Method Combination For Document Filtering

David A. Hull* **Jan O. Pedersen†** **Hinrich Schütze†**

*Rank Xerox Research Center
6 Chemin de Maupertuis
38240 Meylan, France
hull@xerox.fr

†Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, USA
{schuetze,pedersen}@parc.xerox.com

http://www.parc.xerox.com/istl/groups/qca

Abstract

There is strong empirical and theoretic evidence that combination of retrieval methods can improve performance. In this paper, we systematically compare combination strategies in the context of document filtering, using queries from the Tipster reference corpus. We find that simple averaging strategies do indeed improve performance, but that direct averaging of probability estimates is not the correct approach. Instead, the probability estimates must be renormalized using logistic regression on the known relevance judgements. We examine more complex combination strategies but find them less successful due to the high correlations among our filtering methods which are optimized over the same training data and employ similar document representations.

1 Introduction

A text filtering system monitors an incoming document stream and selects documents identified as relevant to one or more of its query profiles. If profile interactions are ignored, this reduces to a number of independent binary decisions to accept or reject each document with respect to a given profile. This description has been adopted as the definition of the filtering track at the Text Retrieval Conference (TREC-4) [Lewis, 1995]. In the TREC context, filtering is distinguished from document routing because it is assumed that documents arrive over time and are not retained, so ranking of the full document set is impossible. These filtering conditions prevail in many practical information access settings [Lewis, 1995].

Text filtering can be approached as a basic text matching problem or an interactive or automatic learning process depending on the degree to which the user is willing or able to provide relevance judgements about filtered documents. In the TREC experiments, a substantial set of labeled documents is provided in advance to train the system, and hence text filtering can be treated as a non-interactive machine learning problem. While the basic filtering task requires only a binary decision about document relevance, we attack a more general problem, that of estimating the probability of relevance. If this probability were known then an optimal decision could be made that maximizes linear utility, a typical performance measure for the document filtering problem [Lewis, 1995]. Of course, the probability can only be estimated based on predictors measurable on incoming documents.

Our approach is both to develop statistical classifiers that produce accurate estimates of the probability of relevance and to seek methods that combine these estimates to improve performance. Combination of evidence has been pursued in other information retrieval (IR) contexts, for example for the ad-hoc search problem (i.e. information retrieval based on a simple query profile [Belkin et al., 1995]) and document routing. However filtering is particularly well suited to probabilistic learning models due to the agreement between the cost functions employed by these classifiers and the evaluation criterion (linear utility).

Combination is well-motivated from both an empirical and a theoretical perspective. Saracevic and Kantor [Saracevic & Kantor, 1996] conducted an extensive user study indicating that query formulations designed by different users for the same information need can retrieve document sets that barely overlap at all. However, the likelihood of relevance tended to increase monotonically with the number of times the document was retrieved, suggesting that combining different query formulations has the potential to improve performance. The theoretical motivation stems from the fact that the accuracy of probabilistic measures of relevance increases and the error variance of the model decreases with each additional independent source of evidence.

In a previous paper [Schütze et al., 1995], we explored a number of statistical approaches to the document routing problem. We demonstrated that sophisticated learning algorithms applied to a reduced document representation can significantly improve performance over traditional relevance feedback techniques. In this paper, we apply these classifiers to the filtering problem and investigate combination methods. We show that obtaining accurate probability estimates through method combination is more difficult than improving ranking strategies. We develop a combination strategy that improves filtering performance at some probability thresholds.

We go on to experiment with a number of complex combination strategies, but find that they are not as successful as simple averaging strategies. In the analysis of these experimental results, we learn that the potential for improvement through method combination is limited because many of our classifiers are strongly correlated, despite the fact that they use different document representations and optimization strategies. We conclude that an important condition for the success of method combination is training individual classifiers more independently, through the use of non-overlapping feature sets or partitions of the training data.

This paper is organized as follows. Section 2 discusses related work in the fields of information retrieval and machine learning. Section 3 introduces our approach to filtering. Sections 4 and 5 describe our experiments with simple and complex combination strategies. In Section 6, we state our conclusions.

2 Method Combination Strategies

A method combination strategy combines the output of a number of different classifiers to form an aggregate estimate of relevance. In general, there is no guarantee that a combination strategy will improve performance over individual methods. For example, if an accurate classifier is combined with one that generates random classifications, then no improvement is realized. See [Littlestone & Warmuth, 1992, Haussler et al., 1994] for theoretic bounds on the maximal loss in performance due to combination. Despite this worst case scenario, there is much empirical evidence that combination is beneficial in information retrieval [Belkin et al., 1995] and machine learning. For example, if the optimal classifier is unknown, combination can be advantageous even if the combined results are worse than the best individual classifier. Our experimental results will confirm this hypothesis. In the remainder of this section, we discuss approaches to combination that have been examined in previous information retrieval and machine learning experiments.

2.1 Method Combination in Information Retrieval

Combination of evidence is inherent to all approaches to information retrieval. At the most basic level, evidence consists of query term occurrence patterns which are merged together using additive (vector space and probabilistic) or logical (boolean) models. The combination strategies seen in the literature range from simple unweighted boolean retrieval to sophisticated methods such as Bayesian inference networks [Turtle & Croft, 1991] and logistic regression [Fuhr & Pfeifer, 1994, Cooper et al., 1994]. More recent interest has focused on the merging of higher level structures, such as multiple query or document representations (e.g. [Bartell et al., 1994, Belkin et al., 1995]).

