# Probabilistic Combination of Text Classifiers Using Reliability Indicators: Models and Results

Paul N. Bennett
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA 15213
pbennett+@cs.cmu.edu

Susan T. Dumais
Microsoft Research
One Microsoft Way
Redmond, WA 98052
sdumais@microsoft.com

Eric Horvitz
Microsoft Research
One Microsoft Way
Redmond, WA 98052
horvitz@microsoft.com

## ABSTRACT

The intuition that different text classifiers behave in qualitatively different ways has long motivated attempts to build a better metaclassifier via some combination of classifiers. We introduce a probabilistic method for combining classifiers that considers the context-sensitive reliabilities of contributing classifiers. The method harnesses *reliability indicators*—variables that provide a valuable signal about the performance of classifiers in different situations. We provide background, present procedures for building metaclassifiers that take into consideration both reliability indicators and classifier outputs, and review a set of comparative studies undertaken to evaluate the methodology.

## Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning; I.5.1 [**Pattern Recognition**]: Models

## General Terms

Algorithms, Experimentation.

## Keywords

Text classification, classifier combination, metaclassifiers, reliability indicators

## 1. INTRODUCTION

Researchers have long pursued the promise of harnessing multiple text classifiers to synthesize a more accurate classification procedure via some combination of the outputs of the contributing classifiers. The pursuit of classifier combination has been motivated by the intuition that, because different classifiers work in related but qualitatively different ways, an appropriate overlaying of classifiers might leverage the distinct strengths of each method.

Classifiers can be combined in different ways. In one approach, a text classifier is composed from multiple distinct classifiers via a procedure that attempts to select the best classifier to use in different situations. For example, we may work to identify the most accurate classifier in some setting, seeking to learn about accuracy over output scores or some combination of output scores and features considered in the text analysis. Other procedures for combining classifiers consider inputs generated by the contributing classifiers. For example, in a voting analysis, a combination function considers the final decisions made by each classifier as votes that influence an overall decision about document classification. In a finer-grained approach to combining multiple classifiers, the scores generated by the contributing classifiers are taken as inputs to the combination function. Whichever approach to combination employed, the creation of enhanced metaclassifiers from a set of text classifiers relies on developing an understanding of how different classifiers perform in different informational contexts.

We have pursued the development of probabilistic combination procedures that hinge on learning and harnessing the *context-sensitive* reliabilities of different classifiers. Rather than rely solely on output scores or on the set of domain-level features employed in text-classification, we introduce the use of *reliability-indicator* variables—a set of features that provide a low-dimensional abstraction on context for learning about reliability. We borrow the reliability-indicator methodology from work initially presented by Toyama and Horvitz [26] to the automated vision community. They introduced the reliability-indicator learning and inference framework and showed how the approach could be applied in vision to integrate several distinct scene analyses into an overall higher-accuracy composite visual analysis. We have found that the reliability-indicator methodology is useful in the text classification realm for providing context-sensitive signals about accuracy that can be used to weave together multiple classifiers in a coherent probabilistic manner to boost overall accuracy.

We will first review related work on the combination of text-classification procedures. Then we introduce the use of reliability indicators in text classification, and show how we employ these variables to learn about the context-sensitive reliabilities of naïve Bayes, unigram, support vector machine (SVM), and decision-tree classifiers. We describe how we integrate the indicator variables with base-level features and scores output by classifiers to build meta-classifiers that show enhanced performance. We highlight our methodology and results by reviewing several sets of experiments. Finally, we summarize our contributions and discuss future directions.

## 2. RELATED WORK

The overlaying of multiple methodologies or representations has been employed in several areas of information retrieval. For ex-

*SIGIR'02,* August 11-15, 2002, Tampere, Finland.

ample, past research in information retrieval has demonstrated that retrieval effectiveness can be improved by using multiple, distinct representations [2, 11, 22], or by using multiple queries or search strategies [3, 24]. In the arena of text classification, several researchers have achieved improvements in classification accuracy via the combination of different classifiers [9, 13, 18, 30]. Others have reported that combined classifiers work well compared to some particular approach [1] but have not reported results that compare the accuracy of the classifier with the accuracies of the individual contributing classifiers. Similarly, systems that seek to enhance classification performance by applying many instances of the same classifier, such as in boosting procedures [23, 28], have compared the overall performance of the final methodology to other systems rather than to the weaker component learners.

Much of the previous work on combining text classifiers has used relatively simple policies for selecting the best classifier or for combining the output of multiple classifiers. As some examples, Larkey and Croft [13] used weighted linear combinations of system ranks or scores; Hull et al. [9] used linear combinations of probabilities or log odds scores; Yang et al. [30] used a linear combination of normalized scores; and Li and Jain [18] used voting and classifier selection techniques. Lam and Lai [12] use category-averaged features to pick a (possibly different) classifier to use for each category.

As we shall highlight below, in contrast to prior research on classifier combination, our work centers on the use of a richer probabilistic combination of inputs, using combination functions learned with Bayesian and SVM learning methods. In this respect, our approach is similar to work by Ting and Witten [25] in stacked generalization, although they did not apply their approach to text problems. We also report baseline comparisons with voting and classifier-selection techniques.

