Information Retrieval, 8, 67–100, 2005

The Combination of Text Classifiers Using Reliability Indicators*

PAUL N. BENNETT pbennett+@cs.cmu.edu
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

SUSAN T. DUMAIS sdumais@microsoft.com
ERIC HORVITZ horvitz@microsoft.com
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

Received March 13, 2003; Revised December 30, 2003; Accepted January 6, 2004

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 signals 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.

Keywords: text classification, classifier combination, metaclassifiers, feature selection, 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. Studies of classifier combination have been motivated primarily by the intuition that overlaying classifiers, that work in related but qualitatively different ways, could leverage the distinct strengths of each method.

Classifiers can be combined in a variety of ways. In one approach, a text classifier is composed from multiple distinct classifiers by selecting the best classifier to use in different situations or contexts. For example, we may perform analytical or empirical studies to identify the most accurate text classifier in some setting, seeking to learn about accuracy over output scores or some combination of output scores and features considered in the 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 the best classification. In a finer-grained approach to combining multiple classifiers, the scores

*This paper revises and extends material originally presented by Bennett et al. (2002).

generated by the contributing classifiers are taken as inputs to the combination function. Whichever approach to combination is 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 the discriminatory context for learning about reliability. We borrow the reliability-indicator methodology from work presented initially by Toyama and Horvitz (2000) in the context of automated vision. 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 text classification 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 can 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 metaclassifiers that improve text classification 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

Appropriately combining information sources to form a more effective output than any of the individual sources is a problem that has been investigated in many fields. The challenges of integrating information have gone under the labels of diagnosis (Horvitz et al. 1988), pattern recognition (Duda et al. 2001), sensor fusion (Klein 1999), distributed data mining (Kargupta and Chan 2000), and a variety of ensemble methods (Dietterich 2000). Diagnosis centers on identifying disorders from multiple pieces of evidence, such as reasoning about probability distributions over a patient's diseases from a set of symptoms and test results. Pattern recognition and sensor fusion typically address challenges with integrating information from multiple modalities (e.g., auditory and visual) while distributed data mining addresses how results retrieved from distinct training data sets can be unified to provide one coherent view to the user. Ensemble methods first solve a classification or regression problem by creating multiple learners that each attempt to solve the task independently, then use the procedure specified by the particular ensemble method for selecting or weighting the individual learners. Ensemble methods include such techniques as Bayesian averaging, bagging, boosting, stacking, cascade generalization, hierarchical mixture of experts, and the work presented in this paper.

Text classification addresses the task of labeling a text document with one or more labels from a set of predefined content-based categories. These categories may be primary document topics (e.g., *Health & Fitness*, *Business & Finance*, etc.), hierarchical indices of technical content (e.g., medical hierarchies (Hersh et al. 1994)), genres (e.g., legal, fiction, etc. (Kessler et al. 1997)), or a variety of other distinctions (e.g.., *normal e-mail* vs. *junk e-mail*, as used by Sahami et al. (1998), or *urgent e-mail* vs. *non-urgent e-mail*, as explored by Horvitz et al. (1999)). Text classification methods can thus provide a backend for many information retrieval tasks, e.g., routing, tagging and filtering. The interested reader should see Sebastiani (2002) for a broad survey of recent applications of machine learning to text classification.

The overlaying of multiple methodologies or representations has been employed in several areas of information retrieval. For example, previous research in information retrieval has demonstrated that retrieval effectiveness can be improved by using multiple, distinct representations (Bartell et al. 1994, Katzer et al. 1982, Rajashekar and Croft 1995), or by using multiple queries or search strategies (Belkin et al. 1993, Shaw and Fox 1995). In the realm of text classification, several researchers have achieved improvements in classification accuracy by combining different classifiers (Al-Kofahi et al. 2001, Hull et al. 1996, Larkey and Croft 1996, Li and Jain 1998, Yang et al. 2000). Similarly, several investigators have enhanced classification performance by applying many instances of the same classifier, such as boosting procedures (Schapire and Singer 2000, Weiss et al. 1999).

Much of the previous work on combining text classifiers has centered on the use of basic policies for selecting the best classifier or for combining the output of multiple classifiers. As some examples, Larkey and Croft (1996) used weighted linear combinations of system ranks or scores; Hull et al. (1996) used linear combinations of probabilities or log odds scores; Yang et al. (2000) used a linear combination of normalized scores; Li and Jain (1998) used voting and classifier selection techniques; and Lam and Lai (2001) used category-averaged features to pick a (potentially different) classifier to use for each category.

