

Prospective Data Fusion for Batch Filtering *

Andrei Anghelescu*, Endre Boros^R, Dmitriy Fradkin*,
David Lewis[†], Vladimir Menkov**,
David Neu^R Kwong Bor Ng⁺,
Paul Kantor⁼

*angheles@cs.rutgers.edu, boros@rutcor.rutgers.edu,
dfradkin@paul.rutgers.edu, ddlewis@worldnet.att.net
vmenkov@cs.indiana.edu, neu@rutcor.rutgers.edu,
kbnng@scils.rutgers.edu, kantor@scils.rutgers.edu*

19 May 2003

Abstract

This paper explores the problem of “global” data fusion rules, or combination of evidence. Three rather different schemes for computing the similarity between documents and a set of judged documents are employed. The goal is to learn whether rules for score combination (sometimes called “fusion rules”) can be *learned* from a random selection of filtering problems **L** and then applied to another collection of filtering problems **T**. Results are encouraging, and set the stage for further research into “when” (that is, under what conditions) it will be effective to use two or more systems in fusion.

1 Introduction

The work reported here is part of a larger project on identifying interesting messages in large streams of data. The question at issue is whether having multiple systems or schemes for information filtering can lead to better overall system performance. This question can be posed in two ways, which we will call “local” and “global”. The local question asks whether, for a given profile or

Supported in Part by the National Science Foundation under Grant Number EIA-0087022. PBK and KBN are supported in part by the Advanced Research and Development Activity (ARDA) AQUAINT Program under contract number 2002-H790400-000. The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency. Author Affiliations: () Division of Computer Sciences, Rutgers, the State University of New Jersey; (^R) RUTCOR, Rutgers Center for Operations Research; ([†]) Independent Consultant, Chicago IL. (**) Independent Consultant, Penticton, British Columbia Canada; (⁺) Queens College, City University of New York; (=) SCILS and DIMACS, Rutgers, the State University of New Jersey, to whom correspondence should be addressed.

topic, it is possible to find an excellent rule for combining the scores produced by two systems, so that the combined rule yields higher performance than the individual methods, [1, 15, 13, 16, 8, 9, 14].

In this setting, the obvious competitor is the best of the two individual methods, which can be presumed known after some period of training. Thus, the question becomes, “can the combined system produce performance better than that of the better system?” There is considerable evidence that for some topics this is true and for others it is not. The *global* question can be framed as follows: If we have information about the performance of two individual systems and their possible combinations on a set of training topics (denoted by $\mathbf{L}$), can we form a rule for combination - which can be carried over with confidence - to a completely different set of topics, which we will call the “test topics” (denoted by $\mathbf{T}$)?

This global question is obviously of interest to system designers and to those who must purchase or select among multiple systems for message filtering.

In this research, the several schemes used to produce different scores are the result of training exercises to refine the classification rule. For the voting method, the training was done on the TREC2002 [4] Batch filtering task. For the other two methods the training was done on the TREC2002 adaptive filtering task. Thus the results of this work are intended to shed light on the underlying issues of data fusion, and do not represent “post-hoc” claims as to performance achievable in the TREC2002 setting. Although the results described here are evaluated using a TREC-style metric, the results here are not TREC results and the data submitted here were not necessarily presented at any TREC conference.

2 Preliminaries

In what follows, we will examine only linear rules for the combination of scores. Thus it is not, strictly speaking, necessary to perform any preliminary normalization. However, the discussion and the interpretation of the results are much simpler if we first normalize the raw scores, $S_i^{raw}(d)$, assigned by system i to document d , according to the formula:

$$s_i(d) = \frac{S_i^{raw}(d) - \min_d(S_i^{raw}(d))}{\max_d(S_i^{raw}(d)) - \min_d(S_i^{raw}(d))} \quad (1)$$

Note that we have suppressed a topic index, which will be needed in the eventual discussion.

We form the fusion rule for a pair of systems (1 and 2) by a two-step linear discriminant process. For a given topic, all the documents whose relevance judgment is known can be represented by points in the $s_1 - s_2$ plane. The first step is to define the normal vector, β to a candidate separating line. This is defined by minimizing the objective function

$$M(\beta) = \sum_{d \in P} (1 - \beta \cdot \mathbf{s}(d))^2 + \sum_{d \in N} (\beta \cdot \mathbf{s}(d))^2, \quad (2)$$

where $\mathbf{s}(d) = (\mathbf{s}_1(d), \mathbf{s}_2(d))$, and $a \cdot b$ is the Euclidean inner product of a and b in the plane. This is a logistic regression. It is not assured of finding the very best solution to the classification problem, but is a widely used approach to estimating the separating hyperplane to which β is the normal.