Perhaps the most common approach to method combination in information retrieval has been to test a series of carefully selected weighting functions [Belkin et al., 1995, Salton et al., 1983, Fox & Shaw, 1994, Belkin et al., 1993, Turtle & Croft, 1991, Schütze & Pedersen, 1994, Lee, 1995]. Many of these functions have parameters which are optimized by manual search over the parameter space using performance on IR test collections as the criterion to select the appropriate weights. A more principled approach is to use automatic methods to select the appropriate weights, such as logistic regression [Fuhr & Pfeifer, 1994, Cooper et al., 1994], least squares [Fuhr, 1989], neural networks [Kwok, 1995], or numerical optimization [Bartell et al., 1994]. However, a key problem with many of these techniques is the mismatch between the cost function used for weight optimization and the evaluation criterion. As Lewis [Lewis, 1995] points out, it is important to optimize and evaluate using the same measures since this better guarantees that the automatic classifier is seeking an optimal value of interest.

Bartell et al. [Bartell et al., 1994] recognize this problem and select a rank correlation measure for training that closely matches the behavior of average precision at fixed recall. Even though the match is not exact, they are rewarded with dramatic improvements in performance. The other studies referred to above also find significant improvements in performance due to combination, although the degree of improvement varies, depending on the number and type of retrieval methods being combined. Previous work has focussed on the ad hoc search problem. In this paper, we present experiments with combination for a different information retrieval task: document filtering [Lewis, 1995]. We attempt to accurately estimate the probability of relevance of incoming documents which can be used to optimize performance as measured by linear utility. We then merge these estimates using a variety of combination strategies.

2.2 Method Combination in Machine Learning

The most basic combination strategy is averaging. Tumer and Ghosh [Tumer & Ghosh, 1995] show that under certain assumptions (e.g., similar variances across classifiers) the expected error of the average estimate is $\frac{1}{N}$ of the error of a single classifier if the N classifiers are independent. This error reduction decreases towards zero the more dependent the classifiers are. A similar result can be found in [Ali & Pazzani, 1995]: there is a linear relationship between the degree of error reduction and the degree to which patterns of errors made by individual models are uncorrelated. The greatest error reduction is achieved with negative correlation.

One approach that explicitly deals with inter-method correlation is Perrone's Generalized Ensemble Method (GEM), a linear combination method [Perrone & Cooper, 1995]. A method weight is smaller the more correlated the method is with other methods. However, in an exploratory experiment we did not find an improvement using GEM, possibly because our correlation estimates were not accurate enough. Another problem is that the cost function of GEM is mean squared error, which does not match our evaluation criterion.

There are many strategies for combining essentially identical classifiers whose decisions differ because they have different initializations or are trained on different parts of the training set. A simple scheme is majority voting: Each classifier votes for or against class membership and the majority of votes decides (see e.g. [Breiman, 1994]). In minimal cross validation, k different partitions of the training set are created and for each partition a classifier is trained on one part of the partition and evaluated on another. Then, the classifier with the best performance in the evaluation is chosen. *Stacking* is a generalization of cross validation in which the estimates of all individual classifiers are used. A second learning algorithm is trained to integrate them into a single classification decision [Wolpert, 1992, Breiman, 1993]. *Boosting* is another multi-level training algorithm in which two lower-level classifiers are consulted first. A third classifier makes the decision if they disagree [Drucker et al., 1993b, Drucker et al., 1993a].

In this paper, we are instead interested in the combination of different types of classifiers rather than variations of the same classifier, hence the strategies outlined above are inappropriate. For example, majority voting gives equal weight to each classifier whereas a better strategy is to give more influence to reliable classifiers and less to unreliable ones. A winner-take-all approach such as minimal cross validation relies on only one classifier instead of integrating information from all classifiers. Finally, multi-level schemes such as boosting have the disadvantage that the training of individual classifiers and their combination are interwoven. In a practical IR setting, the individual classifiers are often black boxes whose fixed results are combined in a second independent step of relevance assessment.

A combination method that can combine different classifiers is the Adaptive Mixtures of Experts (AME) neural network architecture. AME consists of a set of neural networks and a gating network which decides on a per-pattern basis how to combine estimates from the individual networks. The parameters of the gating network are trained with gradient descent [Jacobs et al., 1991] or the expectation-maximization (EM) algorithm [Lei Xu & Hinton, 1995]. As with boosting, this approach often cannot be applied to information retrieval since it requires joint training of the individual classifiers and the gating network. Tresp and Taniguchi also change the weighting of classifiers from pattern to pattern, making it dependent either on the variance of the input or on how well represented the input pattern was in the training set (that is, classifiers with lower variance for an input pattern and good coverage of the input pattern's local space are weighted higher) [Tresp & Taniguchi, 1995].