Larkey and Croft [13] used rank-based measures of performance because they were interested in interactive systems in which a rank list of codes for each document would be displayed to users. Many other applications such as automatic routing or tagging require that binary class membership decisions be made for each document as it is processed. We focus on classifier combination to enhance such classification decisions. This goal appears to be more challenging than the use of classifiers for document ranking. As an example, Hull et al. [9] found that while combination techniques were able to improve document ranking, they did considerably less well at estimating probabilities.

## 3. PROBLEM APPROACH

Our work differs from earlier combination approaches for text classification by (1) the use of expressive probabilistic dependency models to combine lower-level classifiers, leveraging special signaling variables, we refer to as reliability indicators, and (2) a focus on measures of classification performance rather than the more common consideration of ranking.

### 3.1 Reliability Indicators

Previous approaches to classifier combination have typically limited the information considered at the metalevel to the output of the classifiers [25].

We address the challenge of learning about the reliability of different classifiers in different neighborhoods of the classification domain at hand by introducing variables referred to as *reliability indicators*. A reliability indicator is an evidential distinction with states that are linked probabilistically to regions of a classification problem where a classifier performs relatively strongly or poorly.

The reliability-indicator methodology was introduced by Toyama and Horvitz [26] and applied initially to the task of combining several different machine-vision analyses in a system for identifying the head and pose of computer users. The researchers found that different visual processing modalities had distinct context-sensitive reliabilities that depended on dynamically changing details of lighting, color, and the overall configuration of the visual scene. The authors introduced reliability indicators to capture properties of the vision analyses, and of the scenes being analyzed, that appeared to provide probabilistic indications of the reliability of the output of each of the modalities. To learn probabilistic models for combining the multiple modalities, data was collected about ground truth, the observed states of indicator variables, and the outputs from the concurrent vision analyses. The data was used to construct a Bayesian network model with the ability to appropriately integrate the outputs from each of the visual modalities in real time, providing an overall higher-accuracy composite visual analysis.

The value of the indicator-variable methodology in machine vision stimulated us to explore the analogous application of the approach for representing and learning about reliability-dependent classifier contexts. For the task of combining classifiers, we formulate and include sets of variables that hold promise as being related to the performance of the underlying classifiers. We consider the states of reliability indicators and the scores of classifiers directly, and, thus, bypass the need to make ad hoc modifications to the base classifiers. This allows the metaclassifier to harness the reliability variables if they contain useful discriminatory information, and, if they do not, to fall back to using the output of the base classifiers.

As an example, consider three types of documents where: (1) the words in the document are either uninformative or strongly associated with one class; (2) the words in the document are weakly associated with several disjoint classes; or (3) the words in the document are strongly associated with several disjoint classes. Classifiers (*e.g.*, a unigram model) will sometimes demonstrate different patterns of error on these different document types. We have pursued the formulation of reliability indicators that capture different association patterns among words in documents and the structure of classes under consideration. We seek indicator variables that would allow us to learn context-sensitive reliabilities of classifiers, conditioned on the observed states of the variable in different settings.

As a concrete example, Figure 1 shows a portion of the type of combination function we can capture with the reliability-indicator methodology. The nodes on different branches of a decision tree include the values output by base classifiers as well as the values of reliability indicators for the document being classified. The decision tree provides a probabilistic context-sensitive combination rule indicated by the particular relevant branching of values of classifier scores and indicator variables. In this case, the portion of the tree displayed shows a classifier-combination function that considers thresholds on scores provided by a base-level linear SVM (*OutputOfSmox*) classifier and a base-level unigram classifier (*OutputOfUnigram*), and then uses the context established by reliability-indicator variables (*UnigramVariance* and *%FavoringInClassAfterFS*) to make a final decision about a classification. The annotations in the figure show the threshold tests that are being performed, the number of examples in the training set that satisfied the test, and a graphical representation of the probability distribution at the leaves. The likelihood of class membership is indicated by the length of the rectangle at the leaves of the tree.

The variable *UnigramVariance* represents the variance of unigram weights for words present in the current document. The intuition is that the unigram classifier would be accurate when there is low variance in weights. The variable *%FavoringInClassAfterFS* is the percentage of words (after feature selection) that occur more often in documents within a target class than in other classes. Classifiers that weight positive and negative evidence differently should be distinguished by this variable. For the *Strive-D (norm)* classifier excerpt shown in Figure 1 we have further normalized the metafeatures to have zero mean and unit standard deviation so most values fall between -1 and 1 as a result.

The indicator variables used in our studies are an intuitive attempt at formulating states to represent influential contexts. We defined variables to represent a variety of contexts that showed promise as being predictive of accuracy—*e.g.*, the number of features present in a document before and after feature selection, the distribution of features across the positive vs. negative classes, and the mean and variance of classifier-specific weights.