The selected normal vector is defined by

$$\beta^* = \operatorname{argmin}_{\beta} (M(\beta)). \quad (3)$$

Since β is a pair of numbers but contains only one piece of information, we replace it by a single number α defined so that

$$\alpha/(1 - \alpha) = \beta_1/\beta_2 \quad (4)$$

In addition, the system must select a threshold value τ which will define the classification rule applied to incoming documents. Specifically, a document will be sent for evaluation if and only if $\alpha s_1 + (1 - \alpha)s_2 \geq \tau$. The result is that some number $P(\alpha, \tau)$ of positive (relevant) documents will be selected, and some number $N(\alpha, \tau)$ of negative (not relevant) documents will be selected. We use the ‘‘TREC-like’’ utility function

$$U_{11} = 2P(\alpha, \tau) - N(\alpha, \tau). \quad (5)$$

We define the optimal value of τ as $\tau^* = \operatorname{argmax}_{\tau} (U_{11}(\tau))$.

In reality, this process is done separately for each topic, and each pair of systems, resulting in a pair of parameters: α, τ (we suppress the asterisk at this point).

We note at the outset that our approach is not totally optimal because, in the results to be shown later in the paper, we actually use the modified TREC score, which was introduced to avoid runaway negative utility values, and is bounded from below. Results would presumably be somewhat different if this objective were also used to define α^*, τ^* .

3 The Specific Methods

The three different methods that we will describe are a simple Rocchio-type classifier **R** [12]; a classifier that uses the centroid of a positive (or relevant document), the centroid of the negative cases, and the origin in the document vector space, which we call the ratio classifier (represented by **C**); and, finally, a complex scoring and representation scheme which selects terms based on their

ability to resolve positive and negative documents. Since this scheme lets five different methods for computing the resolving power “vote” on the term selection, we denote it by $\mathbf{V}$.

In general, we represent documents in terms of the frequency with which specific stemmed (Porter stemmer) non-stop (SMART stop words) tokens appear. If $f(d, i)$ is the frequency with which the i -th term appears in document d then components of the vector representing d are given by $d_i = 1 + \log(f(d, i))$ with the value 0 if $f(d, i) = 0$. The weighting of terms is given by an “inverse document frequency” or IDF metric $\rho(t, t') = \delta_{t,t'} \log \frac{1+C}{1+C(t)}$. Here $C(t)$ is the number of documents in which the term t appears at least once, C is the total number of documents in the corpus, and $\delta_{t,t'}$ is the Kronecker delta, which vanishes unless the indices are equal, where it takes the value 1. Inner products are defined in terms of this metric tensor by the equation $(d, d') = \sum_{t,t'} f(d, t) \rho(t, t') f(d', t')$.

The Rocchio method is modified in that at each step the 30 terms having greatest weight in the updated classification vector (or query vector) are retained and all the others are dropped. This introduces some non-linearity in the learning process.

The specific algorithm selects the top $k = 30$ terms, according to the projection of the current form of the query onto the corresponding axis, $Q(t)$. Is given in pseudocode as

Algorithm 1 Query term selection

Require: query vector Q , k

```

1: for  $t \in Q$  do
2:   if  $Q(t) < 0$  then
3:      $Q(t) = 0$ 
4:   end if
5: end for
6:  $S = \text{reverse}(\text{sort}(Q))$ 
```

Ensure: $S[1 : \min(|S|, k)]$, the top k positive components of Q

Each of these three systems has been developed (and certain parameters fixed once for all) in experiments conducted for the TREC2002 conference. These parameters have not been optimized for the present experiments, but simply represent a possible starting point for the investigations on fusion.

The Rochhio and Centroid methods used for their training materials a complex set of information: (1) an original topic or query statement (2) a set of three judged relevant documents (3) a set of pseudo-judged documents. All materials are drawn from the the RCV1 corpus, i.e. Reuters newswire stories from August 20, 1996 through August 19, 1997. They are separated into a training set of 83,650 documents (20-Aug-1996 to 30-Sep-1996) and a test set of 723,141 documents (1-Oct-1996 to 19-Aug-1997) Both methods were implemented using the Lemur toolkit [17].

