by mentioning some possible extensions of this work.

2. METHODOLOGY

The intuition for our methodololgy can be described as follows. Consider a user who submits a given query to multiple search engines and receives a collection of ranked lists in response. How would the user select documents to read in order to satisfy his or her information need? In the absence of any knowledge about the quality of the underlying systems, the user would probably begin by selecting some document which is "highly ranked" by "many" systems; such a document has, in effect, the collective weight of the underlying systems behind it. If the selected document were relevant, the user would begin to "trust" systems which retrieved this document highly (i.e., they would be "rewarded"), while the user would begin to "lose faith" in systems which did not retrieve this document highly (i.e., they would be "punished"). Conversely, if the document were non-relevant, the user would punish systems which retrieved the document highly and reward systems which did not. In subsequent rounds, the user would likely select documents according to his or her faith in the various systems in conjunction with how these systems rank the various documents; in other words, the user would likely pick documents which are *ranked highly* by *trusted* systems.

How can the above intuition be quantified and encoded algorithmically? Such questions have been studied in the machine learning community for quite some time and are often referred to as "combination of expert advice" problems. One of the seminal results in this field is the Weighted Majority Algorithm due to Littlestone and Warmuth [11]; in this work, we use a generalization of the Weighted Majority Algorithm called **Hedge** due to Freund and Schapire [7].

Hedge is an on-line allocation strategy which solves the combination of expert advice problem as follows. (See Figure 1.) **Hedge** is parameterized by a tunable learning rate parameter $\beta \in [0,1]$, and in the absence of any *a priori* knowledge, begins with an initially uniform "weight" w_i^1 for each expert i (in our case, $w_i^1 = 1\ \forall i$). The relative weight associated with a expert corresponds to one's "faith" in its performance.

For each round $t \in \{1, \ldots, T\}$, these weights are normalized to form a probability distribution $\mathbf{p}^t$ where

$$\mathbf{p}_i^t = \frac{w_i^t}{\sum_j w_j^t},$$

Algorithm Hedge(β)

Parameters: $\beta \in [0,1]$
initial weight vector $\mathbf{w}^1 \in [0,1]^N$ **with** $\sum_{i=1}^N w_i^1 = 1$
number of trials T

Do for $t = 1, 2, \ldots, T$

1. **Choose allocation** $\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$.
2. **Receive loss** $\ell^t \in [0,1]^N$ **from environment.**
3. **Suffer loss** $\mathbf{p}^t \cdot \ell^t$.
4. **Set the new weight vector to be** $w_i^{t+1} = w_i^t \beta^{\ell_i^t}$.

Figure 1: Hedge Algorithm

and one places $\mathbf{p}_i^t$ "faith" in system i during round t.

This "faith" can be manifested in any number of ways, depending on the problem being solved. If the underlying experts are making predictions about which stocks will rise in the next trading day, one might invest one's money in stocks according to the weighted predictions of the underlying experts. If a stock goes up, then each underlying expert i which predicted this rise would receive a "gain," and the investor would also receive a gain in proportion to the money invested, p_i^t. If the stock goes down, then each underlying expert i which predicted a rise would suffer a "loss," and the investor would also suffer a loss in proportion to the money invested. This is encoded in **Hedge** as follows. In each round t, expert i suffers a *loss* ℓ_i^t, and the algorithm suffers a weighted average (*mixture*) loss[1] of $\sum_i p_i^t \ell_i^t$.

Finally, the **Hedge** algorithm updates its "faith" in each expert according to the losses suffered in the current round, $w_i^{t+1} = w_i^t \beta^{\ell_i^t}$. Thus, the greater the loss an expert suffers in round t, the lower its weight in round $t+1$, and the "rate" at which this change occurs is dictated by the tunable parameter β.