Currently, our indicator variables roughly fall into of four types. Different indicator variables measure (1) the amount of information present in the original document, (2) the information lost or mismatch between the representation the classifier used and the original document, (3) the sensitivity of decision to evidence shift, and (4) some basic voting statistics. *DocumentLength* is an example of type 1. The performance of classifiers is sometimes correlated with document length, because longer documents give more information to use in making a classification. *DocumentLength* can also be informative since some classifiers will perform poorly over longer documents because they do not model length (*e.g.* they double count evidence and longer documents are more likely to deviate from a correct determination). *PercentRemoved* is an example of type 2. This represents the percent of features removed by feature selection. If most of the document was not represented by the feature set used in a classifier, then some classifiers may be unreliable. Others (*e.g.* decision trees that model missing attributes) may continue to be reliable. When the base classifiers are allowed to use different representations, then these features can play an even more important role. An example of type 3 is the *UnigramVariance* variable. Low variance means the decision of the classifier is unlikely to change with a small change in the document content; high variance increases the chances that the decision would change with only a small change in document content. Finally, *NumVotingForClass* or *PercentAgreement* are examples of the last type. These simple voting statistics improve the metaclassifier search space (since the metaclassifier is given the base classifier decisions as input as well). For a 2-class case the *PercentAgreement* variable may provide little extra information but for n-classes it can be used to determine if the base classifiers have fractured their votes among a small number of classes or across a wide array. We found all of these types to be useful in the final combination scheme, and the early analysis has not found that any one type dominates the combination models.

Reliability-indicator variables are qualitatively different than variables representing the output of classifiers because we do not assume that the reliability indicators have some threshold point that classifies the examples better than random, nor do we assume that classification confidence shows monotonicity trends as in classifiers. We currently do not exploit these underlying assumptions for the classifiers, but we hope to do so in the future. At a more practical level, the reliability indicators are usually much simpler than typical classification algorithms.

## 3.2 Strive: Metaclassifier with Reliability Indicators

We refer to our classifier combination learning and inference framework as *Strive* for **St**acked **R**eliability **I**ndicator **V**ariable **E**nsemble. We select this name because the approach can be viewed as essentially extending the stacking framework by introducing reliability indicators at the metalevel. The Strive architecture is depicted in Figure 2.

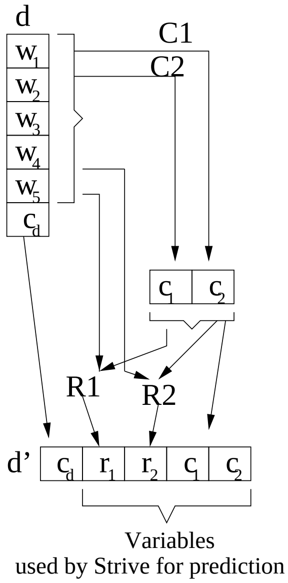

**Figure 2: The basic architecture of Strive.**

We map the original stacking learning problem to a new learning problem. Originally, we have some document, $d$, with words, $w_1, \ldots, w_n$ and class, $c_d$. Our classifiers, $C_i$, predict $c_d$ from the words. We denote their predictions by $c_i$. Then, our set of reliability indicators, $R_i$, use the words and the classifier outputs to generate their values, $r_i$. We now have a new document, $d' =< r_1, \ldots, r_j, c_1, \ldots, c_k >$ with class $c_d$ which we will give to the metaclassifier as the learning problem.

We require the outputs of the classifiers to train the metaclassifier. Thus, we perform cross-validation over the training data, and use the values obtained while an example serves as a validation item as the input to the metaclassifier.

## 3.3 BestSelect Classifier

When classifier combination is formulated as the process of *selecting* on a per-example basis the best base classifier, we can introduce a natural upper bound on combination. Such an upper bound can be useful as a benchmark in experiments with classifier combination procedures.

To classify a given document, if any of the classifiers correctly predict that document, the best combination would select any of the correct classifiers. Thus, such a classification combination errs only when all of the base classifiers are incorrect. We refer to this classifier as the BestSelect classifier. If all of the base classifiers are better than random, the BestSelect gives us a rough idea of the best we can do combining them in a selection framework.

We note that we are not using a pure selection approach, as our framework allows the possibility of choosing a class that none of the base classifiers predicted. In cases where the classifiers are not better than random (or are logically dependent), such an upper bound may be uninformative. If we simply seek an answer to the question, “How much of what we could optimistically expect to gain with this set of base classifiers have we gained?”, then this loose bound provides a more pessimistic view of the results than is actually the case. Therefore, we evaluate the scores of the BestSelect classifier as a useful point of reference in our experiments.