3.1 Details of the Rocchio Algorithm

The training stage of the algorithm consists of the following steps:

1. start with the encoding of the query description as Q_{up}
2. update Q with the labeled training examples (all are positive), using scalar weights represented by α, β

$$Q_{up} = \alpha Q_0 + \beta \sum_{d \in Train} d$$

3. order all documents in the training set by their scores relative to Q_{up} . The temporary score is the inner product: $s(d, Q_{up}) = (d, Q_{up})$
4. define the top $p\%$ as pseudo-positives, the remaining as pseudo-negatives
5. uniformly sample n_+ documents from the pseudo-positives (let the selected set be PP)
6. uniformly sample n_- documents from the pseudo-negatives (let the selected set be PN)
7. update Q with

$$Q_{up} = \alpha Q_0 + \beta \sum_{d \in PP} w_+ d - \gamma \sum_{d \in PN} w_- d,$$

where w_+, w_- are weights assigned to pseudo-positive and pseudo-negative documents, respectively.

8. calculate the threshold, τ_0 , that maximizes the U_{11} score [See Eq. 5], given Q and the set of training documents.

We determined experimentally that the following values of parameters work better than the others that we tried. (Generally, we tried increasing or decreasing each of these by a factor of two. Not all combinations were explored.):

α	1
β	1
γ	0.125
n_+	20
w_+	0.1
n_-	100
w_-	0.05

These parameters were used to produce the ‘‘Rocchio Model’’ document scores which are identified as $S_{\mathbf{R}}(d)$ in what follows.

3.2 Details of the Centroid Algorithm

The centroid algorithm orders documents by a measure of the ratio of their similarities to the centroids of the positive and negative examples. Thus it builds on the relevance feedback information available to Rocchio, with a key difference. Scores are calculated using the (regularized) ratio of distances between normalized vectors. Specifically, if $\mathbf{p}, \mathbf{n}$ are the unit vectors corresponding to the centroids of the positive and negative examples, and $\mathbf{d}$ is the unit vector corresponding to the document being scored, then

$$s_C(d) = \frac{1 - (\mathbf{n}, \mathbf{d})}{1 - (\mathbf{p}, \mathbf{d})} \quad (6)$$

If the denominator vanishes, the value 10^6 is used as a default.

3.3 The Voting Method

In this method each of five different schemes for assessing the “discriminatory power” of a term is computed. Each of these schemes then “casts one vote” for each of the terms to which it gives a score in the top 50. We then retain the top 50 terms, ranked by the number of votes that they have received. This method currently uses a Boolean formulation of the bag-of-words model.

For each of the $n > 0$ distinct terms in the document collection we associate an index in $V = \{1, 2, \dots, n\}$. Letting $\mathbb{B} = \{0, 1\}$, we represent each document in the collection as an n -dimensional *Boolean* vector $\mathbf{x} \in \mathbb{B}^V$. Each component of $\mathbf{x}$ corresponds to one of the distinct terms in the collection, with $x_i(d) = 1(0)$ if the i^{th} term is present in (absent from) the document d .

3.3.1 Ranking Functions

For each $i \in V$, each of the ranking functions is presented here in terms of the following four values

- $a_i \equiv$ the number of relevant documents containing the i^{th} term
- $b_i \equiv$ the number of irrelevant documents containing the i^{th} term
- $c_i \equiv$ the number of relevant documents which do not contain the i^{th} term
- $d_i \equiv$ the number of irrelevant documents which do not contain the i^{th} term

For each $i \in V$, the relationship between a_i , b_i , c_i and d_i and the document collection is given by the following 2×2 contingency table

	$y(d) = 1$	$y(d) = 0$	
$x_i(d) = 1$	a_i	b_i	$a_i + b_i = \theta_i$
$x_i(d) = 0$	c_i	d_i	$c_i + d_i = \theta_i$
	$a_i + c_i = P $	$b_i + d_i = N $	m

where the marginals θ_i and $\bar{\theta}_i$ represent the number of documents containing the i^{th} term and the number of documents which do not contain the i^{th} term respectively, and $y \in \mathbb{B}$ is defined as

$$y = \begin{cases} 1 & \text{if } x \in P, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, the marginals $|P|$ and $|N|$ are constant for all terms while the marginals θ_i and $\bar{\theta}_i$ vary for each term. The total number of documents in the collection is $C = a_i + b_i + c_i + d_i$ which is obviously also a constant.

For the simplicity of notations, we shall view all ranking functions as functions of the four parameters a , b , c and d , though clearly there are only two independent values among these. For the remainder of this discussion we shall drop the subscripts i .