Larkey and Croft (1996) 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 is more challenging than the use of classifiers for document ranking. As an example, Hull et al. (1996) found that, although combination techniques were able to improve document ranking, they did considerably less well at estimating probabilities required for online classification decisions.

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 (1999) in stacked generalization and Gama (1998a, 1998b) in cascade generalization, although they did not apply their approach to text problems. We also report baseline comparisons with voting and classifier-selection techniques.

3. Problem approach

Our work is distinguished 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, referred 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 (Ting and Witten 1999) and/or the original feature space (Gama 1998a). Since a classifier rarely is the best choice across a whole domain, an intuitive alternative is to identify the document-specific context that differentiates between regions where a base classifier has higher or lower reliability.

Figure 1 shows an example using four base classifiers: decision tree, SVM, naïve Bayes, and unigram. When given a test document as input, each of the four base classifiers outputs a probability distribution over possible class labels (depicted graphically as a histogram in the figure). The metaclassifier uses this information along with document context (to be described in more detail) to produce a final classification of the document.

We address the challenge of learning about the reliability of different classifiers in different neighborhoods of the classification domain by introducing variables referred to as *reliability indicators* which represent the analytic "context" of a specific document. 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.

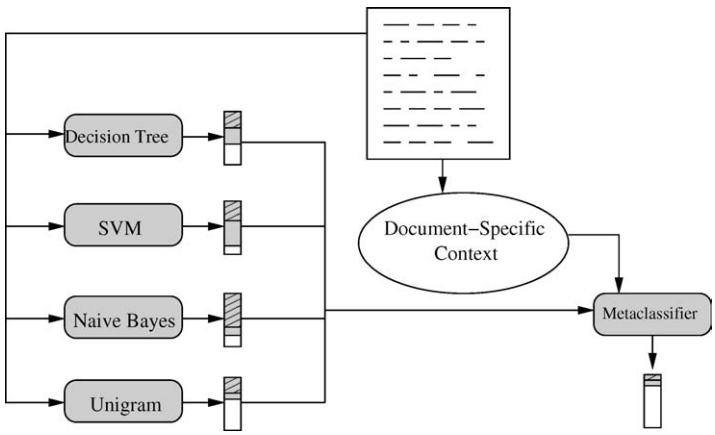

Figure 1. Schematic characterization of reliability-indicator methodology. The methodology formalizes the intuition shown here that document-specific context can be used to improve the performance of a set of base classifiers. The output of the classifiers is a graphical representation of a distribution over possible class labels.

The reliability-indicator methodology was introduced by Toyama and Horvitz (2000) and applied initially to the task of combining, in a probabilistically coherent manner, several distinct machine-vision analyses in a system for tracking 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 provided 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 approach for representing and learning about reliability-dependent contexts in text classification problems. 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 in a graceful manner 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. If we can characterize a document as belonging to one of these model-specific failure types, then we can assign the appropriate weight to the classifier's output for this kind of document. 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.

To highlight the approach with a concrete example, figure 2 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

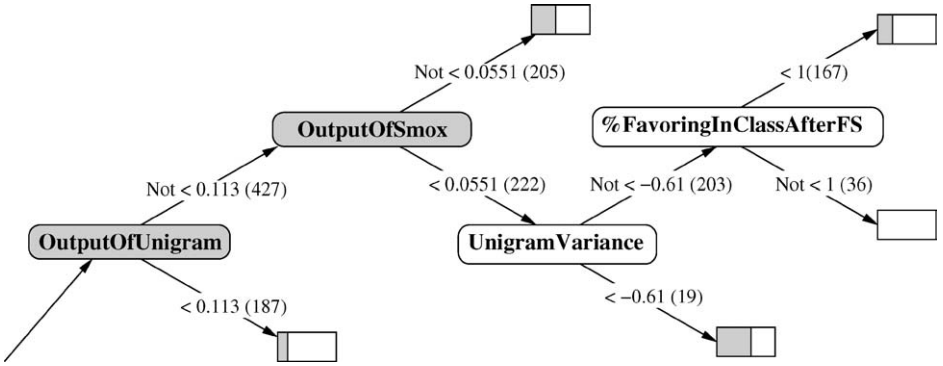

Figure 2. Portion of decision tree, learned by *STRIVE-D* (*norm*) for the *Business & Finance* class in the MSN Web Directory corpus, representing a combination policy at the metalevel that considers scores output by classifiers (dark nodes) and values of indicator variables (lighter nodes).