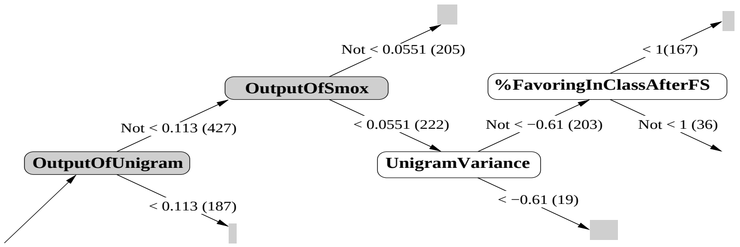

**Figure 1: Portion of decision tree, learned by *Strive-D (norm)* for the *Travel & Vacations* class in the MSN Web Directory corpus, representing a combination policy at the metalevel that considers scores output by classifiers (dark nodes) and values of indicator variables (lighter nodes).**

# 4. EXPERIMENTAL ANALYSIS

We engaged in a large number of experiments to test the value of probabilistic classifier combination with reliability indicator variables. We shall describe the corpora, methodology, and results.

## 4.1 Data

We examined several corpora, including the *MSN Web Directory*, *Reuters*, and *TREC-AP*.

### *4.1.1 MSN Web Directory*

The MSN Web Directory is a large collection of heterogeneous web pages (from a May 1999 web snapshot) that have been hierarchically classified. We used the same train/test split of 50078/10024 documents as that reported in [6].

The MSN Web hierarchy is a 7-level hierarchy; we used all 13 of the top-level categories. The class proportions in the training set vary from $1.15\%$ to $22.29\%$. In the testing set, they range from $1.14\%$ to $21.54\%$. The classes are general subject categories such as *Health & Fitness* and *Travel & Vacation*. Human indexers assign the documents to zero or more categories.

For the experiments below, we used only the top 1000 words with highest mutual information for each class; approximately 195K words appear in at least 3 training documents.

### *4.1.2 Reuters*

The Reuters 21578 corpus [15] contains Reuters news articles from 1987. For this data set, we used the ModApte standard train/ test split of 9603/3299 documents (8676 unused documents). The classes are economic subjects (*e.g.*, “acq” for acquisitions, “earn” for earnings, etc.) that human indexers decided applied to the document; a document may have multiple subjects. There are actually 135 classes in this domain (only 90 of which occur in the training and testing set); however, we only examined the 10 most frequent classes since small numbers of training examples makes estimating some performance measures unreliable due to high variance. Limiting to the 10 largest classes allows us to compare our results to previously published results [7, 10, 19, 20].

The class proportions in the training set vary from $1.88\%$ to $29.96\%$. In the testing set, they range from $1.7\%$ to $32.95\%$.

For the experiments below we used only the top 300 words with highest mutual information for each class; approximately 15K words appear in at least 3 training documents.

### *4.1.3 TREC-AP*

The TREC-AP corpus is a collection of AP news stories from 1988 to 1990. We used the same train/test split of 142791/66992 documents that was used in [17]. As described in [16] (see also [14]), the categories are defined by keywords in a keyword field. The title and body fields are used in the experiments below. There are 20 categories total.

The frequencies of the 20 classes are the same as those reported in [17]. The class proportions in the training set vary from $0.06\%$ to $2.03\%$. In the testing set, they range from $0.03\%$ to $4.32\%$.

For the experiments described below, we use only the top 1000 words with the highest mutual information for each class; approximately 123K words appear in at least 3 training documents.

## 4.2 Classifiers

We employed several base-level classifiers and classifier combination methods in our comparative studies. We review the classifiers and combination methods here.

### *4.2.1 Base Classifiers*

We worked to keep the representations for the base classifiers analyzed in our experiments nearly identical so as to isolate the benefits gained solely from the probabilistic combination of classifiers with reliability indicators. We would expect that varying the representations (*i.e.*, using different feature selection methods or document representations) would only improve the performance as this would likely decorrelate the performance of the base classifiers.

We selected four classifiers that have been used traditionally for text classification: decision trees, linear SVMs, naïve Bayes, and a unigram classifier.

For the decision-tree implementation, we employed the WinMine decision networks toolkit and refer to this as *Dnet* below [5]. Dnet builds decision trees using a Bayesian machine learning algorithm [4, 8]. While this toolkit is targeted primarily at building models that provide probability estimates, we found that Dnet models usually perform acceptably on error rate. However, we found that the performance of Dnet with regard to other measures is sometimes poor.

For linear SVMs, we use the *Smox* toolkit which is based on Platt’s **S**equential **M**inimal **O**ptimization algorithm. After experimenting with a binary and a continuous model, we used a continuous model as it seemed to perform at approximately the same level.

The *naïve Bayes* classifier has also been referred to as a multivariate Bernoulli model. In using this classifier, we smoothed word and class probabilities using a Bayesian estimate (with the word prior) and a Laplace m-estimate, respectively.

The *unigram* classifier uses probability estimates from a unigram language model. This classifier has also been referred to as a multinomial naïve Bayes classifier. Probability estimates are smoothed in a similar fashion to smoothing in naïve Bayes classifier.

#### 4.2.2 Basic Combination Methods

We performed experiments to explore a variety of combination methods. We considered several different combination procedures. The first combination method is based on selecting one classifier for each binary class problem, based on the one that performed best for a validation set. We refer to this method as the *Best By Class* method.