Such non-constant weighting functions can be a powerful combination technique. Consider a user interested in "Alternative Cancer Therapies". If this phrase occurs in a document, then it is most likely relevant. However, if it does not occur the document might still be relevant if a sufficient degree of alternative evidence were present. A term-based classifier could recognize the first case and a topic-based classifier the second case [Schütze et al., 1995]. The gating network in AME could then learn to defer the decision to the term-based classifier if the phrase were present and to the topic-based classifier otherwise. Such an input-based combination strategy is not possible with a constant weighting function. However, since AME requires joint training of individual classifiers and the weighting function and since it is doubtful that most filtering queries justify the complex modularized architecture of AME, we will only explore constant weighting functions in this paper. See [Jacobs, 1995] for a review of different types of non-constant combination. Others also find that averaging is the best strategy rather than the more complicated strategies suggested in theory [Hampshire & Waibel, 1990, J. Ghosh & Beck, 1992].

3 Document Filtering

Document filtering requires a decision to accept or reject documents as they appear one at a time *independently* of previously or subsequently examined documents. To make these judgements, the system must produce some measure of value, or likelihood of relevance, for each individual document. Our approach is to treat document filtering as a machine learning problem. A training set of evaluated documents is used to construct a classification rule to estimate the probability of relevance of incoming documents. The training process can be divided into four steps: (1) a preliminary filtering step to reduce size of training set to 2000 documents (if necessary), (2) calculation of several distinct document representations, (3) application of machine learning techniques, and (4) conversion of scores from (3) into probability estimates (if necessary). The testing process for incoming documents works in a similar fashion. We have examined four different learning algorithms: Rocchio query expansion, nearest neighbors (NN), linear discriminant analysis (LDA), and a neural network (Net) fitting a logistic model. For more details on this work, see [Schütze et al., 1995, Hearst et al., 1996]. In these experiments, Rocchio expansion and nearest neighbors use the full term-based document representation, LDA uses local LSI factors [Deerwester et al., 1990, Hull, 1994][1], and the neural network uses a combination of local LSI and selected term features.

Since documents are not ranked, filtering algorithms cannot be analyzed using traditional IR performance measures, such as precision averaged at fixed recall levels or precision at a constant number of documents retrieved. Instead, it has been suggested that filtering be evaluated using utility-based measurements [Lewis, 1996]. For the filtering experiments at the recent TREC-4 conference, evaluation was based on a linear measure of the form: $u = u_r * R - u_n * N$, where u_r is the value (utility) of retrieving a relevant document, u_n is the cost of retrieving a document that is not relevant ($u_r, u_n > 0$), and R and N are the number of relevant and non-relevant documents respectively. The goal of the system is to maximize the utility u of the retrieved set.

This criterion reduces to a simple decision rule based on the estimated probability of relevance. Document d_i should be accepted if doing so has positive expected utility, i.e. if $E(u(d_i)) > 0$:

$$\begin{aligned} E(u(d_i)) &= E(u_r * I[d_i \in \text{rel}] - u_n * I[d_i \in \text{non-rel}]) \\ &= u_r * p_i - u_n * (1 - p_i) = (u_r + u_n)p_i - u_n \end{aligned}$$

where I is an indicator function and p_i is the probability that d_i is relevant. Therefore, $E(u(d_i)) > 0 \implies p_i > u_n/(u_r + u_n)$, or d_i should be accepted if its probability of relevance exceeds this threshold. Following the methodology developed for TREC-4, we choose to measure performance for three different utility functions: (u_r, u_n) = (1,3), (1,1), and (3,1), which are optimized by selecting documents with a probability of relevance greater than p = 0.75, 0.5, and 0.25, respectively.

As in our previous work, we use the TIPSTER corpus and topics 51-150 to measure performance (this corresponds to TREC-2 and TREC-3 routing experiments)[2]. Performance is evaluated using the pooled relevance judgments from all sites. An alternative technique suggested for TREC-4 uses sampling to obtain unbiased utility estimates [Lewis, 1996]. We decided to accept some bias in our results (all unevaluated documents are assumed not relevant) in order to have a lower variance (more documents evaluated for pooling than by sampling) and to save the work of sampling. The corpus is processed using the TDB system developed at Xerox PARC [Cutting et al., 1991]. Document features

[1] LSI factors are orthogonal linear combinations of variables derived from a singular value decomposition of the term-document cooccurrence matrix.

[2] We do not use the TREC-4 topics because of an overlap between training and test sets in the experimental set-up.

include all words and adjacent word pairs that occur more than five times in the corpus, and each document is partitioned into coherent subtexts using the TextTiling algorithm [Hearst, 1994]. A preliminary filtering step selects the 2000 most similar documents to the Rocchio expanded query as the training set, where each document is represented by its highest scoring subsection. All test documents are also passed through this filter.

threshold	Roc	NN	LDA	Net
p75	0.07	6.89	5.43	9.65
p50	2.2	21.9	27.1	27.9
p25	68	159	184	190

Table 1: Average utility of Rocchio expansion (Roc), Nearest Neighbor (NN), Linear Discriminant Analysis (LDA), and the Neural Network (Net), at three probability thresholds.

threshold	Roc	NN	LDA	Net
p75	2.12	2.76	2.37	2.75
p50	1.76	2.62	2.68	2.93
p25	1.45	2.40	2.86	3.30

Table 2: Average per-query rank of utility scores presented in Table 1.