In [3] we analyzed and compared a number of possible ranking functions, and based on that study, we selected 5 such functions for this TREC experiment. The original numbering of the functions is retained here for consistency with [3]:

Function μ_6

$$\mu_6 = \left| \frac{a}{a+c} - \frac{b}{b+d} \right| = \frac{|ad-bc|}{|P||N|} \quad (7)$$

is the absolute value of the difference between the number of relevant-irrelevant document pairs in the training collection which provide evidence that the i^{th} term is a good classifier of relevant documents and the number of relevant-irrelevant document pairs which provide evidence that the i^{th} term is a good classifier of irrelevant documents, normalized by the total number (i.e. both correctly distinguished and incorrectly distinguished) of relevant-irrelevant document pairs.

Function μ_{10}

$$\mu_{10} = \frac{ad+bc}{(a+c)(b+d)} = \frac{ad+bc}{ab+ad+bc+cd} = \frac{ad+bc}{|P||N|} \quad (8)$$

is the total number of relevant-irrelevant document pairs correctly distinguished by the i^{th} term, normalized by the total number of relevant-irrelevant document pairs in the training collection.

Function μ_γ

$$\mu_\gamma = \frac{ad}{(a+c)(b+d)} = \frac{ad}{|P||N|}$$

is an obvious variant of both μ_6 and μ_{10} .

Function μ_{11}

$$\mu_{11} = \frac{|ad-bc|}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} = \frac{|ad-bc|}{\sqrt{\theta\bar{\theta}|P||N|}} \quad (9)$$

is the absolute value of the *Pearson Product Moment Correlation* coefficient or simply the *correlation coefficient* for the Boolean variables x_i and y as defined above. It measures the degree to which these two variables have a linear relationship.

Function μ_{12}

$$\mu_{12} = \frac{(a+b+c+d)(ad-bc)^2}{(a+b)(c+d)(a+c)(b+d)} = \frac{m(ad-bc)^2}{\theta\bar{\theta}|P||N|} \quad (10)$$

is the χ^2 statistic for the Boolean variables x_i and y as defined above and provides another measure of association for these two variables. We recognize that μ_{11} and μ_{12} are monotone functions of each other for fixed collection size. Effectively, in what follows, the measure that they represent has been allowed to “vote twice”.

3.3.2 Training the Classifier

This section describes the feature selection method and the simple classifier used in the batch filtering sub-task of the filtering track.

The set of unique terms in the training set $P \cup N$ was ranked by each of the five ranking functions described in §3.3.1. Five intermediate feature sets, $S_{\mu i}$, $i \in \{6, 10, \gamma, 11, 12\}$ were constructed using the top ranking $K = 50$ terms of the corresponding ranking functions. Letting

$$\tilde{S} = \cup_{i \in \{6, 10, \gamma, 11, 12\}} S_i$$

we assigned a score $\psi(i) \in \{1, \dots, 5\}$ to each of the terms in $\tilde{S}$, defined as the number of sets S_{ξ} , $\xi \in \{6, 10, \gamma, 11, 12\}$ in which the term appeared. The final feature set S was constructed by selecting the $K = 50$ terms with the highest ψ scores.

Next, to each term in $i \in S$ we assigned the weight

$$\omega(i) = \frac{\frac{a_i + 0.5}{a_i + c_i + 1}}{\frac{b_i + 0.5}{b_i + d_i + 1}}$$

which can be thought of as a Yule-adjusted Bayesian weight of evidence, and to each document $d \in P \cup N$ we assigned the score

$$\Omega(d) = \sum_{i \in S} \log(\omega(i))x_i(d).$$

That is, each document projected onto the selected feature set S is assigned a score equal to the sum of the logarithms of the Bayesian weights of evidence for the terms it contains.

The batch filtering task requires the definition of a static classification rule which specifies whether each document in the test set should be considered relevant and retrieved, or irrelevant and ignored. The rule we utilized specifies that $d \in P[S] \cup N[S]$ will be retrieved if and only if $\Omega(d) \geq \tau$ for some $\tau \in \mathbb{R}$. The threshold τ was selected so as to optimize the utility measure U_{11} over the training set.

4 The Experiments

We considered 50 TREC topics that are called “Assessor” topics, for which judgments were produced by human analysts, and another set of topics called “Intersection” topics where judgments were produced on artificial sets formed by the intersection of previously classified sets. We treat these separately because the experience at TREC 2002 suggested that systems perform quite differently on them.

For each category, we did 25 replications of the following process: Randomly select 25 topics to be the training topics. Compute the average parameters by either of two methods described below. Test those parameters on the remaining 25 topics. Repeat this process with the two subsets interchanged.

Thus there were, for each type of topic, a total of 50 tests run.