Another combination method is based on taking a majority vote of the base classifiers. This approach is perhaps the most popular combination methodology. When performing a majority vote, ties can be broken in a variety of ways (*e.g.*, breaking ties by always voting for *in class*). We experimented with several variants of this method, but we only present results here for the method which relies on breaking ties by voting with the *Best By Class* classifier because it nearly always outperformed the other majority vote methods. We refer to this method as *Majority BBC*.

#### 4.2.3 Hierarchical Combination Methods

*Stacking*

Finally, we investigate several variants of the hierarchical models described earlier. As mentioned above, omitting the reliability-indicator variables is equivalent to stacking [25, 29]. We refer to these classifiers below as *Stack-X* where *X* is replaced by the first letter of the classifier that is performing the metaclassification. Therefore, *Stack-D* uses a decision tree as the metaclassifier, and *Stack-S* uses a linear SVM as the metaclassifier. It should be noted that *Stack-S* is also a weighted linear combination method since it is based on a linear SVM and uses only the classifier outputs.

It can be problematic to learn the weights for an SVM when the inputs have vastly different scales (in addition it may not be possible to pick good weights); therefore, we normalize the inputs to the metaclassifiers to zero mean and unit standard deviation. In order to perform consistent comparisons, we perform the same alteration for the metaclassifiers using Dnet. We also give for one of the Dnet variants the results without performing normalization; as would be expected the impact of normalization for decision-tree learners is relatively minimal (and has both positive and negative influences).

*Strive*

Similar to the notation described above, we add a letter to Strive to denote the metaclassifier being used. So, *Strive-D* is the Strive framework using Dnet as a metaclassifier. For comparison to the stacking methods, we evaluate *Strive-D* and *Strive-S*. Normalization is noted in the same way.

The experiments reported here use a total of 49 reliability indicators (including those specific examples given in Section 3.1). These variables were simply our initial pass at representing appropriate information. In the future, we intend to publish an analysis of which variables are most useful, in addition to extending the set of variables currently employed.

### 4.3 Performance Measures

To compare the performance of the classification methods we look at a set of standard performance measures. The F1 measure [27, 31] is the harmonic mean of precision and recall where $Precision = \frac{Correct\ Positives}{Predicted\ Positives}$ and $Recall = \frac{Correct\ Positives}{Actual\ Positives}$. We can often assess a cost function in classification settings that can be described as $C(FP, FN) = FP * P(FalsePos) + FN * P(FalseNeg)$ where $FP$ is the cost of a false positive classification and $FN$ is the cost of a false negative classification. The most commonly used function in the literature is the error rate which is $FP = FN = 1$. However, the importance of varying cost functions has been recognized by many researchers because applications rarely have equal costs for different types of errors [21]. In order to assess how sensitive performance is to the utility measure, we considered results for $C(10, 1)$ and $C(1, 10)$.

In addition, we computed and displayed a receiver operating characteristic (ROC) curve, which represents the performance of a classifier under any linear utility function [21]. We report results on the area under the ROC curve as an attempt to summarize[1] the linear utility space of functions.

For each performance measure, we can either macro-average or micro-average. Macro-averaging, which we present here, is computed separately for each class and then arithmetically averaged; this tends to weight rare classes more heavily. Micro-averaged values are computed directly from the binary decisions over all classes; this places more weight on the common classes. We evaluated the systems with both macro and micro averaging, but because of space limitations, we simply note that our analysis is consistent with the micro results and omit presentation of the full analysis.

### 4.4 Experimental Methodology

As the categories under consideration in the experiments are not mutually exclusive, the classification was done by training $n$ binary classifiers, where $n$ is the number of classes.

Decision thresholds for each classifier were set by optimizing them for each performance measure over the validation data; that is, a classifier could have different thresholds for each of the separate performance measures (and for each class). This ensures that the base classifiers are as competitive as possible across the various measures.

To generate the data for training the metaclassifier, (*i.e.*, reliability indicators, classifier outputs, and class labels), we used five-fold cross-validation on the training data from each of the corpora. The data set obtained through this process was then used to train the metaclassifiers. Finally, the resulting metaclassifiers were applied to the separate testing data described above.

### 4.5 Results

Tables 1, 2, and 3, present the performance results over the three corpora. In terms of the various performance measures, better performance is indicated by larger F1 or ROC area values or by smaller $C(FP, FN)$ values. The best performance (ignoring BestSelect) in each column is given in bold.

To determine statistical significance for the macro-averaged measures, a one-sided macro sign test and macro t-test was performed [31]. Differences with a p-level above 0.05 were not considered statistically significant.

We do not explicitly report significance results for the t-test comparisons; instead, our analysis follows the macro sign test which yielded more conservative comparisons (*i.e.*, the t-test primarily just increased the number of differences found to be significant in the tables).

The classifier combinations are annotated to indicate the results of a macro sign test. A † indicates the method significantly outperforms (at the 0.05 level) the best base classifier. In addition, on the variants of Stack and Strive, a ‡ indicates the method outperforms the basic combination methods. Results for the remaining sign test comparisons are omitted.