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 satisfy 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 bars at the leaves of the tree.

The variable *UnigramVariance* represents the variance of unigram weights for words present in the current document. The intuition behind the formulation of this reliability-indicator variable is that the unigram classifier would show a tendency toward higher accuracies 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. Appendix A gives further details about the reliability indicators used in these experiments.

The indicator variables used in our studies represent an attempt to formulate states that capture influential contexts. We constructed variables to represent a variety of contexts that held promise as being predictive of accuracy. These include such variables as 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.

We can broadly group reliability-indicator variables into one of four types, including variables that measure (1) the amount of information present in the original document, (2) the information loss or mismatch between the representation used by a classifier and the original document, (3) the sensitivity of the decision to evidence shift, and (4) some basic voting statistics.

DocumentLength is an example of a reliability-indicator variable 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 because some classifiers will perform poorly over longer documents as they do not model the influence of document length on classification performance (e.g., they double count evidence and longer documents are more likely to deviate from a correct determination).

PercentRemoved serves as an example of type 2. This variable represents the percent of features removed in the process of feature selection. If most of the document was not represented by the feature set employed by a classifier, then some classifiers may be unreliable. Others classifiers (e.g., decision trees that model missing attributes) may continue to be reliable. When the base classifiers are allowed to use different representations, type 2 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 the document.

Finally, *NumVotingForClass* or *PercentAgreement* are examples of type 4 reliability indicators. These simple voting statistics improve the metaclassifier search space (since the metaclassifier is given the base classifier decisions as input as well). For a two-class case the *PercentAgreement* variable may provide little extra information but for greater number of 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 four types of reliability indicators to be useful in the final combination scheme, and preliminary analyses did not indicate that any one type dominates in the combination models.

Beyond the key difference in the semantics of their usage, reliability-indicator variables differ qualitatively from variables representing the output of classifiers in several ways. For one, we do not assume that the reliability indicators have some threshold point that classifies the examples better than random. We also do not assume that classification confidence shows monotonicity trends as in classifiers.

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 graphically in figure 4.

Our methodology maps the original classification task into a new learning problem. In the original learning problem (figure 3), the base classifiers simply predict the class from a word-based representation of the document, or more generally, each base classifier outputs a distribution (possibly unnormalized) over class labels. *STRIVE* adds another layer of learning to the base problem. A set of reliability-indicator functions use the words in the document and the classifier outputs to generate the reliability indicator values, r_i, for a particular document. This process can be viewed as yielding a new representation of the document that consists of the values of the reliability indicators, as well as the outputs of the base classifiers. The metaclassifier uses this new representation for learning and classification. This enables the metaclassifier to employ a model that uses the output of the

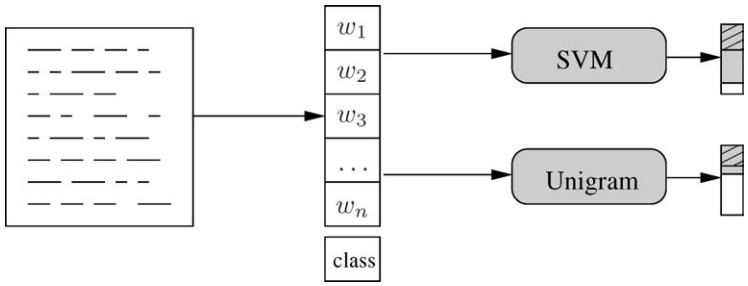

Figure 3. Typical application of a classifier to a text problem. In traditional text classification, a word-based representation of a document is extracted (along with the class label during the learning phase), and the classifiers (here an SVM and Unigram classifier) learn to output scores for the possible class labels. The shaded boxes represent a distribution over class labels.

