Pergamon

Information Processing & Management, Vol. 31, No. 3, pp. 431-448, 1995
Copyright © 1995 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0306-4573/95 $9.50 + .00

0306-4573(94)00057-3

COMBINING THE EVIDENCE OF MULTIPLE QUERY REPRESENTATIONS FOR INFORMATION RETRIEVAL[1]

N.J. BELKIN and P. KANTOR
School of Communication, Information & Library Studies, Rutgers University,
New Brunswick, NJ 08901-1071, U.S.A.
e-mail: [belkin/kantorp]@cs.rutgers.edu

and

E.A. Fox and J.A. Shaw
Department of Computer Science, Virginia Tech, Blacksburg, VA 24061-0106, U.S.A.
e-mail: [fox/joeshaw]@cs.vt.edu

Abstract—We report on two studies in the TREC-2 program that investigated the effect on retrieval performance of combination of multiple representations of TREC topics. In one of the projects, five separate Boolean queries for each of the 50 TREC routing topics and 25 of the TREC ad hoc topics were generated by 75 experienced online searchers. Using the INQUERY retrieval system, these queries were both combined into single queries, and used to produce five separate retrieval results for each topic. In the former case, progressive combination of queries led to progressively improving retrieval performance, significantly better than that of single queries, and at least as good as the best individual single-query formulations. In the latter case, data fusion of the ranked lists also led to performance better than that of any single list. In the second project, two automatically produced vector queries and three versions of a manually produced P-norm extended Boolean query for each routing and ad hoc topic were compared and combined. This project investigated six different methods of combination of queries, and the combination of the same queries on different databases. As in the first project, progressive combination led to progressively improving results, with the best results, on average, being achieved by combination through summing of retrieval status values. Both projects found that the best method of combination often led to results that were better than the best performing single query. The combined results from the two projects have also been combined by data fusion. The results of this procedure show that combining evidence from completely different systems also leads to performance improvement.

1. INTRODUCTION

In this paper, we discuss two different projects within the TREC-2 program, conducted at two different institutions, each having the same foci. These were to investigate the effect of making use of several different formulations of a single information problem on information retrieval (IR) system performance, and to investigate different methods for combination of these formulations. Although these two different projects, one at Rutgers University, the other at Virginia Tech (hereafter, Rutgers and VT, respectively), were conceived and conducted independently, both the bases for their work and their results were remarkably similar. Our intention here is to describe the different methods used by the two projects, and their results, in order to demonstrate the robustness and general applicability of this approach.

The basis for this work lies in both theory and empirical evidence. From the empirical point of view, it has been noted for some time that different representations of the same information problem retrieve sets (or ranked lists) of documents that contain different rel-

[1]This paper is based upon two papers published in the proceedings of TREC-2, Belkin *et al.* (1994), and Fox & Shaw (1994).

evant, as well as non-relevant documents (see, e.g., McGill *et al.*, 1979; Katzer *et al.*, 1982; Fox, 1983; Saracevic & Kantor, 1988). There is some implication from this evidence (made explicit by Saracevic & Kantor, 1988) that taking account of the different results of the different formulations could lead to retrieval performance better than that of any of the individual query formulations.

From the theoretical point of view, IR can be considered as a problem of inference (see, e.g., van Rijsbergen, 1986). That is, IR is concerned with estimating, given available evidence about such things as information problems and documents (or in general, retrievable information objects), the likelihood (or probability, or degree) of relevance of a document to the information problem. From this point of view, different query formulations constitute different sources of evidence that could be used to infer the probable relevance of a document to an information problem, and it is thus reasonable to consider ways to use (i.e., combine) these sources of evidence in the inference process.

These ideas are general to any source of evidence that might be used for IR, such as the evidence of different retrieval techniques, or different document representation techniques, or, in general, different IR systems. One aspect of the Rutgers project uses the example of different query formulations as a simulation of the general problem of combination of evidence from different systems.

An additional argument is available for the special case of different query representations. That is, if we consider an information problem to be a complex and, in general, difficult-to-specify entity (see, e.g., Taylor, 1968; Belkin *et al.*, 1982), then we might conclude that each different representation, derived from some statement by the user, is a different interpretation of the user's underlying information problem, highly unlikely to be like anyone else's (or any other system's) interpretation. Given the empirical evidence, whether any one such interpretation is 'better' than another seems moot. However, we might say either that each captures some different, yet pertinent aspect of the user's underlying problem, or that those aspects of the different interpretations common to them all (or to more than one) reflect some 'core' aspect of the problem. Although techniques for making use of the different interpretations might vary according to which of these two views one takes, the general position suggests that it will always be a good idea to take advantage of as many such interpretations as possible. Both projects therefore considered the issue of combination of different query representations within the 'same' IR system.

A further, important combination issue was addressed by the VT project alone. This was the problem of how to combine the results of searching different document collections with the same query formulation, a problem that becomes practically quite important when dealing with different large, heterogeneous databases.

The Rutgers project considered the problem of inference in IR at two levels of analysis. The first level, as introduced by Turtle & Croft (1991a), asks about the effect of combining the evidence obtained when two or more formal query statements for the same information problem are used within one IR system. The second level, which was simulated in the Rutgers study, asks about combination of evidence provided by two or more distinct systems, ranking the same set of documents in response to the same problem. The VT project also followed this same general pattern, with combination of results of searches with different query formulations being at the first level, and combination of results of searches with the same query over different databases being at the second level. To distinguish these two levels, and in keeping with earlier discussions of the issues involved, we henceforth refer to the direct combination of evidence from the same system as "query combination," and we refer to the combination of evidence from differing systems as "data fusion." Earlier work by us has also addressed various aspects of this general question (Fox *et al.*, 1993; Belkin *et al.*, 1993). Those studies specifically address the question of query combination, and were direct precursors to the projects reported on here. In Section 2, we present a framework for combination of evidence, which relates our ideas of query combination to those of data fusion. This framework also then gives us a means to relate the results of the two projects reported on in this paper to one another, and to interpret their implications in terms of general concepts of combination of evidence.

2. CONCEPTS OF QUERY COMBINATION AND DATA FUSION

2.1 *Query combination vs data fusion*

We use the phrase "query combination" to describe combination of evidence that takes place internal to a single system. In this case, the rule of combination can take advantage of details related to the internal representation of similarity between documents and queries. Although the combination of evidence scheme used in the Rutgers experiments involves submission of a single compound query, made up of a number of different query formulations for the same information problem, the specific workings of the IR system they used produced, in fact, the weighted sum of similarity scores. This is mathematically equivalent to the methods for query combination used in the VT experiments.

VT used several different rules of query combination, all of which begin with the sum of the similarity scores, between a topic and document, summed over the several schemes considered. The rule called CombSUM uses this sum as the new retrieval score. The rule called CombANZ divides this sum by the number of schemes for which the document appeared in the top 1000 for the given topic. It is somewhat akin to a logical OR. The rule called CombMNZ multiplies the sum of scores by the number of schemes for which the document appeared in the top 1000, for the given topic.