The distinction between the two methods has to do with the selection of the fusion parameters (α, τ) . We define $\bar{\alpha}$ to be the average alpha for all the topics in the learning set. Under method M1, for each topic in the test set, we separately find the best threshold for that topic using its own best value of α . Then the values of α and τ are averaged separately to produce the values used in testing. Method 2 uses the computed value of $\bar{\alpha}$ and recomputes a new threshold for each topic in the training set, based on $\bar{\alpha}$. The average of those thresholds, $\bar{\tau}$, is then used to complete the specification of method M2.

5 The Results

The results of repeating this approach for all 3 possible combinations of filtering methods, for both the “assessor” and the “intersection” topics, and for both methods M1 (the hybrid method) and M2 (using means of the weight parameter and the threshold, are shown in the Tables that follow.

6 The Conclusions

As these results show, for each of the possible pairings, and for each of the specific methods and problem sets, there is a very good probability to do better than an oracle. (The oracle is presumed to know in advance which system will work better on each topic, and apply that system to the topic.) This probability, averaged over all 25 testing topics, is simply the fraction of the upper rectangle, in each graph of Table 3, that is included in the bars above the axis. A detailed table (a portion is shown in Table 2) shows that there are a number of topics for which the fusion or combination that we propose here is consistently (actually, 25 times out of 25) better than using an oracle.

We note that because we use a technique of repeated split half testing and training, our conclusions have a stochastic character. For example, Table 2 shows that for topic 105, the combination of Rocchio and Centroid schemes beats the performance of an oracle 100% of the time. This means that for each of the 25 different training sets that we chose (from the 49 available) the

Fusion	Score 1	Score 2	Method1			Method2		
			Fusion Score	Beats Best	Beats Stronger	Fusion Score	Beats Best	Beats Stronger
r-c-ass:	0.442	0.500	0.531	0.482	0.546	0.528	0.426	0.527
r-c-int:	0.462	0.384	0.474	0.377	0.609	0.478	0.377	0.627
r-v-ass:	0.441	0.369	0.443	0.309	0.380	0.440	0.246	0.344
r-v-int:	0.462	0.419	0.430	0.218	0.378	0.449	0.273	0.494
c-v-ass:	0.500	0.369	0.505	0.396	0.457	0.505	0.410	0.470
c-v-int:	0.384	0.419	0.443	0.366	0.636	0.445	0.341	0.636

Table 1: The columns show trec-style scores averaged over all topics and runs (separately for assessor and intersection topics). Rows are labelled by the pair of systems combined (from r=Rocchio, c=Centroid and v=Voting) and the type of topics (ass=Assessor and int=Intersection). The column “Beats Best” shows the fraction of runs where the combination performed better than either one of the classifiers. The column headed “Beats Stronger” shows the fraction of runs where the fusion beats the simple rule of picking the best of the two systems for the training split and using it for all the test cases.

resulting average fusion rule was highly effective for topic 105. For topic 104, this occurred in only 20 of the training cases, and so forth. Thus our method of studying the problem produces probabilistic estimates of the effectiveness of fusion, for a given problem. We take the average of these probabilities as a single measure of effectiveness of the entire approach. This is the number summarized in Table 1, under the heading “best”. When one has a rule that is effective about 50% of the time, it is clearly important to understand better which half of the cases it will work for.

It is not entirely clear how to estimate the statistical significance of results such as those presented here. One approach is to take, as a null hypothesis, that the result of this global fusion will be better than the performance of an oracle, simply by chance. That is, that the probability of beating the oracle, on any particular replication, for any particular topic, is 0.5. It follows that the chance (under the null hypothesis) H_0 that the fusion will beat an oracle 25 times out of 25 is just 2^{-25} . Now, in the upper left hand corner of Table 3, we see that the R-C combination, applied to 50 assessor topics, experienced this unlikely success for nine of the topics. Of course, under the null hypothesis, those topics could have been chosen in any of $\frac{50!}{9!41!}$ ways. So the probability of the observed event (or an even more impressive accumulation of 25 victories for the fusion over the oracle) in more than nine cases is given by:

$$p = \sum_{j \geq 9} \frac{50!}{j!(50-j)!} 2^{-25j} < 1.91 \times 10^{-57} \quad (11)$$

So, this particular null hypothesis (which seems a reasonable interpretation

of the notion that fusion is no better than an oracle) can be rejected with great confidence! The estimate of confidence should perhaps be reduced to some degree by the fact that the training sets for each topic have some overlap with each other. We do not explore that issue further here. Note also that we do not see another (even stronger) possible result, which would have fusion beat the oracle by a significant amount (say 20 to 5) in *all* of the topics. This would mean that fusion can be used, without regard to the specific topic at hand. In fact, it might be supposed that topics for which fusion is especially effective are somehow “unusual”. On the contrary, as the graphs of Table 4 show, these topics are in fact more or less central in the scatter plot based on the parameters of the fusion method.