Over time, the "best" underlying experts will get the "highest" weights, and the cummulative (mixture) loss suffered by **Hedge** will be not much higher than that of the best underlying expert. Specifically, Freund and Schapire show that if $L_i = \sum_t \ell_i^t$ is the cummulative loss suffered by expert i, then the cummulative (mixture) loss suffered by **Hedge** is bounded by

$$L_{\mathbf{Hedge}} \leq \frac{\min_i\{L_i\} \cdot \ln(1/\beta) + \ln N}{1-\beta}$$

where N is the number of underlying experts.

2.1 Hedge Application

We employ the **Hedge** algorithm to simultaneously solve the problems of metasearch, pooling, and system evaluation as follows. On a per query basis, each underlying retrieval system is an "expert" providing "advice" about the relevance of various documents to the given query. We must define a method for selecting likely relevant documents based

[1] It is assumed that the losses and/or gains are bounded so that they can be appropriately mapped to the range $[0,1]$.

on system weights and document ranks, and we must also define an appropriate loss that a system should suffer for retrieving a particular relevant or non-relevant document at a specified rank. While a loss function which converges to some standard measure of performance such as average precision might be desirable, we instead work with a simpler but related loss. The loss function is designed to reflect a document's complete contribution to a system's *total precision* (TP)—the sum of the precisions at all document levels. It is defined for document d_k at rank r_k by: $\ell = \frac{1}{2} \cdot (-1)^{I(\mathrm{rel}(d_k))} \cdot \sum_{r=r_k}^{r_{\max}} \frac{1}{r}$, where $I(\mathrm{rel}(d_k))$ is an indicator function for the relevance of document d_k and $r_{\max}$ is the total number of documents retrieved for this query. In the limit of complete relevance judgments, one can show that the total loss of a system converges to the negative of the total precision plus a system-independent constant. For our purposes, this measure demonstrates a close empirical relationship to other popular measures of performance (such as average precision at relevant documents) while it has the advantage of being simple and "symmetric" (the magnitude of the loss or gain is independent of relevance). We note that this loss can be easily approximated since its magnitude is the difference between two harmonic numbers. Let $H_k = \sum_{i=1}^{k} 1/i$ be the i-th harmonic number; we then have

$$\begin{aligned}
\ell &= \frac{1}{2} \cdot (-1)^{I(\mathrm{rel}(d_k))} \cdot \sum_{r=r_k}^{r_{\max}} \frac{1}{r} \\
&= \frac{1}{2} \cdot (-1)^{I(\mathrm{rel}(d_k))} \cdot \left(\sum_{r=1}^{r_{\max}} \frac{1}{r} - \sum_{r=1}^{r_k - 1} \frac{1}{r} \right) \\
&= \frac{1}{2} \cdot (-1)^{I(\mathrm{rel}(d_k))} \cdot (H_{r_{\max}} - H_{r_k - 1}) \\
&\approx \frac{1}{2} \cdot (-1)^{I(\mathrm{rel}(d_k))} \cdot (\ln r_{\max} - \ln(r_k - 1)) \\
&= \frac{1}{2} \cdot (-1)^{I(\mathrm{rel}(d_k))} \cdot \ln \frac{r_{\max}}{r_k - 1}.
\end{aligned}$$

Note that the magnitude of this loss is highest for documents which are highly ranked.

Given this loss function, we implement a simple pooling strategy designed to maximize the learning rate of the **Hedge** algorithm. At each iteration, we select the document which would maximize the weighted average (mixture) loss if it were non-relevant. Since the loss suffered by a system is high Since this is exactly the unlabelled document with the maximum expectation of relevance as voted by a weighted linear combination of the systems, the strategy is also appropriate for selecting documents to be output in a metasearch list.

3. RESULTS

TREC	MNZ	COND	Hedge-0	$\%MNZ$	$\%COND$
3	0.423	0.403	0.418	-0.012	+0.037
5	0.294	0.307	0.309	+0.051	+0.006
6	0.341	0.315	0.345	+0.012	+0.095
7	0.320	0.308	0.323	+0.009	+0.049
8	0.350	0.343	0.352	+.0014	+0.026

Table 2: Hedge-0 Method vs. Metasearch Techniques Comb-MNZ and Condorcet.