A query combination rule of this type was also used by us in experiments combining the Rutgers and VT results. Scores for each system were normalized by dividing all scores by the largest similarity score between any document and all of the topics in the same TREC group (that is, ad hoc and routing similarity scores were normalized separately). One should note that this procedure can produce quite un-intuitive results when very different systems are combined. For example, when a Boolean system is combined with an ordinary ranking system, all of the scores produced by the Boolean system will rank above all but at most one of the scores produced by the ordinary system. The more nearly a system approximates Boolean scoring, the more it will dominate a procedure of this kind. This is related to a broader class of techniques involving pattern recognition in either "rank space" or "attribute space" (Kantor, in preparation).

When two very different and distinct systems are considered, the internal representations they use to produce their document rankings may be, in general, incommensurable. In such cases, the combination of evidence from the different systems must be based on rank information alone. Such combination we call here "decision level *data fusion*." Below, we discuss methods for combining evidence in such circumstances.

2.2 *Data fusion logics*

A data fusion logic is an extension, to the case of ranked lists, of the logical operations that can be performed on retrieved sets. When there are only two sets, the operations are AND and OR. For ranked lists these are implemented by moving a pointer down both lists simultaneously, and forming the intersection (for AND, union for OR) of the sets of items appearing above the pointer on the two lists.

For three lists there are several candidate logics. The most important are the ones we call MAX, MIN, and MED. To understand these names, note that the ranked list formed by the AND logic just described is one in which each item appears with a rank given by the MAXimum of its three ranks in the input lists. Similarly, the list formed by OR can be achieved by using the MINimum of the ranks that the item has in the input lists. These are both sensitive to the presence of a very bad system, which positions relevant items at random in the retrieved list. Under the MAX fusion we must wait a long time to see a relevant item. Under the MIN fusion, irrelevant items will be accepted as soon as they appear on the poorer list. The MEDian fusion rule assigns to each item a rank that is the median of the three ranks that locate it in three input lists. It is expected to be more stable against the presence of a single poor (or "noisy") input list.

In Fig. 1 we show a single-level curve (or threshold) for each of three rules, or logics, that might be assigned to combine the outputs of two search engines. Each document (or, in general, each information object) is represented by a point in a plane whose coor-

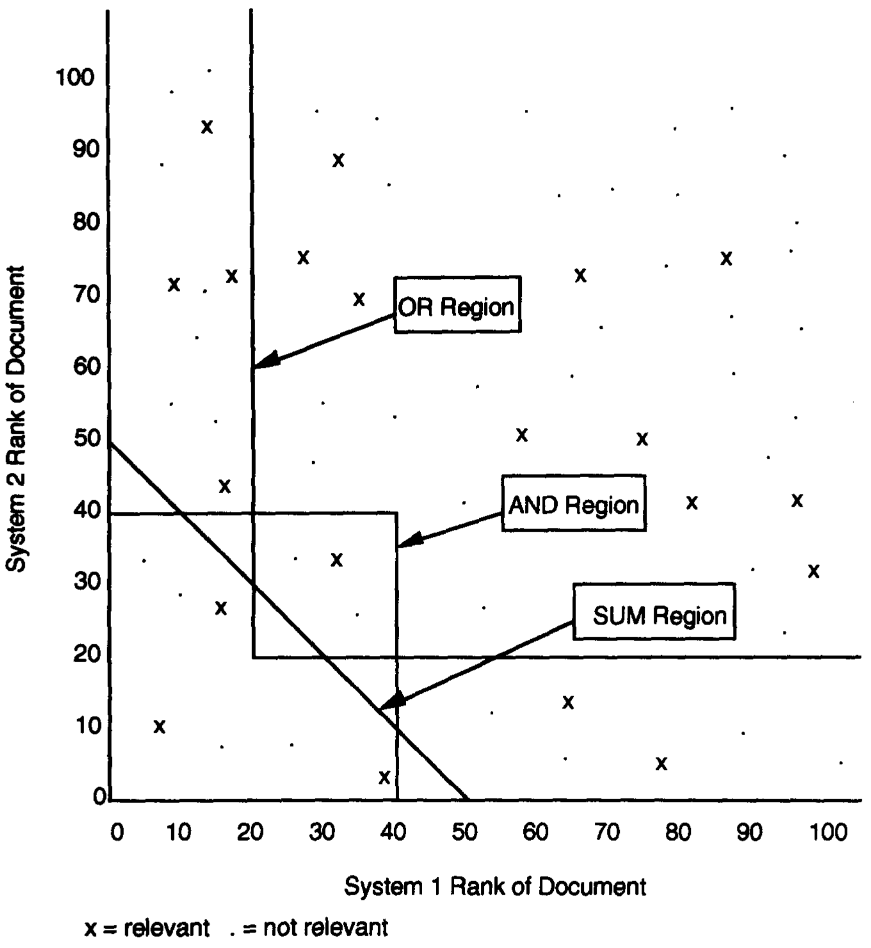

Fig. 1. Three rules for data fusion.

dinates are the ranks assigned to that item by the two systems. The dots represent nonrelevant items, and the Xs represent relevant ones.

The AND rule (which uses the MAXimum of the ranks) produces an expanding sequence of squares, which eventually sweeps up all of the items. The OR rule (which uses the MINimum of the ranks) produces an expanding sequence of L-shaped regions (their complements are squares) that eventually sweeps up all of the items. Finally, the SUM-OF-RANKS rule, in which the combined list is constructed by summing the ranks of the documents in the two lists, produces triangular regions that eventually sweep up all of the items.

Note that combination based on the maximum of the ranks assigned by several schemes is "opposite" to combination based on the maximum of the similarity scores assigned by several schemes (as done in the VT project, with its CombMAX rule; see section 3.3), as ranks vary inversely to similarity scores. However, since the ranks are not linear in the scores, we cannot say that, for example, MAX based on ranks is the same as MIN based on scores (the OR logic). A score-based logic, when represented on the graph of Fig. 1, would result in rectangles whose proportions change as they sweep across the entire quadrant.

2.3 *Justification(s) for data fusion*

The ultimate justification for data fusion, and combination of evidence in general, will be found (or lost) in empirical investigations such as those reported here. But there are a range of interesting reasons why it might be expected to work. We briefly mention several of them here.

2.3.1 *Imprecision of topic formulations, when interpreted by diverse humans or systems.* No utterance completely captures a unique meaning for all readers. Thus, it might be expected that diverse interpretations of an utterance are scattered, with some probability distribution, about the "intention" of the utterer. In addition, the interpretation of the "judge" (in the TREC experiments each topic had a single judge) may also be displaced from the "intention." It can be shown [Kantor, in preparation] that under a standard vector model for queries, in this situation some kind of combination of queries will be more effective than even the best of the input queries a substantial fraction of the time.