Thus, while some questions remain open, these results do suggest that in some sense the “correct fusion rule” is characteristic of the two systems being combined, and not entirely conditioned on the specific filtering task. This opens the door to many avenues of investigation. Among the most promising are (1) application of other methods for learning the rule of combination in the two dimensional space defined by the scores [such as nonlinear methods, Support Vector Machines, etc.] (2) extension of the entire formalism to sets of three or more systems (3) a renewed look at the question of whether examination of other characteristics of systems (besides the scores that they assign to relevant and not relevant documents) can form a basis for either (a) specification of a fusion rule or (b) specification of whether any linear fusion rule is likely to be effective. This question has been raised by Ng in [9], and has been addressed also by Vogt and Cottrell in [16] but has not, to our knowledge, been satisfactorily resolved as yet. It appears that it is relatively easy to predict how well a fusion scheme will score overall, but it is far more difficult to predict whether (that is, for which topics) that overall performance will compare favorably to that of an oracle.

tID	Sb-r	Sb-c	Sb-comb	S-r	S-c	S-comb	Best	Just Beats Average
101	0.916	0.922	0.923	0.745	0.843	0.890	1.000	0.000
102	0.843	0.698	0.843	0.728	0.489	0.821	1.000	0.000
103	0.634	0.678	0.738	0.600	0.621	0.734	1.000	0.000
104	0.699	0.709	0.738	0.683	0.629	0.708	0.800	0.200
105	0.793	0.793	0.800	0.602	0.726	0.828	1.000	0.000
106	0.333	0.484	0.495	0.222	0.437	0.373	0.080	0.680
107	0.468	0.631	0.658	0.310	0.335	0.190	0.000	0.160
108	0.644	0.622	0.800	0.618	0.504	0.725	1.000	0.000
109	0.464	0.635	0.477	0.367	0.435	0.373	0.160	0.080
110	0.495	0.753	0.753	0.000	0.716	0.512	0.160	0.600
...	...	...	...	...	...	...	...	...
141	0.484	0.642	0.626	0.378	0.443	0.443	0.480	0.520
142	0.431	0.333	0.597	0.000	0.000	0.000	0.000	1.000
143	0.406	0.348	0.391	0.024	0.000	0.000	0.000	0.720
144	0.782	0.848	0.788	0.523	0.578	0.742	1.000	0.000
145	0.333	0.630	0.556	0.000	0.483	0.237	0.040	0.400
146	0.730	0.910	0.916	0.665	0.778	0.834	0.960	0.040
147	0.608	0.657	0.676	0.567	0.415	0.552	0.400	0.440
148	0.908	0.925	0.950	0.661	0.724	0.806	0.960	0.040
149	0.468	0.345	0.503	0.386	0.345	0.456	0.840	0.120
150	0.420	0.654	0.642	0.210	0.552	0.480	0.120	0.760
Av:	0.584	0.657	0.688	0.442	0.500	0.531	0.482	0.320

Table 2: Averaged results of 25 2-fold cross-validation runs, combining Rocchio and Centroid-Ratio methods (a number of rows were left out to save space) Columns headed “Sb-*” contain the best scores achievable using the specified method. Columns headed “S-*” contain the average of the scores achieved on cross-validation runs. The column headed “Best” shows the fraction of times that the combination was better than either of the single classifiers. The column headed “Just Beats Average” shows the fraction of times that the combination was better than the average of the two single classifiers but was not better than the best. The last row is the average of scores over all topics (i.e. of all the rows in the table, including those left out)