Table 1 presents the average utility score over the queries for each of four learning algorithms. Table 2 presents the same results ranked on a query by query basis and then averaged over queries (i.e. for each query we rank the utility scores of the four methods and then the ranks of each method are averaged over the 100 queries). Neither of these approaches is ideal for averaging over a large query sample. Queries which return lots of documents and thus have high variability in their scores are much more important for the average score than queries with few documents returned. While average rank normalizes for the unequal variability between queries, its value depends on which methods are being compared. Despite these drawbacks, it is hoped that examining both of these measures will provide a reasonable picture of overall performance. For measuring statistical significance, the Friedman Test [Hull, 1993] (for three or more methods) and the Sign Test (for two methods) are considered most appropriate. All future references to significance refer to these tests, and correspond to a p-value less than 0.01[3].

For this experimental setting, the neural network is consistently the most successful learning algorithm for filtering, producing the best results at all probability thresholds. We note that nearest neighbors performs about as well as the network for the threshold p = 0.75, but its relative performance decreases rapidly for lower thresholds. It is not surprising that a technique which uses the relevance of a document's nearest neighbors works best at a high probability threshold. While LDA performs almost as well as the network in terms of average utility for p = 0.50 and p = 0.25, its average rank is quite a bit lower. This difference becomes statistically significant for p = 0.25.

[3]We should mention that the Friedman Test assumes independence between the methods being compared, which is certainly not the case for many of the combination results presented here, so statements of significance should be interpreted with caution.

	Roc	NN	LDA	Net
avg.prec	0.369	0.398	0.418	0.429
avg.rank	1.90	2.37	2.82	2.91

Table 3: Average precision at all relevant documents in the top 1000 (avg.prec) and average per-query rank of these values (avg.rank) for the classifiers presented in Table 1.

For comparison, we also present the results for the same queries evaluated using average precision at all relevant documents in the top 1000 (a rank-based measure) in Table 3. Similar to Table 2, the avg.rank row is computed by ranking the methods by average precision for each query and averaging the ranks thus obtained over all 100 queries. Once again the neural network yields the best performance, although LDA is not significantly worse according to this measure. It seems that the logistic model used by the network is slightly more appropriate for probability estimation than the gaussian model used by LDA. However, the network also uses a larger feature set than LDA, so there are other factors to consider. For more details on the trade-offs between feature set type and learning algorithm see [Schütze et al., 1995]. In general, we conclude that the neural network is the best performing and most robust algorithm for document filtering with large training sets. The question remains whether we can extract useful information from the remaining algorithms using a combination strategy.

4 Averaging Strategies

Given that the four learning algorithms use different document representations and optimization strategies, there is some reason to hope that a combination of the classifiers might perform better than any individual method. The most basic approach to combination is to take the simple average of the probability estimates from each classifier. However, given that we are working in a probabilistic domain, it may make more sense to look at averaging the log-odds ratios ($\log(p/(1-p))$). Table 4 gives a practical demonstration of the trade-off between these two methods.

p1	p2	p3	p4	avg.prob	avg.odds
0.999	0.5	0.5	0.5	0.625	0.849
0.99	0.5	0.5	0.5	0.623	0.759
0.9	0.5	0.5	0.5	0.600	0.634
0.5	0.5	0.5	0.5	0.500	0.500
0.1	0.5	0.5	0.5	0.400	0.366
0.01	0.5	0.5	0.5	0.378	0.241
0.001	0.5	0.5	0.5	0.375	0.151

Table 4: An example illustrating the change in average probability (avg.prob) and average log-odds ratio (avg.odds) based on changes in the one of the individual probability estimates (p1-p4).

Let us assume the four different probability estimates are being averaged. If three of these estimates remain fixed and the fourth is allowed to vary arbitrarily, we get two different patterns, depending on the choice of averaging strategy. The average probabilities have lower and upper asymptotes of 0.375 and 0.625 respectively and never extend beyond these bounds. On the other hand, the average log odds (reconverted to a probability) is bounded by 0 and 1. In practice, this means that if one classifier is very certain that a document is relevant or not relevant, the average log odds will reflect this certainty much more directly than will the average probability.

threshold	Net		avg.prob		avg.odds	
	score	rank	score	rank	score	rank
p75	9.65	2.18	3.60	1.67	6.33	2.14
p50	27.9	2.29	22.4	1.88	22.3	1.84
p25	191	2.25	185	2.13	177	1.62
avg.prec	0.429	1.46	0.443	2.24	0.445	2.31

Table 5: Average score and rank over the query sample for the neural network (Net), for averaging based on probabilities (avg.prob) and for averaging based on log-odds ratios (avg.odds).

Table 5 compares these two combination strategies to the best performing individual classifier (neural network). These results present an interesting contrast. While the simple combination strategies are able to rank documents better than the network (difference is statistically significant), they do considerably less well at estimating probabilities. For the filtering evaluation, the neural network works consistently better than the combination strategies, although there are some interesting differences depending on the threshold. For p = 0.75, the average probability is much worse, while for p = 0.25, it is the average log odds which is clearly failing.