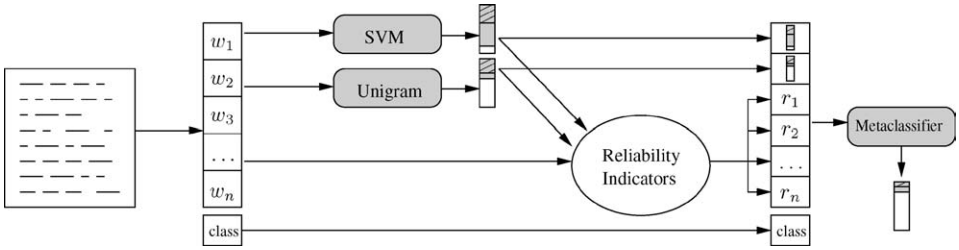

Figure 4. Architecture of *STRIVE*. In *STRIVE*, an additional layer of learning is added where the metaclassifier can use the context established by the reliability indicators and the output of the base classifiers to make an improved decision. The reliability indicators are functions of the document and/or the output of the base classifiers.

base classifiers as well as the context established by the reliability indicators to make a final classification.

We require the outputs of the base classifiers to train the metaclassifier. Thus, we perform cross-validation over the training data and use the resulting base classifier predictions, obtained when an example serves as a validation item, as training inputs for the metaclassifier. We note that in the case where the set of reliability indicators are restricted to be the identity function over the original data, then the resulting scheme can be viewed as a variant of cascade generalization (Gama 1998a).

4. Experimental analysis

We performed a large number of experiments to test the value of probabilistic classifier combination with reliability-indicator variables. We now 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 by Dumais and Chen (2000).

The MSN Web hierarchy is a seven-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 have assigned 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 195 K words appear in at least three training documents.

4.1.2. Reuters. The Reuters 21578 corpus (Lewis 1997) 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 taggers 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 ten most frequent classes since small numbers of *testing* examples makes estimating some performance measures unreliable due to high variance. Limiting to the ten largest classes allows us to compare our results to previously published results (Zhang and Oles 2001, Dumais et al. 1998, Joachims 1998, McCallum and Nigam 1998, Platt 1999b). 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 15 K words appear in at least three 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 by Lewis et al. (1996). As described by Lewis and Gale (1994) (see also Lewis (1995)), the categories are defined by keywords in a keyword field. The title and body fields are used in the experiments below. There are twenty categories in total.

The frequencies of the twenty classes are the same as those reported by Lewis et al. (1996). 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 123 K 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. In an attempt to isolate the benefits gained from the probabilistic combination of classifiers with reliability indicators, we worked to keep the representations for the base classifiers in our experiments nearly identical. 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 (WinMine Toolkit v1.0, 2001). *Dnet* builds decision trees using a Bayesian machine learning algorithm (Chickering et al. 1997, Heckerman et al. 2000). While this toolkit is targeted primarily at building models that provide probability estimates, we found that *Dnet* models usually perform acceptably for the goal of minimizing error rate. However, we found that the performance of *Dnet* with regard to other measures is sometimes poor.

For linear SVMs, we used the *Smox* toolkit which is based on Platt's **S**equential **M**inimal **O**ptimization algorithm (Platt 1999a). We have experimented with both binary and continuous feature representations, and both perform at approximately the same level of accuracy. In order to keep the base representations used by each classifier as similar as possible, we used a continuous model.

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 the *naïve Bayes* classifier.

4.2.2. Basic combination methods. We performed experiments to explore a variety of classifier-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 centers on taking a majority vote of the base classifiers. This approach is perhaps the most popular methodology used for the combination of text classifiers. 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 as this procedure 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 transforms *STRIVE* to a stacking methodology (Ting and Witten 1999, Wolpert 1992). 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. We note that *Stack-S* is also a weighted linear combination method since it is based on a linear SVM and uses only the classifier outputs.

We found it was challenging to learn the weights for an SVM when the inputs have vastly different scales. At times, it is not possible to identify good weights. To address the problem of handling inputs with greatly varying scales, we use an input normalization procedure: We normalize the inputs to the metaclassifiers to have 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 in the absence of the normalization procedure; as might be expected the impact of normalization for decision-tree learners is relatively minimal (and has both positive and negative influences). To denote the metaclassifiers whose inputs have been normalized in this manner, we append "(*norm*)" to their names.

STRIVE. Similar to the notation described above, we add a letter to *STRIVE* to denote the particular metaclassifier method 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, as above, is again noted by appending "(*norm*)" to the system names.