### 4.6 Discussion

First, we note that the base classifiers are competitive and consistent with the previously reported results over these corpora [6,

[1] We note that the area under the curve is not a precise summary of the linear utility space.

| Method | Macro F1 | Error | C(1,10) | C(10,1) | ROC Area |
|---|---|---|---|---|---|
| Dnet | 0.5502 | 0.0583 | 0.3023 | 0.0771 | 0.8812 |
| Smox | 0.6705 | 0.0455 | 0.2239 | 0.0799 | 0.9125 |
| Naïve Bayes | 0.5527 | 0.0649 | 0.2853 | 0.0798 | 0.8915 |
| Unigram | 0.5982 | 0.0594 | 0.2589 | 0.0812 | 0.9003 |
| Best By Class | 0.6705 | 0.0455 | 0.2236 | 0.0783 | N/A |
| Majority BBC | 0.6668 | 0.0476 | 0.2173$^{\dagger}$ | 0.0761 | N/A |
| Stack-D (norm) | 0.6775 | 0.0446$^{\dagger\ddagger}$ | 0.2118$^{\dagger}$ | 0.0784 | 0.9292$^{\dagger}$ |
| Stack-S (norm) | 0.6732 | 0.0450 | 0.2174$^{\dagger}$ | 0.0757 | 0.9210$^{\dagger}$ |
| Strive-D | 0.6877$^{\dagger\ddagger}$ | **0.0429**$^{\dagger\ddagger}$ | **0.1939**$^{\dagger\ddagger}$ | 0.0742 | 0.9383$^{\dagger}$ |
| Strive-D (norm) | 0.6908$^{\dagger\ddagger}$ | 0.0434 | 0.1949$^{\dagger\ddagger}$ | 0.0742 | **0.9398**$^{\dagger}$ |
| Strive-S (norm) | **0.6948**$^{\dagger\ddagger}$ | 0.0430$^{\dagger\ddagger}$ | 0.2037$^{\dagger}$ | **0.0712** | 0.9114 |
| Strive-D (norm, omit Smox) | 0.6670 | 0.0464 | 0.2062$^{\dagger\ddagger}$ | 0.0754 | 0.9361$^{\dagger}$ |
| BestSelect | 0.8365 | 0.0270 | 0.0905 | 0.0616 | N/A |

**Table 1: Performance on MSN Web Directory Corpus**

| Method | Macro F1 | Error | C(1,10) | C(10,1) | ROC |
|---|---|---|---|---|---|
| Dnet | 0.7846 | 0.0242 | 0.0799 | 0.0537 | 0.9804 |
| Smox | 0.8480 | 0.0157 | 0.0580 | 0.0390 | 0.9815 |
| Naïve Bayes | 0.6574 | 0.0320 | 0.1423 | 0.0527 | 0.9703 |
| Unigram | 0.7645 | 0.0234 | 0.0713 | 0.0476 | 0.9877 |
| Best By Class | 0.8592 | 0.0153 | 0.0518 | 0.0409 | N/A |
| Majority BBC | 0.8524 | 0.0160 | 0.0448$^{\dagger}$ | 0.0446 | N/A |
| Stack-D (norm) | 0.8636$^{\dagger}$ | 0.0153 | 0.0449 | 0.0392 | 0.9893 |
| Stack-S (norm) | 0.8720$^{\dagger\ddagger}$ | 0.0143$^{\dagger}$ | 0.0445$^{\dagger}$ | 0.0365 | 0.9930$^{\dagger}$ |
| Strive-D | 0.8547 | 0.0154 | 0.0472 | 0.0358 | 0.9903 |
| Strive-D (norm) | 0.8526 | 0.0157 | 0.0468 | 0.0359 | 0.9897 |
| Strive-S (norm) | **0.8749**$^{\dagger}$ | **0.0125**$^{\dagger\ddagger}$ | **0.0382**$^{\dagger}$ | **0.0353** | **0.9939** |
| Strive-D (norm, omit Smox) | 0.8433 | 0.0168 | 0.0484 | 0.0425 | 0.9900 |
| BestSelect | 0.9529 | 0.0050 | 0.0100 | 0.0202 | N/A |

**Table 2: Performance on Reuters Corpus**

7, 10, 14, 16, 19]. The results reported for Reuters are not directly comparable to those reported by Yang & Liu [31] as they report results over all 90 classes and do not give a breakdown for the 10 most frequent categories. Furthermore, the fact that the linear SVM Smox tends to be the best base classifier is consistent with the literature [7, 10, 31].

### *MSN Web Directory*

Examining the results for the MSN Web Directory corpus in table 1 reveals several points. First, the basic combiners have only one significant win, C(1,10) for the Majority BBC approach. The results directly support the idea that a very good learner (Smox) tends to be brought down when combined via a majority vote scheme with weak learners; in addition, the win most likely results from the fact that the base learners (other than Smox) have a tendency to predict positive. When false negatives are weighed more heavily, the shift toward predicting positive helps reduce the number of false negatives.

Next, we see that Stacking posts several significant wins and appears to have some advantages over the base classifiers. However, the Stacking combination shows little significant improvement over the basic combination methods.