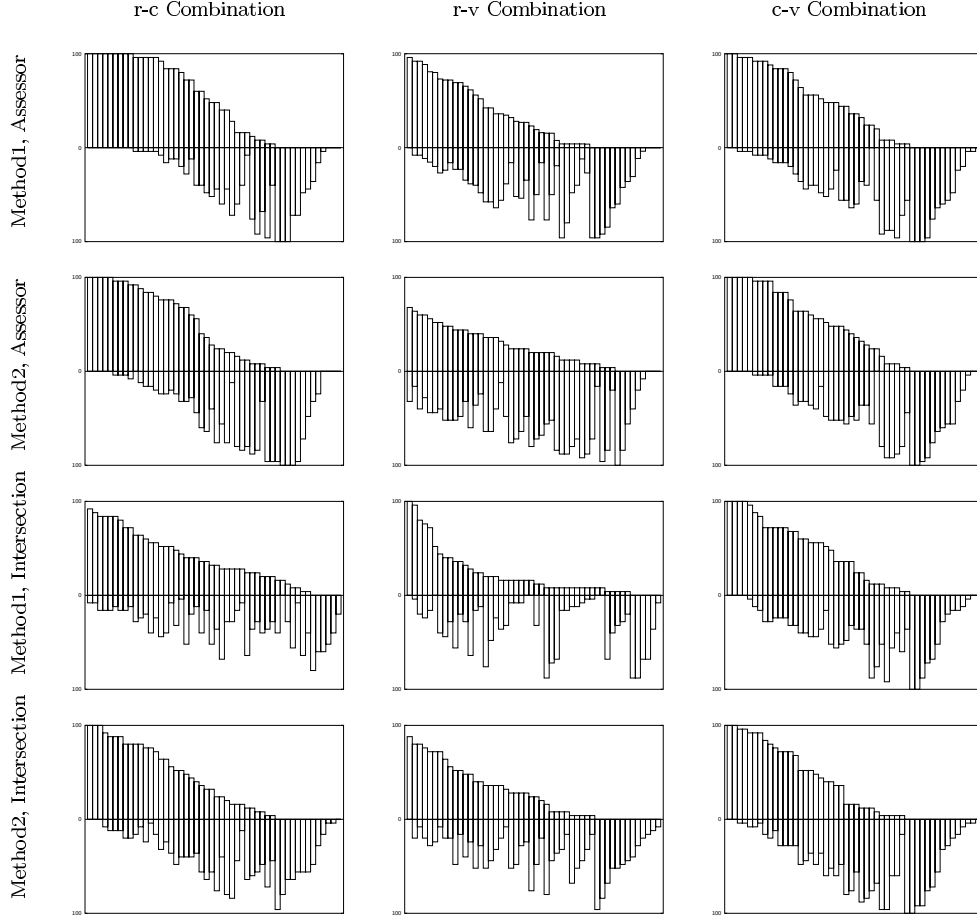


Table 3: Performance of fusion. Bars above the horizontal axis indicate, for each topic, the percentage of runs where the combination performed better than the best of the two single classifiers. Bars below the horizontal axis indicate the percentage of runs where the combination performed worse than the best classifier but better than the average of the two classifiers. Topics are sorted by the “top” value, so the order of topics on the horizontal axis is different in each of the twelve figures. The total height of the bar indicates the chance that the combination performs better than the average for the given topic.