The Hedge algorithm demonstrated uniformly excellent performance across all TRECs tested (TRECs 3, 5, 6, 7, and 8) in all three measures of performance- as an online metasearch engine, as a pooling strategy for finding large fractions of relevant documents, and finally as a mechanism for rapidly evaluating the relative performance of retrieval systems.

In the following discussion, we demonstrate results using standard TREC-style pools of depth k and Hedge pools of an equivalent size. Depth-n will refer to an evaluation with respect to a TREC-style pool of all documents retrieved by some system at depth n or less, and Hedge-m refers to an evaluation with respect to the pool generated by the Hedge algorithm after judging m documents. Standard TREC routines were used to evaluate the systems with respect to these pools, yielding mean average precision (MAP) scores for the underlying systems as well as the rankings of those systems induced by these scores.

Although the three goals of the algorithm, as stated, are somewhat intertwined, the action of the Hedge algorithm as an online metasearch engine may be seen as enabling those of pooling and system evaluation. The high quality ranked list produced by the metasearch engine consists of documents ranked in order of expected relevance, and therefore provides a foundation for document pooling— either performed iteratively, or in multi-document batches. Documents with high probability of relevance, in turn, prove to be good discriminators of system quality, and thus, pools generated in this manner enable rapid evaluation of system performance.

We examine the performance of the Hedge system as an evolving metasearch list. At each iteration, the document chosen to be judged is the one with the highest expectation of relevance. Thus, it is appropriate to build an online metasearch list from these selections. To complete the metasearch list, the remaining documents are likewise ranked by weighted linear combination.

As shown in Table(**??**), the Hedge algorithm begins in the lower limit of 0 online relevance judgements with a baseline MAP score which is equivalent or slightly better, in almost all instances, to the performance of the well known CombMNZ and Condorcet metasearch methods (the lower dashed lines in Figure 2(a). Proceeding from the 0 level, Hedge online metasearch results quickly surpass those of the best underlying retrieval system (the upper dashed lines). In TRECs 3,5, and 7, the best system is equalled in 10 or fewer judgements. TRECs 6 and 8 require somewhat more judgements to achieve the performance of the best underlying system. This reflects the fact that in both competitions the best systems are outliers, with few retrieved documents in common with the generic pack, and thus Hedge must initiate more judgements prior to their discovery.

Figure 2(b) demonstrates the algorithm's success in finding relevant documents. The vertical axis corresponds to percentage of total relevant documents and the dashed line indicates the number of relevant documents present in the Depth-n pools for depths 1-10, 15, and 20. Hedge performance far surpasses the discovery rates of the depth pooling method when compared at equivalent depths of documents judged. But even more indicative of the success of the algorithm is a comparison of the number of judgements required to achieve equivalent recall percentages. For example, examining the TREC 8 curves along the horizontal axis, we see that the Depth-n method requires approximately