*Strive-D* and *Strive-S (norm)* show advantages that are robust across a variety of performance measures. Each shows a small (about 5% error reduction) but consistent improvement across a variety of performance measures. When viewed in terms of the approximate ceiling as established by the BestSelect model, the error reduction provided by the Strive combination methods is an even greater portion of the total possible reduction.

As can be inferred from the sign tests, these results are very consistent across classes. For example, on ROC area, *Strive-D* beats the base classifiers and basic combiners on 13/13 classes, and it beats the stacking methods on 12/13 classes. The notable exception is the performance of *Strive-S (norm)* on ROC area; graphical inspection of the ROC curves suggests this result is likely based on too much weight being placed on the strong classifier for a curve.

Often, there is a crossover in the ROC curve between two of the base classifiers further out on the false-positives axis. Most utility measures in practice correspond to the early part of the curve (this depends on the particular features of the given curve). Smox as a metaclassifier sometimes seems to lock onto the classifier that is strong in the early portion of the curve and loses out on the later part of the curve. Since this portion of the curve rarely matters, one could consider using an abbreviated version of curve area to assess systems.

In Figure 3, we can see that the two Strive variants dominate the

| Method | Macro F1 | Error | C(1,10) | C(10,1) | ROC |
|---|---|---|---|---|---|
| Dnet | 0.6015 | 0.0065 | 0.0342 | 0.0079 | 0.9768 |
| Smox | 0.7335 | 0.0049 | 0.0294 | 0.0077 | 0.9691 |
| Naïve Bayes | 0.5676 | 0.0065 | 0.0455 | 0.0078 | 0.9755 |
| Unigram | 0.6001 | 0.0064 | 0.0347 | 0.0079 | 0.9819 |
| Best By Class | 0.7335 | 0.0049 | 0.0294 | 0.0077 | N/A |
| Majority BBC | 0.7145 | 0.0056 | 0.0292 | 0.0075 | N/A |
| Stack-D (norm) | 0.7357 | 0.0049 | **0.0241**$^{\dagger\ddagger}$ | 0.0086 | 0.9856 |
| Stack-S (norm) | 0.7351 | 0.0049 | 0.0288 | **0.0075** | 0.9656 |
| Strive-D | 0.7325 | **0.0048** | 0.0276 | 0.0078 | 0.9858 |
| Strive-D (norm) | 0.7280 | 0.0049 | 0.0271 | 0.0077 | 0.9855 |
| Strive-S (norm) | **0.7431** | 0.0048 | 0.0295 | 0.0076 | 0.9634 |
| Strive-D (norm, omit Smox) | 0.6938 | 0.0055 | 0.0313 | 0.0077 | **0.9858** |
| BestSelect | 0.8763 | 0.0034 | 0.0149 | 0.0062 | N/A |

**Table 3: Performance on TREC-AP Corpus**

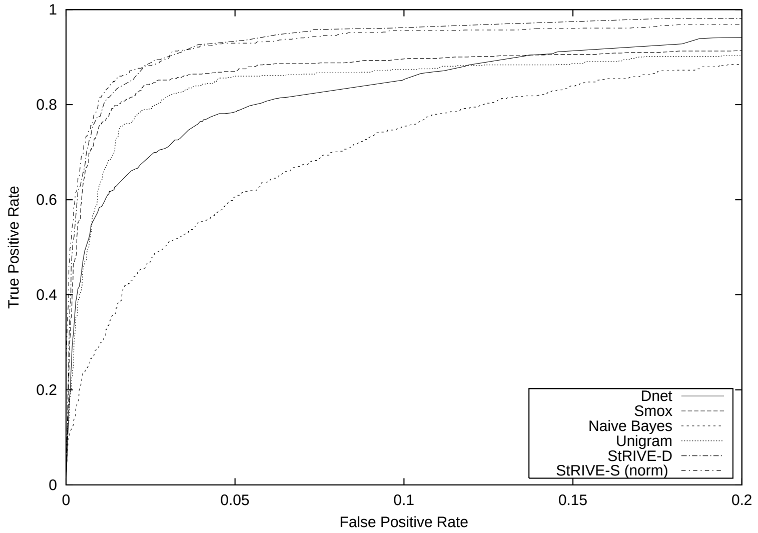

**Figure 3: The ROC curve for the *Health & Fitness* class in the MSN Web Directory corpus**

four base classifiers. In fact, *Strive-D* dominates (*i.e.*, its quality is greater than any other curve at every point) most of the MSN Web Directory corpus. We also can see (note the truncated scale) the base classifiers catching up with *Strive-S (norm)* on the right side of the curve. They, in fact, do surpass it. As a result, *Strive-D* is usually a more appropriate choice if the utility function penalizes false negatives significantly more heavily than false positives.

Referring back to Figure 1, we can understand why the decision tree is more appropriate for tracking crossovers in some cases. In this case, it is establishing a score region for Smox and a score region for Dnet where the reliability indicators give further information about how to classify an example.