Table 4 can help illuminate these inconsistencies. The average probabilities will have much less variability than the average log odds, as discussed above. Since a probability threshold of p = 0.75 is relatively extreme, averaging of probability estimates results in many fewer documents exceeding the threshold than averaging of log odds. For a low threshold such as p = 0.25, the situation is reversed. In either case, returning more documents seems to improve performance, indicating a tendency for the combination strategies to underestimate the true probability of relevance.

threshold	Net		avg.prob		avg.odds	
	score	rank	score	rank	score	rank
p75	9.65	1.82	11.0	2.08	11.1	2.10
p50	27.9	1.74	31.6	2.22	31.2	2.04
p25	191	2.32	177	1.87	175	1.82

Table 6: Average score and rank over the query sample from Table 5 after renormalization of the probability estimates.

As empirical evidence suggests that the averaging of probability estimates is not the ideal approach, we look into another option. Ranking based on combined scores performs well even if the actual probability estimates are biased, which leads to an alternative filtering strategy: use the ranking generated by averaging but renormalize the probability estimates via logistic regression using the relevance judgements from the training set. The combined score for each document is treated as the predictor variable for a univariate logistic model which adjusts the probability estimates to reflect the observed distribution of relevant documents. The results of this experimental run appear in Table 6. We find that after renormalization, the probability estimates are much more accurate, scoring significantly better than the neural network for p = 0.50 and 0.75. However, for p = 0.25 both averaging techniques perform significantly worse than the network. For the moment, we have no good explanation for this contrast. However, we can conclude that method combination can help filtering performance for the higher probability thresholds.

5 Complex Combination Strategies

Averaging is the simplest possible combination method. It can be generalized by seeking the best linear combination of the classifiers. For the standard ad hoc search problem, the only source of evidence for optimization is previously evaluated queries. This information has been used very successfully by Bartell et al. [Bartell et al., 1994] and others to optimize the weights of different types of evidence. The result is a single set of weights that can be applied to all future queries. For the document filtering problem, we can attempt to estimate combination weights on a per-query basis using the training data. That is, we now think of combination as a meta-filtering problem, where the inputs are the probabilities estimated from the original set of filtering strategies. The relevance judgements are used for training both the original filters and the meta-filter.

While this approach seems reasonable on the surface, overfitting looms as a serious problem. By combining multiple results, we are adding more parameters to the filtering model, and their additional variability may outweigh the extra information obtained from combining multiple classifiers. In order to reduce overfitting, we try to remove the dependence of the probability estimates obtained from the training data on the relevance judgements by using 5-fold cross-validation. The training set is partitioned into 5 sections. One section is removed and the filtering algorithms are trained on the remaining four sections. The resulting filtering rule is used to estimate probabilities on the removed section. This process is repeated for all five sections. For example, one filtering rule is trained on sections 2 through 5 and applied to section 1 to compute probability estimates for section 1 that are then used for combination; a second filtering rule is trained on sections 1, 3, 4 and 5 and applied to section 2 to compute probability estimates for section 2; and so forth.

The resulting probability estimate for each training document is independent of its relevance judgement and an unbiased estimator of the true probability of relevance. Therefore, when the relevance judgement is used again to model the optimal combination weight, the weights will not be biased towards the filtering technique that best fit the training data. Rather, they will tend to prefer the filtering techniques that produce the best probability estimates on unseen data.

The natural approach to partitioning the training set is to use random sampling. However, in IR, this is generally not the ideal procedure, due to the fact that there are very often duplicate or near-duplicate documents. If two versions of the same relevant document are put in different partitions, then the probability estimates are likely to be heavily biased in favor of that document. Therefore, we begin by clustering the document set into many small groups of similar documents. These small clusters are then randomly assigned to partitions. This approach should insure that the validated probability estimates are not unduly influenced by duplicate documents.

We have developed three different methods for the meta-filtering step of the classification rule. The first two represent generalizations of the averaging strategies applied in the previous section. Rather than use the average probability, we can find the linear combination of filtering probability estimates which minimizes the sum of the squared difference between itself and the binary vector of relevance judgements. This is known as linear regression [Fuhr, 1989], and the solution can be generated from a few simple matrix operations. Instead of averaging the log odds scores, we can use logistic regression [Fuhr & Pfeifer, 1994, Cooper et al., 1994] to estimate the linear combination of odds scores which maximizes the likelihood according to a logistic model. The parameter estimates for the logistic model can be solved by the iterative application of weighted least squares techniques.

These regression techniques have been applied to IR problems in the past. In this case, the numerical computations are much simpler since there are only a small number of techniques to combine rather than a large number of document features. The parameters for the least squares model are not restricted in any way, and therefore may take on negative values. A negative parameter estimate for a particular filtering strategy might indicate that it is better predictor of non-relevance than of relevance! However, dependence complicates the issue since a negative estimate for one classifier might compensate an overly positive estimate for another classifier in the fitting process. We choose to interpret negative parameter estimates as evidence that the combination strategy is overfitting the data. Therefore, we restrict all parameters to be greater than zero using the following algorithm. All negative parameters in the model are set to zero and the model refit using only the remaining parameters. This requires multiple iterations of least squares for linear regression. Since logistic regression is already fit iteratively, it is only a question of eliminating the negative parameters and continuing the iterative process using the current estimates as starting values.

Regression techniques are only one possible approach to combining filtering estimates. Many other techniques from machine learning would be equally applicable. As we have learned that improving the ranking of documents through combination is easier than improving the probability estimates, we concentrated on the former problem first, as success there is deemed to be a prerequisite for a possible solution to the latter problem. For our third technique, we decided to try to directly optimize the combination weights for a rank-based measure (average precision). Unfortunately, the most efficient numerical optimization algorithms are designed for differentiable functions, while average precision is a piecewise constant function.