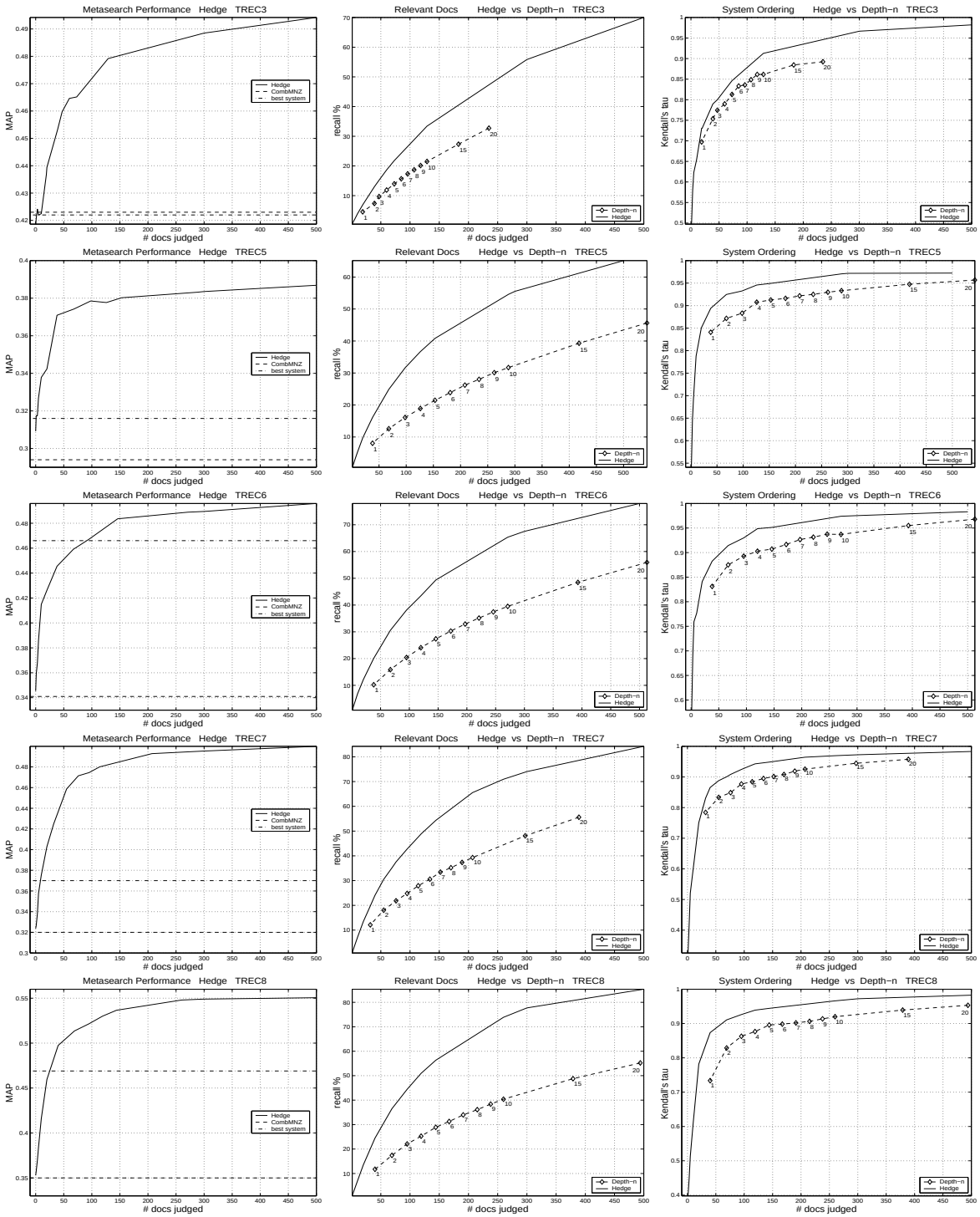

Figure 2: (a)Hedge-m: metasearch performance. (b)Hedge-m vs. Depth-n: percent of total relevant documents discovered. (c) Hedge-m and Depth-n vs. actual ranks: k-τ.

104 judgments to match the Hedge-40 return rate, and the Hedge-69 rate (36 percent) is unmatched until Depth-8 (199 judgments). After almost 500 judgements, Depth-20 has found only approximately 55% of relevant documents— a rate achieved by Hedge in less than 150 judgements.

Figure 2(c) compares the system rankings produced by the Hedge algorithm against those of Depth-n pooling at equivalent levels of judged documents using the Kendall's τ measure. Again, the dashed line indicates the results of system evaluations performed using standard TREC routines, given Depth-n pools of size 1-10,15, and 20. Examination of TREC 8 demonstrates typical performance. At 40 docu-