2.3.2 *Signal processing arguments.* In a signal processing model, at any choice of threshold (t) there is some probability ($d(t)$) that a relevant item will be retrieved, and some probability ($f(t)$) that a nonrelevant item will be retrieved. If two systems are independent, and using the same threshold, the corresponding probabilities for the AND-combined systems are (suppressing for a moment the threshold parameter) $d(1)d(2)$ and $f(1)f(2)$. In general (Cherikh, 1989), this can be expected to yield a system with a better operating characteristic than either of the independent systems. Intuitively, there are so many nonrelevant documents that could be brought to the top of the list as "noise" that there is little chance that two independent systems would bring the same ones to the top of the list. There are far fewer relevant documents, and so the chance that those will appear at the top of both lists is correspondingly greater.

2.3.3 *Expansion of parameter space.* When some objective function (say, precision at 100 documents) is to be maximized, any expansion of the space of possible retrieval rules can lead to, at worst, the same best value and, at best, a better value. When the outputs of two systems are already available to use as solution choices, expanding the parameter space by considering different combinations of them can, logically, only improve the performance. Of course, it may be that the best combination of two inputs (say, A and B) is "ignore A and use only B." The justification for exploring data fusion is that this does not happen all of the time. In fact, there are a substantial number of cases (in our experiments) where it does not happen. Our model moves from the purely logical rigor of this argument to an empirical investigation when we specify distinct operations for the fusion, rather than always selecting the best fusion rule after the fact.

3. METHODS

3.1 *Introduction*

The two studies complement one another in terms of the methods used for constructing and combining the queries, and in terms of the retrieval systems to which the queries were put. Previous work (e.g., Turtle & Croft, 1991a), as well as the theoretical positions discussed in section 1, suggests that queries that are in some sense independent would be most useful in combination. The Rutgers study addressed this issue by having five different experienced searchers construct queries for each topic. The VT study took another approach, constructing two different types of queries for each topic, one manually constructed Boolean, the other automatically constructed vector, with variations on each. The methods of combination were somewhat more similar. Rutgers combined queries directly, and also combined ranked lists according to a rule for choosing rank of a document in the final list based on its ranks in the five lists. VT studied six different rules for combination, using retrieval status values (RSVs) for determining position in the combined list. Some of their rules were similar to the rank-based rules studied at Rutgers; others were more similar to the query combination technique. And the two groups used two quite different systems for indexing and retrieval, Rutgers using the INQUERY system (Callen *et al.*, 1992), VT a modified version of SMART (Buckley, 1985).

3.2 *Rutgers methods*

The Rutgers study is based on analysis of the entire set of 50 routing topics, and a selected sample of 25 ad hoc topics. The sample was stratified according to the domain of the topic, in an effort to represent the distribution of domains in the entire set of ad hoc topics.

The query formulations used in this study were generated by volunteer online searchers, all of whom were experienced users of large bibliographic retrieval systems. Seventy-five searchers participated in the study, 50 for the routing topics and 25 for the ad hoc topics. For each of the TREC topics in our analysis, five different searchers were asked to generate Boolean search statements. Each of the volunteer searchers generated a query formulation for five different topics, resulting in five independently generated query formulations for each topic. After formulating each query, searchers answered four questions about the process: how long it took to formulate the query; how closely related the topic was to their normal searches; how easy is was for them to formulate the query; and the extent to which they had enough information to construct the query. In addition to these questionnaire items, the ad hoc searchers were also asked how many years of online searching experience they had. Searchers for the routing queries were not asked this question.

The Rutgers study used the INQUERY retrieval engine (version 1.5) developed at the University of Massachusetts (Callan *et al.*, 1992). INQUERY is a probabilistic inference network-based system, based upon the idea of combining multiple sources of evidence in order to infer plausibly the relevance of a document to a query. The underlying formalism is that of a Bayesian probabilistic inference network (Pearl, 1988), which provides strict rules for how to combine sources of evidence. Turtle and Croft (1991b) give a detailed description of the model and its implementation; a more general description is available in Belkin and Croft (1992). Here, we note a few characteristics of the system that are germane to the project at hand.

First, INQUERY provides a natural means for combination of multiple query formulations, as a function of its design. Second, it incorporates a large set of operators, which allow, in addition to sophisticated natural language query formulations, complex Boolean formulations. The Boolean operators in INQUERY are not strict, however, allowing ranking of output, and leading to significantly better performance than strict Boolean retrieval (Turtle & Croft, 1991a).

Each of the Boolean query formulations produced by the searchers was translated into INQUERY syntax. Two methods of query combination were then used, each specific to the TREC-2 tasks of responding to ad hoc and routing topics. The first, labeled "prior," was applied to the ad hoc topics. In this procedure, the five query formulations for each topic are combined directly, into one query, using the INQUERY "unweighted sum" operator. This query is then used as the search statement in the experiments.

The second combinatorial procedure, called "wtrain," was used for the routing topics. Here, a separate search for each separate query formulation was done for all 50 topics, on the training set supplied from the TREC-1 data. From these results, the average 11-point precision (in the "official" results reported at TREC-2; precision at 100 documents for the "unofficial results" reported in this paper) of each query formulation was used as a weight for that formulation in the combination of all five formulations for each topic. For this, INQUERY's "weighted sum" operator was used. This procedure corresponds to constructing a simple combined query, learning something about how that query's components perform on the current database, and taking account of that evidence to modify the query formulation for searching the next database.

These methods of combining queries give a very straightforward way to test our hypotheses about the effectiveness of multiple sources of evidence. For the Rutgers experiments (as opposed to the results submitted to TREC-2, which were just the prior and wtrain results as described above), the query formulations for both ad hoc and routing topics were divided into five different groups. In each group, each topic was represented by one query, and no searcher was represented more than once in any one group. This distribution was meant to control for possible searcher effects. Retrieval runs were then done for each single group, and for each combination of groups, for both ad hoc and routing topics. This allowed comparison of retrieval performance of different levels of query combination, and comparison of retrieval performance of combined queries with uncombined.

Data fusion was accomplished by a list-merging method that is the natural extension of a 3-out-of-5 data fusion logic in the binary case. This is a version of the MEDian rule discussed in section 2.2. The basic data used were the five lists of documents retrieved by the five different query formulations for each topic. Every document has some rank in each of the five lists being joined together. An effective rank is calculated by taking the third highest of the five ranks that the document has. This has the same effect as moving a threshold along the list of effective ranks, and including a document in the output when it has appeared on three of the lists. Since there are five scores all together, this can also be thought of as a median rule.

In practice, to maintain consistency with other parts of our work, Rutgers did not calculate the rank of every document, but worked with the lists of the top 1000 documents produced in response to each query formulation. This meant that some documents would appear on all five of the lists, others on just four, or three, or even fewer. Of course, the whole logic of data fusion suggests that those that appear on more lists are more likely to be relevant. This was implemented, in fact, by forming a combined sort key consisting of (10-degeneracy, 3rd rank). The degeneracy is the number of lists on which a specific document appears in the top 1000. A lexicographic sort was used, so that all items with degeneracy 5 appeared before any items with degeneracy 4, and so on. Within a given degeneracy, items with lower values for the 3rd rank were ranked first.