As the parameter space is relatively small (4 dimensions), we adopted a simple approach based on direct grid search of the parameter space. We imposed the additional constraints that the weights be non-negative and sum to one and then selected a grid of 81 points on the unit simplex in 4 dimensions. While this approach is neither efficient nor guaranteed to come close to the optimal solution, it is very easy to implement. If it appears to have potential, we may consider more sophisticated options in the future. To clarify, our direct optimization procedure works as follows. The average precision at each of the 81 grid points is computed over the training data and the set of weights that generates the highest score is used to combine the filters on the test data.

While we expected significant differences between the logistic and the linear model, in practice we found that they produced basically identical performance. Therefore, we included only the linear regression results in Table 7. None of the more complex weight optimization strategies improved the document ranking when compared to the average log odds, which performed significantly better than the other strategies. We then duplicated this experiment using the utility functions for evaluation (results not shown) and found that neither regression nor direct optimization improved filtering performance[4]. Surprisingly, averaging the probability estimates followed by a logistic transformation works better than logistic regression applied directly to the probability estimates, indicating that good combination weights are difficult to estimate from our training data. It appears that simple strategies work best for combining the filtering algorithms that we have developed.

threshold	Net	avg.odds	regression	grid search
avg.prec	0.429	0.445	0.439	0.439
avg.rank	1.65	3.08	2.72	2.55

Table 7: Average precision and rank for average log odds (avg.odds), weights estimated by linear (regression), and weight optimization via (direct) search.

6 Conclusions

In this paper, we have examined and tested the ability of various combination strategies to improve performance. We have found that simple averaging of probabilities or log odds ratios generates a significantly better ranking of the documents. This translates directly into improved performance for rank-based measures used for routing evaluation, such as average precision at all relevant documents. Direct combination does not generate better probability estimates. However, when the average scores are renormalized via logistic regression, the resulting probability estimates are better than any individual filtering strategy for probability thresholds 0.50 and 0.75. We have not found a combination strategy that works well for estimating probabilities less than 0.50.

To build on this success, we attempted to fit query-specific combination weights from the training data. We generated parameter estimates using both linear and logistic regression but failed to reach the standard set by the simple averaging strategies. A final effort which directly optimized the evaluation measures over the training set was equally unsuccessful. Defeated in this endeavor, we now take stock of the reasons for our failure.

Combination strategies work best when the results being combined are generated independently. While our techniques use a number of different document representations and ranking strategies, they are all trained on the same documents, which makes them far from independent. In order to examine this assumption, we computed the correlation matrix for the various probability estimates for each query, and then averaged over the query sample (shown in Table 8). From these results, the strong patterns of dependence among the classifiers are obvious. LDA and the Neural Network, which both use features generated by Latent Semantic Indexing, have a correlation of 0.834! In hindsight, it is not surprising that our combination strategies were not more successful.

As a further test, we evaluated the test data over the 81 selected grid points used in our previous experiments and se-

[4] For the grid search, we tried both optimizing the utility measures directly and taking the optimal solution based on average precision and renormalizing the probability estimates

	Roc	NN	LDA	Net
Roc	1.000	0.663	0.309	0.371
NN	0.663	1.000	0.538	0.593
LDA	0.309	0.538	1.000	0.834
Net	0.371	0.593	0.834	1.000

Table 8: Correlation of probability estimates between queries (averaged over queries).

lected the optimal combination weights for each query separately. This should give us a rough estimate of the maximum degree of improvement that we can expect from combining these estimators. The final average precision score generated by this experiment was 0.465, a rather small improvement over the 0.429 obtained by the Neural Network. In this light, the jump from 0.429 to 0.445 created by simple averaging seems much more impressive. Basically, the correlation between methods is high enough to preclude any substantial improvements in performance.

In reality, neither direct averaging nor averaging of log odds (LO) is ideal for combining probability estimates. Using Bayesian inference to derive the correct formula for the combination of evidence [Pearl, 1988], we find that given classifiers $C_1 \ldots C_n$ which are assumed to be conditionally independent given relevance (R) and conditionally independent given non-relevance ($\overline{R}$):

$$LO_B = \log O(R|C_1 \ldots C_n) = \sum_{i=1}^{n} \log O(R|C_i) - (n-1) \cdot \log O(R)$$

See the appendix for a proof. Let us compare this result with our strategy of averaging the log odds, i.e.

$$LO_A = \log O(R|C_1 \ldots C_n) = \frac{1}{n} \sum_{i=1}^{n} \log O(R|C_i)$$

As the combined log odds are proportional to the sum of the component odds ratios in both methods, they are equivalent with respect to ranking the documents (the prior odds $O(R)$ is a constant), which means that evaluation using average precision would give identical scores.