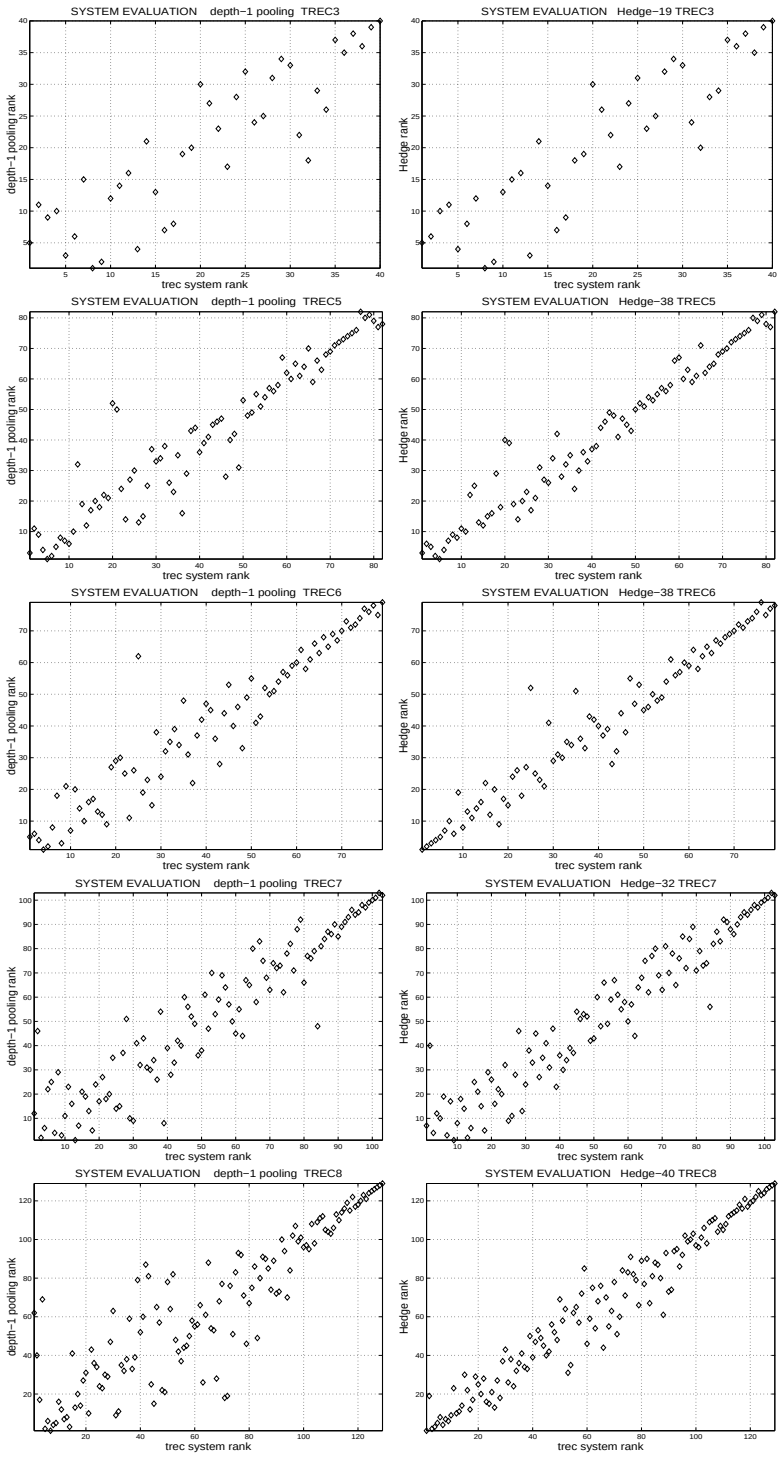

Figure 3: Depth-1 and equivalent Hedge-m rankings vs. actual ranks.

ments, the τ for Hedge is 0.87. This compares with 0.73 for Depth-1— a substantial improvement. Likewise, Hedge-69 achieves an accuracy of 0.91, vs. a Depth-2 accuracy of 0.73.

Next, comparing along the horizontal axis the pool depths required to achieve equivalent rates of ordering accuracy, we see that in order to achieve an accuracy of 0.87(Hedge-40), the equivalent Depth-3 pool requires 95 judgments. An accuracy of 0.91 (Hedge-69) corresponds to a system approaching Depth-8 (198 judgments).