3.3 *VT methods*

All of the queries used in the Virginia Tech TREC-2 experiments were created from the topic descriptions provided by NIST. Two types of queries were used, *P*-norm extended Boolean queries (Fox, 1983) and natural language vector queries. A single set of *P*-norm queries was created, but it was interpreted multiple times with different operator weights (*P*-values). Two different sets of vector queries were created from the topics, one set containing information from more sections of the topic descriptions.

The *P*-norm queries were manually constructed by one of the project researchers as complex Boolean expressions using AND and OR operators. The query terms were derived primarily from the Title, Description, Narrative, and Concepts of the topic descriptions, with a select few queries being supplemented by additional terms selected by the researcher to fill obvious omissions from the topic descriptions. Phrases were simulated using AND operators, since the queries were intended only for soft-Boolean evaluation. The query terms were not specifically weighted; uniform operator weights (*P*-values) of 1.0, 1.5, and 2.0 were used on different evaluations of the query set; these three runs are designated Pn1.0, Pn1.5, and Pn2.0. The SMART ann weighting scheme (Buckley, 1985),

$$\text{term-weight} = 0.5 + 0.5 \cdot tf/\text{maximum-}tf\text{-in-document},$$

was used to evaluate the queries, based on results obtained by Fox (1983) for the much smaller classical document collections, and verified by similar experiments on portions of the TREC-2 training collection.

Two sets of vector queries were automatically constructed directly from the Title, Description, Concepts, and Definitions sections of the topic descriptions. One of the sets also included the Narrative section of the topic descriptions; this set is referred to as the long vector (LV) query set, for obvious reasons, while the other is referred to as the short vector (*SV*) query set. The LV query set averaged over 106 terms per query, whereas the SV query set averaged fewer than 56 terms per query. The automatic construction was simplistic: A term occurring multiple times in a topic was simply included multiple times in the query formulation, which has the effect of weighting that term weight by its frequency in the query.

The two sets of vector queries were evaluated using the standard cosine correlation similarity method as implemented by SMART. The same SMART ann weighting scheme used for the *P*-norm queries was used on the vector queries for several reasons. First, a weighting scheme that did not use any collection statistics was needed for the routing experiments. Second, the methods used in combining runs required a similar range of possible similarity values produced by each run. Finally, the necessity of merging results from each subcollection into a single sets of results, due to a limited amount of disk space, was simplified since the resulting similarity values were not based on collection statistics that would have differed for each collection. Indeed, because the similarity values were based solely on document statistics instead of collection statistics, all that was required to combine the results for the subcollections was a straightforward mergesort of the individual results.

The TREC-1 experiments at Virginia Tech (Fox *et al.*, 1993) involved combining the results from several different retrieval runs for a given collection either by simply taking the top *N* documents retrieved for each run, or by modifying the value of *N* for each run, based on the 11-point average precision for that run. We felt these efforts suffered from considering only the rank of a retrieved document and not the actual similarity value itself. In TREC-2, our experiments concentrated on methods of combining runs based on the similarity values of a document to each query for each of the runs. Additionally, combining the similarities at retrieval time had the advantage of extra evidence over combining separate results files, since the similarity of every document for each run was available, instead of just the similarities for the top 1000 documents for each run. Although our results for four of the training collections indicated that the *P*-norm queries performed better than the vector queries, this result was likely specific to the actual queries involved, and not necessarily true in general. This led to a decision to weight each of the separate runs equally, and not favor any individual run or method. In general, it may be desirable or necessary to weight a single run more or less, depending on its overall performance; this could be especially useful in a routing situation.

Six methods of combining the similarity values were tested in our TREC-2 experiments. The rationale behind the CombMIN method, using the minimum of the individual similarity values as the combined similarity value, was to minimize the probability that a nonrelevant document would be highly ranked, whereas the purpose of the CombMAX combination method, using the maximum of the individual similarity values, was to minimize the number of relevant documents being poorly ranked. There is an inherent flaw with both of these methods; namely, they are specialized to handle specific problems without regard to their effect on the other retrieved documents. For example, the CombMIN combination method will promote the type of error that the CombMAX method is designed to minimize, and vice versa. The CombMED combination method is a simplistic approach to handling this, using the median similarity value to avoid both scenarios. What is clearly needed is some method of considering the documents' relative ranks or similarity values, instead of simply attempting to select a single similarity value from a set of runs. To this end, we tried three other methods of combining retrieval methods: CombSUM, the summation of the set of similarity values or, equivalently, the numerical mean of the set of similarity values; CombANZ, the average of the nonzero similarity values, which ignores the negative effect of a given query failing to retrieve a relevant document; and CombMNZ, the sum multiplied by the number of nonzero similarity values, to provide higher weights to documents retrieved by more retrieval methods. Clearly there are more possibilities to consider; the advantages of those chosen are simplicity, in terms of both execution efficiency and implementation, and generality, in terms of not being specific to a given retrieval method.

4. RESULTS

4.1 *Rutgers results*

The official Rutgers results reported to TREC-2 were for the overall performance of each of two treatments for the ad hoc topics, and of one treatment for the routing topics. For those results, we refer the reader to Harman (1994). Here we report on further investigations on the effect of combination of queries, and of data fusion, on performance.

The first investigation in query combination was to see if combining query formulations has a regular, beneficial effect, as hypothesized. This was done by generating the five different search groups for both ad hoc and routing topics, as described in section 3.2, and doing runs on all single query groups (called 1-way from now on), all 2-way combinations of queries, all 3-way combinations of queries, all 4-way combinations of queries, and the combination of all 5 query formulations. For both the ad hoc and routing topics, the average performance increases monotonically as more evidence is added. The increase is sig-

Table 1. Mean 11-point precision for the average of all groups in each level of combination (avg) and for the best-performing combination at each level (best)

	1-way	2-way	3-way	4-way	5-way	fusion
ad hoc (avg)	0.1420	0.1864	0.2103	0.2249	0.2349	0.2042
ad hoc (best)	0.2712	0.3002	0.2959	0.2702	0.2350	0.2042
routing (avg)	0.1831	0.2283	0.2513	0.2672	0.2807	0.2661
routing (best)	0.2931	0.3173	0.3199	0.3069	0.2807	0.2661

nificant between each adjacent level of combination, using the sign test. The data fusion results are significantly better than 1-way combination in the ad hoc case, and better than 1-, 2-, and 3-way combination for routing.

These results are based on the average performance for the query formulations in any one set. When one takes the best performing query, or query combination, for each topic, rather than the average over all combinations within each level of combination, the ranking of level of combination is very much different, with 2-way and 3-way combination being significantly better than 1-way, 4-way, 5-way, and fusion, for both sets of topics. Table 1 shows the mean 11-point precision for both the average and best cases, and Table 2 shows the number of times that any particular level of combination performed better than any other, for only the routing topics (significance figures are based on this table).