The experiments reported here use a total of 49 reliability indicators (including those specific examples given in Section 3.1). The full list of reliability indicators is described in detail in Appendix A. These reliability indicators were formulated by hand as an initial pass at representing potentially valuable contexts. We are currently taking a closer look at fundamental informational properties of different reliability indicators and have examined procedures for identifying new reliability indicators. We delve more deeply into the nature and authoring of reliability indicators in forthcoming work.

4.2.4. BestSelect classifier. To study the effectiveness of the *STRIVE* methodology, we formulated a simple optimal combination approach as a point of reference. Such an upper bound can be useful as a benchmark in experiments with classifier combination procedures. This bound follows quite naturally, when classifier combination is formulated as the process of *selecting* the best base classifier, on a per-example basis.

To classify a given document, if any of the classifiers correctly predict that document's class, 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* is the theoretical upper-bound on performance when combining a set of classifiers 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 uninformatively loose. Even though we are not working in a pure *selection* framework, we found it is rarely the case the metaclassifier outputs a prediction which none of the base classifiers made. Therefore, we have employed this *BestSelect* bound to assist with understanding the performance of *STRIVE*.

4.3. Performance measures

To compare the performance of the classification methods we look at a set of standard performance measures. The F1 measure (van Rijsbergen 1979, Yang and Liu 1999) is the harmonic mean of precision and recall where $Precision = \frac{Correct\ Positives}{Predicted\ Positives}$ and $Recall = \frac{Correct\ Positives}{Actual\ Positives}$. For F1, we can either macro-average or micro-average. In macro-averaging, the score 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 averaged F1.

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 (Provost and Fawcett 2001). 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 (Provost and Fawcett 2001). We report results on the area under the ROC curve as an attempt to summarize the linear utility space of functions.

4.4. Experimental methodology

As the categories under consideration in the experiments are not mutually exclusive, the classification was carried out 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. For the micro performance measures, obtaining truly optimal performance requires optimizing all the thresholds in a corpus in conjunction; we have taken the more computationally efficient approach of using the macro-optimized thresholds (i.e., each class's threshold is set independently from the thresholds for the other classes).

Table 1. Performance on MSN Web Directory corpus.

Method	Macro F1	Micro F1	Error	C(1,10)	C(10,1)	ROC area
Dnet	0.5502	0.5837	0.0583	0.3023	0.0771	0.8812
Smox	0.6705	0.7001	0.0455	0.2239	0.0799	0.9125
Naïve Bayes	0.5527	0.5619	0.0649	0.2853	0.0798	0.8915
Unigram	0.5982	0.6116	0.0594	0.2589	0.0812	0.9003
Best By Class	0.6705	0.7001	0.0455	0.2236	0.0783	N/A
Majority BBC	0.6668	0.6919	0.0476	0.2173†	0.0761	N/A
Stack-D (norm)	0.6775	0.7038†‡	0.0446†‡	0.2118†	0.0784	0.9292†
Stack-S (norm)	0.6732	0.7037†‡	0.0450	0.2174†	0.0757	0.9210†
STRIVE-D	0.6877†‡	0.7179†‡	**0.0429**†‡	**0.1939**†‡	0.0742	0.9383†
STRIVE-D (norm)	0.6908†‡	0.7178†‡	0.0434	0.1949†‡	0.0742	**0.9398**†
STRIVE-S (norm)	**0.6948**†‡	**0.7233**†‡	0.0430†‡	0.2037†	**0.0712**	0.9114
STRIVE-D (norm, omit Smox)	0.6670	0.6945	0.0464	0.2062†‡	0.0754	0.9361†
BestSelect	0.8365	0.8577	0.0270	0.0905	0.0616	N/A

The best performance (omitting the oracle *BestSelect*) in each column is given in bold. The † and ‡ summarize the macro sign test and, for micro F1, the micro 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 that the method outperforms the basic combination methods.

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 main 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 were performed (Yang and Liu 1999). For micro-F1, a one-sided micro sign test was performed (Yang and Liu 1999). 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 over the main performance results; instead, our analysis follows the macro and micro sign test which yield more conservative comparisons (i.e., the t-test primarily increased the number of differences found to be significant in the tables).