Finally, we note that when we omit the base classifier Smox in *Strive-D (norm, omit Smox)*, the resulting combination improves by a large margin over the base methods; however, the resulting classifier generally still fails to beat Smox. This suggests that there are not enough indicator variables tied to Smox's behavior, or alternatively, that the other classifiers as a group behave like Smox, rather than classify in a complementary fashion.

### *Reuters and TREC-AP*

The results for Reuters and TREC-AP in Tables 2 and 3 are consistent with the above analysis. We note that the level of improvement tends to be less pronounced for these corpora.

## 5. FUTURE WORK

We are excited about the opportunities for probabilistic combination of multiple classifiers with reliability indicators. We are pursuing several research directions. Foremost, we believe that a functional search that generates and tests a larger number of reliability indicators could provide valuable sets of informative reliability indicators.

In the experiments above, the classifiers were specifically held constant in order to distill the effect of indicator variables. In future studies, we will allow representations to vary to induce more variety among the base classifiers.

We are also interested in exploring the use of other classifiers as metaclassifiers. The metaclassifier should be a classifier that handles correlated input well (*e.g.*, use of maximum entropy) as classifiers performing better than random will be necessarily correlated.

# 6. SUMMARY AND CONCLUSIONS

We reviewed a methodology for building a metaclassifier for text documents that centers on combining multiple distinct classifiers with probabilistic learning and inference that leverages reliability-indicator variables. Reliability indicators provide information about the context-sensitive nature of classifier reliability. We reviewed several popular text classification methods, and described several combination schemes. We introduced the *Strive* methodology that uses reliability indicators in a hierarchical combination model and reviewed comparative studies comparing Strive with other combination mechanisms. The empirical evaluations support the conclusion that a simple majority vote in situations where one of the classifiers performs strongly can "water down" that classifier's performance. The experiments also show that stacking and *Strive* seem to provide robust combination schemes across a variety of performance measures.

## Acknowledgments

We thank Max Chickering and Robert Rounthwaite for their special support of the WinMine toolkit, and John Platt for advice and code support for the linear SVM classifier.

# 7. REFERENCES

[1] K. Al-Kofahi, A. Tyrrell, A. Vacher, T. Travers, and P. Jackson. Combining multiple classifiers for text categorization. In *CIKM '01*, pages 97–104, 2001.

[2] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *SIGIR '94*, pages 173–181, 1994.

[3] N. Belkin, C. Cool, W. Croft, and J. Callan. The effect of multiple query representations on information retrieval system performance. In *SIGIR '93*, pages 339–346, 1993.

[4] D. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *UAI '97*, pages 80–89, 1997.

[5] M. Corporation. WinMine Toolkit v1.0. http://research.microsoft.com/˜dmax /WinMine/ContactInfo.html, 2001.

[6] S. T. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR '00*, pages 256–263, 2000.

[7] S. T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98*, pages 148–155, 1998.

[8] D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

[9] D. Hull, J. Pedersen, and H. Schuetze. Method combination for document filtering. In *SIGIR '96*, pages 279–287, 1996.

[10] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML '98*, pages 137–142, 1998.

[11] J. Katzer, M. McGill, J. Tessier, W. Frakes, and P. DasGupta. A study of the overlap among document representations. *Information Technology: Research and Development*, 1:261–274, 1982.

[12] W. Lam and K.-Y. Lai. A meta-learning approach for text categorization. In *SIGIR '01*, pages 303–309, 2001.

[13] L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In *SIGIR '96*, pages 289–297, 1996.

[14] D. D. Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. *SIGIR Forum*, 29(2):13–19, Fall 1995.

[15] D. D. Lewis. Reuters-21578, distribution 1.0. http://www.daviddlewis.com/resources /testcollections/reuters21578, January 1997.

[16] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR '94*, pages 3–12, 1994.

[17] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *SIGIR '96*, pages 298–306, 1996.

[18] Y. Li and A. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[19] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Working Notes of AAAI 1998, Workshop on Learning for Text Categorization*, pages 41–48, 1998.

[20] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[21] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42:203–231, 2001.

[22] T. Rajashekar and W. Croft. Combining automatic and manual index representations in probabilistic retrieval. *Journal of the American Society for Information Science*, 6(4):272–283, 1995.

[23] R. E. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39:135–168, 2000.

[24] J. Shaw and E. Fox. Combination of multiple searches. In D. K. Harman, editor, *TREC-3 Conference*, number 500-225 in NIST Special Publication, pages 105–108, 1995.

[25] K. Ting and I. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[26] K. Toyama and E. Horvitz. Bayesian modality fusion: Probabilistic integration of multiple vision algorithms for head tracking. In *ACCV 2000, Fourth Asian Conference on Computer Vision*, 2000.

[27] C. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[28] S. Weiss, C. Apte, F. Damerau, D. Johnson, F. Oles, T. Goets, and T. Hampp. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14(4), 1999.

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

[30] Y. Yang, T. Ault, and T. Pierce. Combining multiple learning strategies for effective cross validation. In *ICML '00*, pages 1167–1182, 2000.

[31] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99*, pages 42–49, 1999.