Having determined that combination in general is beneficial, we then investigated the effect of weighting the component queries of the five-way combination, for both ad hoc and routing topics. We view this as adaptive combination, amounting to taking account progressively of retrieval performance in modification of query formulation. For the routing task, this is a natural procedure, since there exists the evidence of the queries' performance on the training database, which can be used for modification of the original combined query for searching on the test database. In the ad hoc task, we can imagine that

Table 2. For routing topics, number of times that one level of combination performed better than another, using mean performance for all combinations in a level (average) and performance of the best combination for each topic at each level (best)

	1-way	2-way	3-way	4-way	5-way	fmed5
1-way						
average		3**	2**	1**	2**	8**
best		8.5**	13.5**	22	29	36**
2-way						
average	47**		5.5**	6**	5.5**	13**
best	41.5**		20.5	34*	38**	39**
3-way						
average	48**	45.5**		9**	7**	18*
best	36.5**	29.5		37**	42**	45**
4-way						
average	49**	44**	41**		8**	22.5
best	28	16*	13**		44**	40**
5-way						
average	48**	44.5**	43**	42**		28
best	21	12**	8**	6**		28
fmed5						
average	42**	37**	32*	27.5	22	
best	14**	11**	5**	10**	22	

**significant difference for a single comparison at $p \leq .01$, sign test.
*significant difference for a single comparison at $p \leq .05$, sign test.
Read row with respect to column (e.g., for averaged performance, 5-way performed better than 4-way 42 times, or 4-way performed better than 5-way 8 times).

weighting the results of performance of the queries, given the relevance information, and using the modified combined query on the same database, simulate, as a form of relevance feedback, what might happen in a fully interactive IR system.

The INQUERY retrieval system has an operator, #WSUM, which allows assignment of weights to each component of a query. Our weighting of each component query in the combined query for each topic, in the results reported here, was just the precision for that component query at 100 retrieved documents. In the "official" TREC-2 results we used the average 11-point precision as the weighting factor. This change was made because the former measure is one that can realistically be applied in an operational IR environment. It is justified on two grounds. The first is that we found, through factor analysis, that a single factor accounts for over 90% of the variance in the three TREC-2 effectiveness evaluation measures (average 11-point precision, precision at 100 documents, and precision at the number of relevant documents for the topic). The second is that there was no significant difference in performance between query combinations weighted by 11-point average precision and precision at 100 documents.

For the ad hoc topics, we compared the following conditions in terms of their retrieval performance on the single database:

- unweighted combination of all five queries for each topic (prior);
- the single best-performing query for each topic (btest);
- weighted combination, based on the test results (wtest);
- fusion, using the median rule (fmed5).

Retrieval performance for the first two conditions was available from the official TREC-2 results; for the third, we ran the weighted query against the same database. The "prior" and "fmed5" conditions were our official TREC-2 submissions (referred to in the summary tables of the proceedings (Harman, 1994) as rutcombx and rutfmed, respectively, and in the paper (Belkin *et al.*, 1994) as comb1 and fusion).

For the routing topics, because we had both training and test performance data, the following conditions were compared:

- unweighted combination of all five queries for each topic (prior);
- the single best-performing query for each topic on the training database (btrain);
- the single best-performing query for each topic on the test database (btest);
- weighted combination, based on the training database results (wtrain);
- weighted combination, based on the test database results (wtest);
- weighted combination, mean of training and test weights (wboth); and
- fusion, using the median rule (fmed5).

The "wtrain" results were the official Rutgers TREC-2 routing results, referred to in the summary tables of the proceedings as rutcombx, and in the paper as combx. Both wtest and wboth results were obtained by running the newly weighted queries on the test database.

The results for these comparisons are shown in Table 3, in terms of how often one condition performed better than another. This method of analysis and display provides an immediate means for testing significance of performance differences (the sign test), and also suggests a general metric for comparison of performance of multiple IR systems. Note that in Table 3, the number of cases for the ad hoc topics is 25, whereas for routing it is 50.

Table 3 encapsulates all of the key concepts of the several approaches to combination that the Rutgers project explored. The results for both ad hoc and routing queries are quite similar in pattern. We have two approaches that are a priori and symmetric in their treatment of the query formulations (fmed5 and prior). As expected, the fusion system, using the least information, performs worse. The symmetric query combination, prior, does better, although the difference is not statistically significant. Both of these methods often per-

Table 3. Number of times that one condition performed better than another

	prior	btrain	btest	wtrain	wtest	wboth	fmed5
prior							
ad hoc			8		4**		15
routing		29	21	13.5**	16*	14.5**	28
btrain							
routing	21		13**	14**	14**	12.5**	22.5
btest							
ad hoc	17				9		21**
routing	29	37**		23	20	23	36**
wtrain							
routing	36.5**	36**	27		21.5	18**	40**
wtest							
ad hoc	21**		16				20**
routing	34*	36**	30	28.5		25.5	36.5**
wboth							
routing	35.5**	37.5**	27	32*	24.5		37**
fmed5							
ad hoc	10		4**		5**		
routing	22	27.5	14**	10**	13.5**	13**	

**significant difference for a single comparison at $p \leq .01$, sign test.
*significant difference for a single comparison at $p \leq .05$, sign test.
Read row with respect to column (e.g., wtrain performed better than prior 36.5 times out of 50, for the routing case, or prior better than wtrain 13.5 times).

form better than the best of the individual formulations, although fmed5 does significantly worse than btest in both tasks, whereas there is no significant difference between prior and either of the "best" conditions. Their relations to the other methods of combination are quite similar. The queries that perform best on the routing training set (btrain) do not perform significantly better than any of the combination schemes. But the formulation that performs best on the test set (btest) is significantly better than btrain and fmed5.

Of greater interest are the methods representing adaptive weighting schemes: wtrain, wtest, and wboth. In particular, wtrain, the original adaptive weighting formulation, is significantly better than prior, fmed5, and best1. The weighting based on the test set (wtest) stands also in essentially the same relation to those three other schemes. Finally, the weighting scheme wboth simulates a situation that might arise in updating or tuning a combination rule after two batches of documents have been retrieved. This is accomplished by averaging the weights assigned to each formulation in the training run, with those assigned based on the test run. This scheme shows the same profile as wtrain and wtest when compared with the prior, fmed5, btrain, and btest schemes. It performs significantly better than wtrain, but not significantly better than wtest.

To complete the performance picture, Table 4 shows the 11-point average precision for each treatment, in both ad hoc and routing tasks. From these data, it is clear that there is always a slight advantage to taking account of evidence of performance by combining queries, rather than by choosing to use the best query. This supports the data from Table 3, where wtrain, wtest, and wboth have a significant advantage over btrain, and a slight, nonsignificant, but consistent numerical advantage over btest.