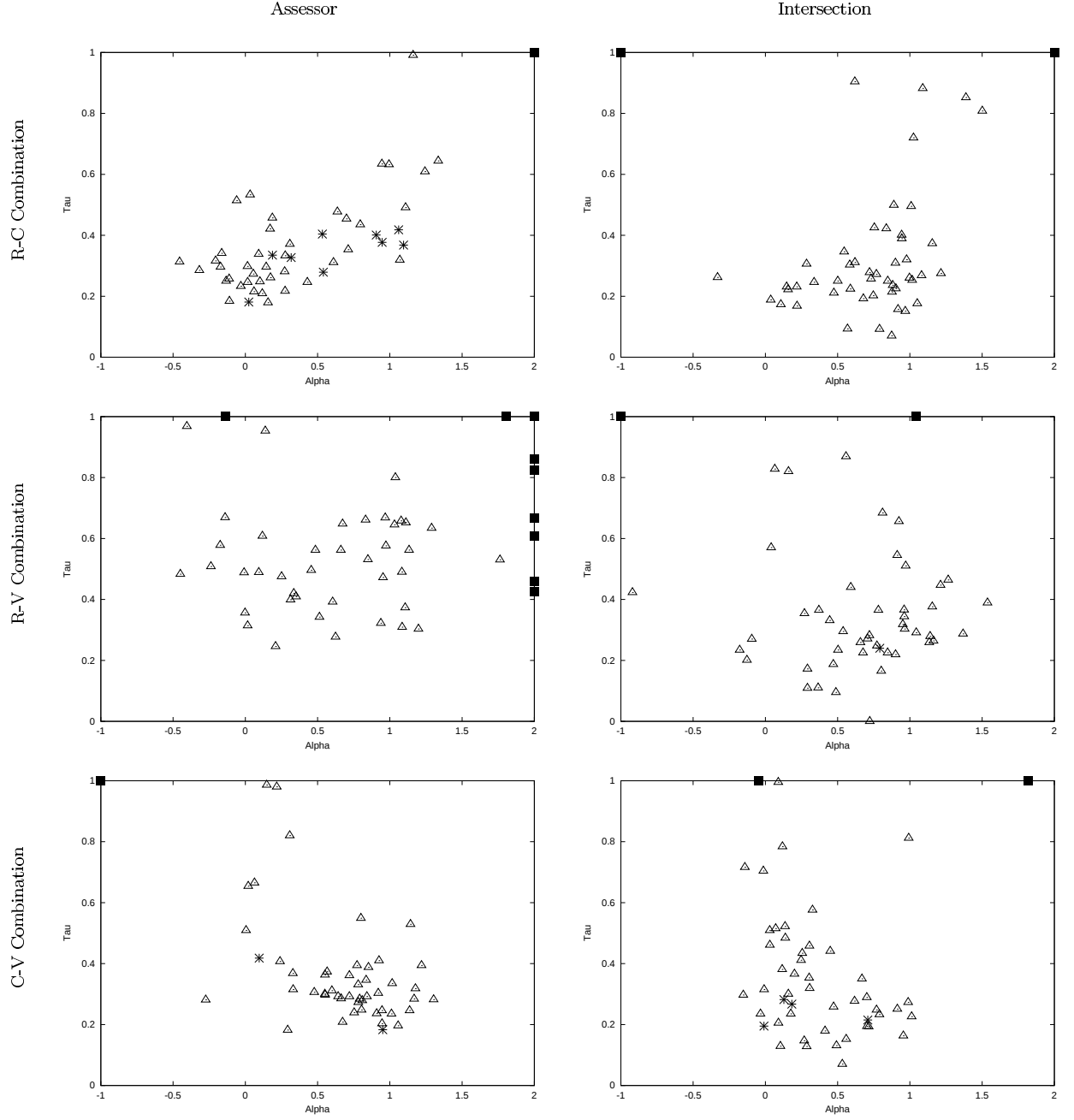


Table 4: The distribution of 50 topics, showing the best values of the fusion coordinates, for each topic, determined using Method M1. The x-coordinate is the value of α , the y-coordinate is the value of the threshold τ . The topics for which the fusion method always beats the oracle are marked by *. Topics for which the point lies outside the plot are indicated by ■, with the coordinate that is in the plotted range determining its position on the border.

7 Acknowledgments

Supported in Part by the National Science Foundation under Grant Number EIA-0087022. PBK and KBN are supported in Part by Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number 2002-H790400-000, the HITIQA project, of SUNY Albany and Rutgers. The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency.

References

- [1] B. Bartell, G. Cottrell, and R. Belew. Learning to retrieve information, 1995.
- [2] Endre Boros, Takashi Horiyama, Toshihide Ibaraki, Kazuhisa Makino, and Mutsunori Yagiura. Finding essential attributes from binary data. *Annals of Mathematics and Artificial Intelligence*, accepted.
- [3] Endre Boros and David J. Neu. Rank based feature selection in information retrieval. Technical report, RUTCOR, Rutgers University, 2003.
- [4] Harman D and url = "http://trec.nist.gov" Voorhees E. title = "The Text REtrieval Conference".
- [5] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall PTR, 1992.
- [6] D. D. Lewis. Feature Selection and Feature Extraction for Text Categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 212–217, San Mateo, California, 1992. Morgan Kaufmann.
- [7] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [8] Belkin N, Kantor PB, and Fox E Shaw J. Combining the evidence of multiple query representations for information retrieval. *Information Processing and Management*, 31(3):431–448, 1995.
- [9] Kwong Bor Ng. *An investigation of the conditions for effective data fusion in information retrieval*. PhD thesis, Rutgers The State University OF New Jersey - New Brunswick, 1998.
- [10] Gottfried E. Noether. *Introduction to Statistics: The Nonparametric Way*. Springer-Verlag, 1991.
- [11] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

- [12] J. J. Rocchio, Jr. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.
- [13] Joseph A. Shaw and Edward A. Fox. Combination of multiple searches. In *Text REtrieval Conference*, pages 0–, 1994.
- [14] Ibraev U, Ng KB, and Kantor P. Exploration of a geometric model of data fusion. In *Proceedings of the 2002 Conference of the American Society for Information Science and Technology.*, 2002.
- [15] Christopher C. Vogt. Adaptive combination of evidence for information retrieval.
- [16] Christopher C. Vogt and Garrison W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.
- [17] Chengxiang Zhai, Thi Nhu Truong, John Lafferty, Jamie Callan, David Fisher, Fangfang Feng, James Allan, and Bruce Croft. *Lemur*. <http://www-2.cs.cmu.edu/~lemur>, 2001.