Finally, a look at the scatter plots in Figure 3 demonstrates another aspect of algorithm performance in the evaluation of system orderings which is somewhat obscured in the traditional Kendall's τ measure. Each pair of plots shows Depth-1 and equivalent Hedge-n predicted ranks vs.

the actual TREC rankings. Note in these plots that the rankings proceed from best systems in the lower left corner to worst in the upper right. TREC 3 plots are ambiguous due to the relatively low number of systems in the competition, but later TRECs demonstrate the common tendency toward difficulty in establishing rankings for the best systems.

While poor systems tend to be easily identifiable due to their lack of commonality with any other systems, many of the better systems likewise exhibit a great deal of variance in returned documents. Thus, while poor systems may be well ranked using standard techniques with depth pools as small as Depth-1, the better systems (and for most purposes, the systems of most interest) tend to be the more difficult to rank correctly. As the Kendall's τ measure of accuracy in system ordering treats documents at all rank levels equally, much of the qualitative superiority of algorithms which perform well in classifying higher ranked systems is obscured by commonly good performance in ranking the poorer systems. Examination of tightened patterns of the Hedge plots in the regions of interest indicates that performance of the algorithm in evaluating system orderings is somewhat better than the excellent performance demonstrated in Figure 2(c).

4. CONCLUSIONS

We have shown that metasearch can be a powerful tool in both (1) selecting small pools of documents to judge in order to assess the performance of retrieval systems and (2) assessing the performance of retrieval systems in the absence of relevance judgments.

This work leaves open a number of interesting questions. First, our use of hard thresholds (i.e., "top k" documents from the metasearch list) in choosing both the meta-pool and the pseudo-relevant documents is somewhat simplistic, though empirically effective. One approach to improve the performance of meta-pooling would be to combine meta-pooling and pseudo-evaluation. For example, one could assume, by default, that the top k documents of the metasearch list are relevant (as we do in pseudo-evaluation), and from these top k documents choose a subset to actually evaluate (as in meta-pooling). Note that meta-pooling, as described, dictates evaluating the top documents from the metasearch list. However, these documents are very likely to be relevant; given a limited number of relevance a judgments, these judgments would almost certainly be better spent on documents somewhat deeper in the metasearch list.

Second, for pseudo-evaluation and for meta-pooling with difficult data sets (such as TREC 8), the challenge is in properly identifying the top retrieval systems in the collection. It is likely that better metasearch techniques and/or the more judicious use of relevance judgments will be necessary to properly identify these top systems with few and/or no relevance judgments.

5. REFERENCES

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

[2] J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1], pages 276–284.

[3] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In W. B. Croft and C. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–181, Dublin, Ireland, July 1994. Springer-Verlag, London.

[4] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In Croft et al. [5], pages 282–289.

[5] W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors. *Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, Aug. 1998. ACM Press, New York.

[6] E. A. Fox and J. A. Shaw. Combination of multiple searches. In D. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 243–249, Gaithersburg, MD, USA, Mar. 1994. U.S. Government Printing Office, Washington D.C.

[7] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, Aug. 1997.

[8] D. Harman. Overview of the third text REtreival conference (TREC-3). In D. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 1–19, Gaithersburg, MD, USA, Apr. 1995. U.S. Government Printing Office, Washington D.C.

[9] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–188, 1995.

[10] 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.

[11] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[12] 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], pages 267–275.

[13] M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In K. Kalpakis, N. Goharian, and D. Grossman, editors, *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 538–548. ACM Press, November 2002.

[14] C. C. Vogt. How much more is better? Characterizing the effects of adding more IR systems to a combination. In *Content-Based Multimedia*

Information Access (RIAO), pages 457–475, Paris, France, Apr. 2000.

[15] 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.

[16] J. Zobel. How reliable are the results of large-scale retrieval experiments? In Croft et al. [5], pages 307–314.