Table 4. Mean 11-point precision for each query method

	prior	btrain	btest	wtrain	wtest	wboth	fmed5
ad hoc	0.2350		0.2712		0.2819		0.2042
routing	0.2807	0.2721	0.2931	0.3012	0.3090	0.3068	0.2661

Table 5. Comparison of the five individual runs and the combination runs on the training data (average precision, topics 51-100)

Run	AP-1	WSJ-1	AP-2	WSJ-2
SV	0.2387	0.2203	0.2543	0.1503
LV	0.2435	0.2414	0.2664	0.1633
Pn1.0	0.2810	0.2941	0.3004	0.2206
Pn1.5	0.3122	0.3199	0.3332	0.2327
Pn2.0	0.3027	0.3217	0.3300	0.2325
CombMIN	0.2863	0.1924	0.3047	0.1308
CombMAX	0.2856	0.3205	0.3337	0.2343
CombMED	0.2943	0.3204	0.3335	0.2328
CombSUM	0.3493	0.3605	0.3748	0.2752
CombANZ	0.3493	0.3367	0.3748	0.2465
CombMNZ	0.3059	0.3368	0.3516	0.2467

4.2 *VT results*

The five VT individual runs were performed and evaluated for each of the two AP and WSJ training collections on topics 51 to 100. The results for these experiments are given in the top half of Table 5. In general, the *P*-norm queries performed better than the vector queries. The most effective *P*-value, however, differed between the collections: The AP runs performed better for the Pn1.5 run, while the Pn2.0 run performed better or approximately equal to the Pn1.5 run for the WSJ collections.

The six combination methods described in Section 3.3 were evaluated against the AP and WSJ training collections for topics 51 through 100, combining the similarity values of each of the five individual runs. The results are shown in the bottom half of Table 5 below the results of each of the individual runs. Note that while the CombMAX runs performed well with respect to most of the individual runs, in most cases they did not do as well as the best of the individual runs. The CombMIN runs performed similarly for the AP collection, but performed worse than every individual run for the WSJ collection.

The CombANZ runs and the CombMNZ runs both performed better than the best of the individual runs, with one exception: The CombMNZ runs perform only slightly better than the CombANZ runs for the WSJ collection, and significantly worse for the AP collection. The CombSUM runs consistently performed better than any individual runs, and better or at worst equal to any of the combination runs.

The CombSUM retrieval run was performed for each of the nine collections on the two training CD-ROMs. Overall, the CombSUM results showed a 16% improvement over the best overall individual run (Pn2.0). Breaking the analysis down to a per topic basis in Table 6, it can be seen that the CombSUM method performs significantly better than the

Table 6. Number of times that one run performed better than another, training Topics 51-100

	Pn2.0	Max.	CombSUM
Pn2.0		20	12.5**
Max.	30		28
CombSUM	37.5**	22	

**significant difference for a single comparison at $p \leq .01$, sign test.
*significant difference for a single comparison at $p \leq .05$, sign test.
Read row with respect to column (e.g., CombSUM performed better than Pn2.0 37.5 times out of 50, or Pn2.0 better than CombSUM 12.5 times).

Table 7. Comparison of individual runs and CombSUM run for the TREC-2 routing results (average precision, topics 51–100)

Run	AP-3	PAT-3	SJM-3	ZF-3	Disk 3
SV	0.1347	0.0189	0.1139	0.0593	0.0589
LV	0.1189	0.0156	0.1056	0.0587	0.0494
Pn1.0	0.2519	0.0257	0.2128	0.1141	0.2039
Pn1.5	0.2869	0.0239	0.2411	0.1189	0.2279
Pn2.0	0.2852	0.0221	0.2390	0.1303	0.2225
CombSUM	0.3196	0.0260	0.2696	0.1304	0.2681
Chg/Best[a]	11.4%	1.2%	11.8%	0.07%	17.6%

[a]Chg/Best is the percent improvement of the CombSUM method over the results for the best run for each collection.

Pn2.0 run. However, comparing the CombSUM results with the best individual run's results (labeled the 'Max.' precision in Table 6) for each query indicates that the CombSUM results may not be significantly better than choosing the best possible method for each given query.

Although combining all five runs produced an overall improvement in retrieval effectiveness over each of the runs, the same does not always hold true when combining only two or three runs. Each of the 10 CombSUM combinations of two runs was performed for both of the AP test collections, as well as a run combining all three of the *P*-norm runs. Most of the combinations of two runs performed worse than the better of the two runs, while performing better than the poorer of the two runs. One notable exception to this is the combination of the two vector runs, which performed noticeably poorer than either of the two runs.

The CombSUM combination method was used for both our official TREC-2 routing and ad hoc results; the other combination methods were not evaluated against this collection or for topics 101 to 150. The exact queries for topics 51 to 100 used for testing our combination method were used for the routing queries against the new collections on disk 3. The results obtained from performing the CombSUM retrieval runs for each of the four collections as well as the merged disk 3 results are shown in Table 7. The ad hoc queries for topics 101 to 150 were evaluated in the same manner, and are reported in Table 8. The two CombSUM entries in the last column of these two tables are the official TREC-2 results.

Overall, in the routing case the CombSUM results showed a 17% improvement over the best overall individual run (Pn1.5), and a 24% improvement over the Pn2.0 run in the ad hoc case. Breaking the ad hoc analysis down to a per-topic basis in Table 9, it can be seen that the CombSUM method performs significantly better than the Pn2.0 run, performing better for 46 out of the 50 topics. However, the results were less significant when comparing the CombSUM results with the best individual run for each query.

Table 8. Comparison of individual runs and CombSUM run for the TREC-2 ad hoc results (average precision, topics 101–150)

Run	AP-1	DOE-1	FR-1	WSJ-1	ZF-1	AP-2	FR-2	WSJ-2	ZF-2	Disks 1 & 2
SV	.3237	.0949	.0630	.2740	.0936	.3068	.0650	.2259	.1166	.2035
LV	.3326	.1018	.2848	.2981	.0602	.2483	.1045	.2159	.0697	.0997
Pn1.0	.3340	.0831	.1777	.3153	.1292	.3133	.1927	.2838	.1722	.2205
Pn1.5	.3682	.0814	.1874	.3332	.1430	.3438	.1982	.2941	.1964	.2543
Pn2.0	.3647	.0750	.1761	.3290	.1307	.3419	.1995	.2828	.2018	.2573
CombSUM	.4153	.1038	.2133	.3778	.1657	.3959	.2000	.3561	.2200	.3206
Chg/Best[a]	12.8%	9.4%	13.8%	13.4%	15.9%	15.1%	0.2%	21.0%	9%	24.6%

[a]Chg/Best is the percent improvement of the CombSUM method over the results for the best run for each collection.

Table 9. Number of times that one run performed better than another, ad hoc topics 101–150

	Pn2.0	Best	CombSUM
Pn2.0		14**	4**
Best	36**		17*
CombSUM	46**	33*	

**significant difference for a single comparison at $p \leq .01$, sign test.
*significant difference for a single comparison at $p \leq .05$, sign test.
Read row with respect to column (e.g., CombSUM performed better than Pn2.0 46 times out of 50, or Pn2.0 better than CombSUM 4 times).

4.3 *Summary of results from Rutgers and VT*

As it customary, we begin this section with a general disclaimer. In this case, we need to point out that all of our results were obtained with very specific kinds of query formulation techniques and very special kinds of queries, and that all of our results were obtained within very special retrieval contexts. It is certainly possible that these circumstances strongly affected our results, so that we cannot make widely general claims for them. On the other hand, because the two projects themselves used quite different techniques, it seems that there is a *prima facie* case for some level of generality of results.