However, if the classifiers were conditionally independent, simply averaging the log odds would significantly bias the probability estimates. From the formula below:

$$LO_B = LO_A + (n-1) \cdot (LO_A - \log O(R))$$

we find that the direction of the bias will depend on the difference between the average log odds LO_A and the log prior odds. In general, the average log odds will be larger than the log prior odds for documents with a reasonably high probability of relevance. This indicates that using LO_A to combine the estimates will often lead us to strongly underestimate the true probability of relevance, which is exactly the behavior we observed in our experiments. Therefore, we have empirical evidence that Bayesian combination of evidence (i.e. using LO_B) may be more accurate. Unfortunately, we did not have time to test this hypothesis for our paper. Since our classifiers are far from independent, it is difficult to make any firm conclusions. We plan to test the Bayesian inference model on more appropriate classifiers in our future work.

Appendix

Assume classifiers $C_1 \ldots C_n$ are conditionally independent given relevance (R) and conditionally independent given non-relevance ($\overline{R}$), then:

$$(*) \ \log O(R|C_1 \ldots C_n) = \sum_{i=1}^{n} \log O(R|C_i) - (n-1) \cdot \log O(R)$$

Proof: (by induction)

Condition $(*)$ is satisfied trivially for $n = 1$. Assume that condition $(*)$ is true for $n-1$ and demonstrate that it is true for n.

$$O(R|C_1 \ldots C_n) = \frac{P(R|C_1 \ldots C_n)}{P(\overline{R}|C_1 \ldots C_n)} = \frac{P(C_1 \ldots C_n|R) \cdot P(R)}{P(C_1 \ldots C_n|\overline{R}) \cdot P(\overline{R})}$$

$$= \frac{P(C_1 \ldots C_{n-1}|R) \cdot P(C_n|R) \cdot P(R)}{P(C_1 \ldots C_{n-1}|\overline{R}) \cdot P(C_n|\overline{R}) \cdot P(\overline{R})}$$

$$= \frac{\frac{P(R|C_1 \ldots C_{n-1})P(C_1 \ldots C_{n-1})}{P(R)} \cdot \frac{P(R|C_n)P(C_n)}{P(R)} \cdot P(R)}{\frac{P(\overline{R}|C_1 \ldots C_{n-1})P(C_1 \ldots C_{n-1})}{P(\overline{R})} \cdot \frac{P(\overline{R}|C_n)P(C_n)}{P(\overline{R})} \cdot P(\overline{R})}$$

$$= \frac{P(R|C_1 \ldots C_{n-1}) \cdot P(R|C_n) \cdot P(\overline{R})}{P(\overline{R}|C_1 \ldots C_{n-1}) \cdot P(\overline{R}|C_n) \cdot P(R)}$$

$$= \frac{O(R|C_1 \ldots C_{n-1}) \cdot O(R|C_n)}{O(R)}$$

Which implies that:

$$\log O(R|C_1 \ldots C_n) = \log O(R|C_1 \ldots C_{n-1}) + \log O(R|C_n) - \log O(R)$$

Applying condition $(*)$ for $\log O(R|C_1 \ldots C_{n-1})$ gives:

$$\log O(R|C_1 \ldots C_n) = \sum_{i=1}^{n} \log O(R|C_i) - (n-1) \cdot \log O(R)$$

References

[Ali & Pazzani, 1995] Ali, K. M., & Pazzani, M. J. (1995). On the link between error correlation and error reduction in decision tree ensembles. Technical Report 95-38, Department of Information and Computer Science, University of California at Irvine.

[Bartell et al., 1994] Bartell, B. T., Cottrell, G. W., & Belew, R. K. (1994). Automatic combination of multiple ranked retrieval systems. In *Proceedings of SIGIR '94*, pp. 173–181.

[Belkin et al., 1993] Belkin, N., Cool, C., Croft, W., & Callan, J. (1993). The effect of multiple query representations on information retrieval system performance. In *Proc. of the 16th ACM/SIGIR Conference*, pp. 339–346.

[Belkin et al., 1995] Belkin, N., Kantor, P., Fox, E., & Shaw, J. (1995). Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management*, 31(3),431–448.

[Breiman, 1993] Breiman, L. (1993). Stacked regression. Technical report, Department of Statistics, University of California at Berkeley.

[Breiman, 1994] Breiman, L. (1994). Bagging predictors. Technical Report 421, Department of Statistics, University of California at Berkeley.

[Cooper et al., 1994] Cooper, W. S., Chen, A., & Gey, F. C. (1994). Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. pp. 57–66. In [Harman, 1994].

[Cutting et al., 1991] Cutting, D. R., Pedersen, J. O., & Halvorsen, P.-K. (1991). An object-oriented architecture for text retrieval. In *Conference Proceedings of RIAO'91, Intelligent Text and Image Handling, Barcelona, Spain*, pp. 285–298. Also available as Xerox PARC technical report SSL-90-83.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6),391–407.

[Drucker et al., 1993a] Drucker, H., Schapire, R., & Simard, P. (1993a). Improving performance in neural networks using a boosting algorithm. In Hanson, S. J., Cowan, J. D., & Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pp. 42–49. Morgan Kaufmann Publishers, San Mateo CA.

[Drucker et al., 1993b] Drucker, H., Shapire, R., & Simard, P. (1993b). Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4),705–719.

[Fox & Shaw, 1994] Fox, E. A., & Shaw, J. A. (1994). Combination of multiple searches. In *The 2nd Text Retrieval Conference (TREC-2), NIST SP 500-215*, pp. 243–252.