The results of both the Rutgers and VT projects for both ad hoc and routing topics seem to show clearly that, in general, the more evidence one has, and uses, in the form of different query formulations, the better the IR performance is going to be. Every instance of combination resulted in improvement of performance, on average, in both projects.

Tables 3, 6, and 9 show that, for both projects, query combination often performs better than choosing the best individual query formulation. Although choosing the "best" query sometimes results in significant performance differences from combined queries, in any run there are always instances (sometimes significant) of combined queries performing better than the best. And on average, combination does better, in both projects.

From the results of Tables 3 and 4, we can see that taking advantage of what one learns about query performance from one iteration does not help a lot, after the first iteration but, on the other hand, it does not hurt, either. This suggests to us that continual modification and reweighting of the multiple query formulations in a combined query is likely to be useful in the general routing environment. But even doing it once, given the initial evidence, seems to help. This also suggests that continuing to add new query formulations to a combined query will likely help performance on subsequent runs.

Having said all this, it is worth considering the results of Tables 1 and 2, which showed that picking the best 2-way or 3-way combination of query formulations was significantly better than using 4-way or 5-way combinations. On the face of it, this runs counter to the general result of "the more, the better." However, it is possible that this result is an artifact of the Rutgers data. For both 2-way and 3-way combinations, it was possible to choose the best from ten different combinations. Because there were only five different query formulations for each topic, there were smaller pools from which to choose, for both single-query formulations, and for the 4-way and 5-way combinations. This issue needs further investigation.

5. COMBINED RESULTS

5.1 *Schemes used to combine results*

Given the interesting and generally positive results of the two separate studies involving combination of evidence or data fusion, it is appropriate to ask whether those results exhaust the possibilities for improvement. Specifically, the two sets of results differ in so many ways (different formulations of queries, different search engines, different logical rules of combination) that they form a realistic case for the notion of data fusion on two systems whose internal workings are not available to the fusion algorithm. To test these ideas we have done two sets of experiments.

In one set of experiments, which are designated as (j*), two results, one generated by the prior or wtrain method of Rutgers (prior for ad hoc, wtrain for routing) and the other by the CombSUM method of VT, were combined using two different data fusion logics (defined below), and a sum of scores (or query combination) rule. The rationale for doing combination is that the two institutions, regarded as "systems," have produced their best ranked lists, and we want to see whether anything is to be gained by combining them.

However, there is not much range of variation possible when there are only two schemes to combine. So we also asked whether combining two of the VT schemes with one Rutgers scheme could produce better results. In the set of experiments called (3*) three sets of results, prior or wtrain, lv, and pl.5, were directly submitted to one data fusion logic and to one sum of scores rule, to see the effect of replacing one of the combined sets by more fundamental constituents.

In each analysis we compared the results of combination with the inputs to that combination. We also compared the results of the several combination schemes with each other. Finally, it is of considerable interest to compare the results of combination with the "perfect foresight" scheme, which uses, for every query, the specific method that turns out to have the best performance for that query. This tests the (seemingly self-evident) notion that since there will be different systems that work better for different queries, the ideal approach would be to use, for every query, the system that "suits it best."

With reference to the query combination and data fusion logics discussed in Section 2, the rules of combination used in this part of our work are defined as follows:

jmax	MAXrule(CombSUM, prior/wtrain)
jmin	MINrule(CombSUM, prior/wtrain)
jsum	Normalized sum of scores rule(CombSUM, prior/wtrain)
3med	MEDianRule(prior/wtrain, lv, pl.5)
3sum	Normalized sum of scores rule(prior/wtrain, lv, pl.5).

5.2 *Empirical results of combined fusion*

Table 10 shows the results of combination of evidence (all five combination rules are referred to in that table as "fusion") from the Rutgers and VT projects. There is much to be found in this table. In the ad hoc cases, the only input that is consistently roughly as

Table 10. Fusion results versus inputs to the fusion

	Ad hoc (25 topics)				Routing (50 topics)			
	CombSUM	prior	lv	pl.5	CombSUM	wtrain	lv	pl.5
jmin	13	22*	18*	19*	35*	28	49*	41*
jmax	15	21*	18*	17	34*	29	49*	39*
jsum	15	19*	19*	21*	36.5*	31	48*	40*
3med	14	21*	21*	20*	28	23	49*	35*
3sum	17	20*	21*	22*	34*	27	47.5*	37.5*

Also significant are:
for Ad hoc jmin(18>)jmax,
for Routing jmin(33>)jmax; jmin(38>)3med; jmax(39>)3med); jmax(32>)3sum; jsum(36>)3med; jsum(42>)3sum.
Read row with respect to column (e.g., jmin had higher average precision than CombSUM 13 times). Rows are labelled by the fusion methods. j- entries combine only two inputs. Scores that would be significant at a 95% confidence level if they had been observed in a single comparison are marked *.

good as the combination schemes is CombSUM. Other than that, the combination schemes "beat" their inputs. In the routing case the only input that is consistently roughly as good as the combination schemes is wtrain. Interestingly, this is an adaptively tuned scheme, for which weights were chosen on the basis of a training set.

As shown in the notes to Table 10, direct comparison of the fusion methods with each other shows a few results significant at the level chosen. For example, on routing queries, jmin beat 3med in 38 (of the 50) cases. This is expressed concisely as jmin(38>)3med. There are two interesting results to be seen here. First, applying rank-based fusion schemes to the lower level systems themselves (the 3* schemes) is less effective than applying them to the pre-processed results given by CombSUM. Second, jmin (the logical OR) scheme works better than jmax for both sets of topics. This tells us something about the regime in which both input systems are operating. Most likely, it means that both systems are retrieving different sets of relevant documents, each with high precision.

In Table 11, we consider the difference in performance between combination of sources of evidence, and choosing the best-performing individual method. As suggested by our earlier observations, the better of the two combined methods is more effective than the best of the two VT schemes and the Rutgers combined method, and is, in fact, more effective than any of the fusion schemes. Hence, perfect foresight would be more effective than any specific fusion scheme we have considered. However, it is important to note that many of the entries in this table are greater than zero. If it were indeed correct that the best overall rule is always to apply that engine or method best for a specific topic, the fusion schemes would never do better than the "best," and every entry in this table would be 0. This shows that data fusion or query combination schemes do, in general, add to the power of information retrieval when more than one system is available. Note that, as might be expected, the fusion schemes beat the best2 in a larger fraction of the cases for ad hoc topics than for routing topics, since for the latter the schemes used here as input have been tuned to improved their performance.

6. CONCLUSIONS

In general, we conclude that our initial research questions with respect to query combination have been positively answered. That is, if one has available several different representations of a single information problem, then it makes sense to use all of them, in combination, to improve retrieval performance, rather than try to identify and use only the best one. In addition, it is reasonably clear that progressive and continuous combination of query formulations leads to continuing and progressive improvement of performance. This may extend to progressive modification of query formulations in the routing situation, for instance, on the basis of each iteration of retrieval.

Table 11. Fusion results vs "perfect foresight" schemes

	Ad hoc (25 topics)		Routing (50 topics)	
	best2[a]	best3[b]	best2	best3
jmin	10	11	14*	20
jmax	11	9	16*	21
jsum	9	12	17.5*	21
best3	9.5	12.5	15.5*	25
3med	10	12	9*	13*
3sum	12	13	14*	17*

[a]best2 represents using the best of the two input systems for each query.
[b]best3 represents using the best of the 3 input schemes when the lower level VT inputs were used.
Scores that would be significant at a 95% confidence level if they had been observed in a single comparison are marked *.

It is clear that combination of evidence based on different query formulations is generalizable to the two separate cases of combination from within the same system (query combination), and combination of evidence from different systems (data fusion). And although the evidence comparing the performance of the "best" single query formulation to the combination of different query formulations for any given topic is equivocal, we especially note that both projects found that particular combinations of evidence often performed better than the "best," sometimes significantly so. This suggests that, in cases where knowledge of relative performance of query formulations is unavailable, it is always a good idea to combine the formulations. Furthermore, even in cases where such knowledge is available, it is almost as effective to use it for weighted query combination as to use it to choose the best single query.

The results also show that, in cases where systems are commensurable, combination using similarity scores directly is better than combination using only evidence of the ranks. However, in combining results from different systems, there is no obvious advantage to query combination over data fusion. Thus, for operational settings in which there are multiple systems with truly incompatible scores, a data fusion method that works with the ranked outputs, rather than the scores directly, is the proper method for combination.

The results from the VT experiments for combination of results from different databases are also encouraging. But whether such combination is possible among systems that have different methods for computing similarity scores is still an open issue. In the context of the digital library, distributed over different databases and different retrieval systems, this is a significant problem, which needs to be addressed further.

Overall, we find strong support for adaptive weighting in query combination. This is applicable to both routing, as shown directly here, and to relevance feedback, which was simulated in the Rutgers ad hoc experiments. We also find strong support for enlarging the set of query representations and, based on the results of combination of Rutgers and VT evidence, for the use of different retrieval systems. This success raises many interesting possibilities. For example, one might systematically explore the n-way query combinations to see how they compare to the adaptive weighting scheme. Or one might apply the notion of adaptive weighting to the best of the n-way query combinations. And both of these approaches could be compounded by their use in multiple retrieval systems. The possibilities for combining these three concepts explode (of course) combinatorially. It appears to us that an important aspect of the results of these two projects is that they indicate directions for further research that can help both to understand the reasons why such query combination works, and to choose specific means for combination of such evidence, which will lead to optimal information retrieval performance.

Acknowledgements—The Rutgers researchers wish to thank the 75 searchers who so generously donated their time and effort to this project. Without them, this research could not have been done. They also wish to thank Audrey Gorman and Kathy Mrowka, Dong Li, Colleen Cool, and especially Richard Quatrain, for all of their hard work. They owe special thanks to W. Bruce Croft and James P. Callan, not only for permission to use the INQUERY system for this investigation, but also for the unstinting support they gave in using it. The Rutgers research was performed with partial funding from a TREC support grant from the ARPA. The Virginia Tech researchers wish to thank Russell Modlin, M. Prabhakar Koushik, and Durgesh Rao for their collaboration during TREC-1. The Virginia Tech research was supported in part by PRC Incorporated, and by a TREC support grant from the ARPA.

REFERENCES

Belkin, N.J., Cool, C., Croft, W.B., & Callan, J.P. (1993). The effect of multiple query representations on information retrieval performance. In *Proceedings of the 16th International Conference on Research and Development in Information Retrieval (SIGIR '93)*, Pittsburgh, 1993 (pp. 339–346). New York, ACM.

Belkin, N.J., & Croft, W.B. (1992). Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM, 35*(12), 29–38.

Belkin, N.J., Kantor, P., Cool, C., & Quatrain, R. (1994). Combining evidence for information retrieval. In D. Harman (Ed.), *TREC-2, Proceedings of the Second Text REtrieval Conference* (pp. 35–44). Washington, D.C., GPO.

Belkin, N.J., Oddy, R.N., & Brooks, H.M. (1982). ASK for information retrieval. *Journal of Documentation, 38*, 61–71, 145–164.

Buckley, C. (1985). Implementation of the SMART information retrieval system. Technical Report 85-686. Ithaca, NY: Cornell University, Department of Computer Science.

Callan, J.P., Croft, W.B., & Harding, S.M. (1992). The INQUERY retrieval system. In *DEXA 3, Proceedings of the Third International Conference on Database and Expert System Applications* (pp. 78-83). Berlin: Springer-Verlag.

Cherikh, M. (1989). Optimal decision and detection in the decentralized case. Ph.D. dissertation, Cleveland, OH: Case Western Reserve University.

Fox, E.A. (1983). Extending the Boolean and vector space models of information retrieval with *P*-norm queries and multiple concept types. Ph.D. dissertation, Ithaca, NY: Cornell University Department of Computer Science.

Fox, E.A., & Shaw, J.A. (1994). Combination of multiple searches. In D. Harman (Ed.), *TREC-2, Proceedings of the Second Text REtrieval Conference* (pp. 243-352). Washington, D.C., GPO.

Fox, E.A. *et al.* (1993). Combining evidence from multiple searches. In D.K. Harman (Ed.), *TREC-1, Proceedings of the First Text REtrieval Conference* (pp. 319-328). Washington, D.C., GPO.

Harman, D. (Ed.) (1994). *TREC-2, Proceedings of the Second Text REtrieval Conference.* Washington, DC: GPO.

Kantor, P. (1993). Vector space models of data combination in information retrieval. Technical Report APlab/TR-93-3. New Brunswick, NJ: Rutgers University, School of Communication, Information & Library Studies.

Kantor, P. (in preparation). Pattern recognition techniques in data fusion for information retrieval. Technical Report APLab. New Brunswick, NJ: Rutgers University, School of Communication, Information & Library Studies.

Katzer, J., McGill, M.J., Tessier, J.A., Frakes, W., & Dasgupta, P. (1982). A study of the overlap among document representations. *Information Technology: Research and Development, 1,* 261-274.

McGill, M., Koll, M., & Norreault, T. (1979). An evaluation of factors affecting document ranking by information retrieval systems. Syracuse: Syracuse University School of Information Studies.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference.* Los Altos, CA: Morgan Kaufmann.

van Rijsbergen, C.J. (1986). A new theoretical framework of information retrieval. In *Proceedings of the 1986 International Conference on Research and Development in Information Retrieval (SIGIR '86),* Pisa, 1986 (pp. 194-200). New York, ACM.

Saracevic, T., & Kantor, P. (1988) A study of information seeking and retrieving. III. Searchers, searches, overlap. *Journal of the ASIS, 39,* 197-216.

Taylor, R.S. (1968). Question negotiation and information seeking in libraries. *College and Research Libraries, 29,* 178-194.

Turtle, H.R., & Croft, W.B. (1991a). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems, 9,* 187-222.

Turtle, H.R., & Croft, W.B. (1991b). Efficient probabilistic inference for text retrieval. In *RIAO 3,* Conference Proceedings (pp. 644-661). Barcelona.