[Fuhr, 1989] Fuhr, N. (1989). Optimum polynomial retrieval funcions based on the probability ranking principle. *ACM Transactions on Information Systems*, 7(3),183–204.

[Fuhr & Pfeifer, 1994] Fuhr, N., & Pfeifer, U. (1994). Probabilistic information retrieval as a combination of abstraction, inductive learning, and probabilistic assumptions. *ACM TOIS*, 12(1).

[Hampshire & Waibel, 1990] Hampshire, J. B., & Waibel, A. H. (1990). A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Neural Networks*, 1(2),216–228.

[Harman, 1994] Harman, D., editor (1994). *The Second Text REtrieval Conference (TREC-2)*. U.S. Department of Commerce, Washington DC. NIST Special Publication 500-215.

[Haussler et al., 1994] Haussler, D., Kivinen, J., & Warmuth, M. K. (1994). Tight worst-case loss bounds for predicting with expert advice. Technical Report UCSC-CRL-94-36, Baskin Center for Computer Engineering and Information Sciences, University of California at Santa Cruz.

[Hearst et al., 1996] Hearst, M., Pedersen, J., Pirolli, P., Schütze, H., Grefenstette, G., & Hull, D. (1996). Xerox TREC-4 site report. In Harman, D., editor, *The Third Text REtrieval Conference (TREC-4)*, Washington DC. U.S. Department of Commerce. to appear.

[Hearst, 1994] Hearst, M. A. (1994). Multi-paragraph segmentation of expository discourse. In *Proceedings of the 32nd Meeting of the Association for Computational Linguistics*.

[Hull, 1993] Hull, D. (1993). Using statistical testing in the evaluation of retrieval performance. In *Proc. of the 16th ACM/SIGIR Conference*, pp. 329–338.

[Hull, 1994] Hull, D. (1994). Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of SIGIR '94*, pp. 282–289.

[J. Ghosh & Beck, 1992] J. Ghosh, L. D., & Beck, S. (1992). A neural network based hybrid system for detection, characterization and classification of short-duration oceanic signals. *IEEE Jl. of Ocean Engineering*, 17(4),351–363.

[Jacobs, 1995] Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural Computation*, 7,867–888.

[Jacobs et al., 1991] Jacobs, R. A., Jordan, M. I., Nowlan, S., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3,1–12.

[Kwok, 1995] Kwok, K. L. (1995). A network approach to probabilistic information retrieval. *ACM Transactions on Information Systems*, 13(3),324–353.

[Lee, 1995] Lee, J. H. (1995). Combining multiple evidence from different properties of weighting schemes. In *Proceedings of SIGIR '95*, pp. 180–188.

[Lei Xu & Hinton, 1995] Lei Xu, M., & Hinton, G. E. (1995). An alternative model for mixtures of experts. In G. Tesauro, D. S. T., & Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge MA.

[Lewis, 1995] Lewis, D. D. (1995). Evaluating and optimizing autonomous text classification systems. In *Proceedings of SIGIR 95*, pp. 246–254.

[Lewis, 1996] Lewis, D. D. (1996). The TREC-4 filtering track. In Harman, D., editor, *The Third Text REtrieval Conference (TREC-4)*, Washington DC. U.S. Department of Commerce. to appear.

[Littlestone & Warmuth, 1992] Littlestone, N., & Warmuth, M. K. (1992). The weighted majority algorithm. Technical Report UCSC-CRL-91-28, Baskin Center for Computer Engineering and Information Sciences, University of California at Santa Cruz.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, pp. 34–37. Morgan Kaufmann.

[Perrone & Cooper, 1995] Perrone, M. R., & Cooper, L. N. (1995). When networks disagree: Ensemble methods for hybrid neural networks. In Mammone, R. J., editor, *Artificial Neural Networks for Speech and Vision*, number 4 in Chapman & Hall Neural Computing. Chapman & Hall.

[Salton et al., 1983] Salton, G., Fox, E. A., & Wu, H. (1983). Extended boolean information retrieval. *Communications of the ACM*, 26(11),1022–1036.

[Saracevic & Kantor, 1996] Saracevic, T., & Kantor, P. (1996). A study of information seeking and retrieving iii: Searchers, searches, overlap. *Journal of the American Society for Information Science*, 39(3),197–216.

[Schütze et al., 1995] Schütze, H., Hull, D. A., & Pedersen, J. O. (1995). A comparison of classifiers and document representations for the routing problem. In *Proceedings of SIGIR 95*, pp. 229–237.

[Schütze & Pedersen, 1994] Schütze, H., & Pedersen, J. O. (1994). A cooccurrence-based thesaurus and two applications to information retrieval. In *Proceedings of RIAO*, pp. 266–274, Rockefeller University, New York.

[Tresp & Taniguchi, 1995] Tresp, V., & Taniguchi, M. (1995). Combining estimators using non-constant weighting functions. In G. Tesauro, D. S. T., & Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge MA.

[Tumer & Ghosh, 1995] Tumer, K., & Ghosh, J. (1995). Theoretical foundations of linear and order statistics combiners for neural pattern classifiers. *IEEE Transactions on Neural Networks*.

[Turtle & Croft, 1991] Turtle, H., & Croft, W. B. (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3),187–222.

[Wolpert, 1992] Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5,241–259.