The classifier combinations are annotated to indicate the results of the macro sign test and, for micro F1, the micro sign test. A † indicates the method significantly outperforms

Table 2. Performance on Reuters corpus.

Method	Macro F1	Micro F1	Error	C(1,10)	C(10,1)	ROC
Dnet	0.7846	0.8541	0.0242	0.0799	0.0537	0.9804
Smox	0.8480	0.9102	0.0157	0.0580	0.0390	0.9815
Naïve Bayes	0.6574	0.7908	0.0320	0.1423	0.0527	0.9703
Unigram	0.7645	0.8674	0.0234	0.0713	0.0476	0.9877
Best By Class	0.8592	0.9126	0.0153	0.0518	0.0409	N/A
Majority BBC	0.8524	0.9097	0.0160	0.0448	0.0446	N/A
Stack-D (norm)	0.8636	0.9181	0.0153	0.0449	0.0392	0.9893
Stack-S (norm)	0.8720†‡	0.9201†‡	0.0143†	0.0445	0.0365	0.9930†
STRIVE-D	0.8547	0.9106	0.0154	0.0472	0.0358	0.9903
STRIVE-D (norm)	0.8526	0.9085	0.0157	0.0468	0.0359	0.9897
STRIVE-S (norm)	**0.8749**†	**0.9235**†‡	**0.0125**†‡	**0.0382**†	**0.0353**	**0.9939**
STRIVE-D (norm, omit Smox)	0.8433	0.8961	0.0168	0.0484	0.0425	0.9900
BestSelect	0.9529	0.9725	0.0050	0.0100	0.0202	N/A

The best performance (omitting the oracle *BestSelect*) in each column is given in bold. The † and ‡ summarize the macro sign test and, for micro F1, the micro 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 that the method outperforms the basic combination methods.

Table 3. Performance on TREC-AP corpus.

Method	Macro F1	Micro F1	Error	C(1,10)	C(10,1)	ROC
Dnet	0.6015	0.5798	0.0065	0.0342	0.0079	0.9768
Smox	0.7335	0.6966	0.0049	0.0294	0.0077	0.9691
Naïve Bayes	0.5676	0.5349	0.0065	0.0455	0.0078	0.9755
Unigram	0.6001	0.5695	0.0064	0.0347	0.0079	0.9819
Best By Class	0.7335	0.6966	0.0049	0.0294	0.0077	N/A
Majority BBC	0.7145	0.6751	0.0056	0.0292	0.0075	N/A
Stack-D (norm)	0.7357	0.6982	0.0049	**0.0241**†‡	0.0086	0.9856
Stack-S (norm)	0.7351	0.6997	0.0049	0.0288	**0.0075**	0.9656
STRIVE-D	0.7325	0.6991†‡	**0.0048**	0.0276	0.0078	0.9858
STRIVE-D (norm)	0.7280	0.6973	0.0049	0.0271	0.0077	0.9855
STRIVE-S (norm)	**0.7431**	**0.7078**†‡	0.0048	0.0295	0.0076	0.9634
STRIVE-D (norm, omit Smox)	0.6938	0.6527	0.0055	0.0313	0.0077	**0.9858**
BestSelect	0.8763	0.8195	0.0034	0.0149	0.0062	N/A

The best performance (omitting the oracle *BestSelect*) in each column is given in bold. The † and ‡ summarize the macro sign test and, for micro F1, the micro 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 that the method outperforms the basic combination methods.

(at the 0.05 level) the best base classifier. In addition, on the variants of *Stack* and *STRIVE*, a ‡ indicates that the method outperforms the basic combination methods. Results for the remaining sign test comparisons are omitted for brevity.

Tables 4, 5 and 6, present the performance results for additional experiments (described in Section 4.6.3) we conducted to determine how much could be gained by incorporating the

Table 4. Results of incorporating reliability indicators directly into a base classifier for the MSN Web Directory corpus.

Method	Macro F1	Micro F1	Error	C(1,10)	C(10,1)	ROC area
Dnet	0.5502	0.5837	0.0583	0.3023	0.0771	0.8812
Dnet + RIV–	0.6547	0.6841	0.0489	0.2136	0.0760	0.9295
Dnet + RIV	0.6825	0.7127	0.0447	0.2022	0.0781	0.9374
Stack-D (norm)	0.6775	0.7038	0.0446	0.2118	0.0784	0.9292
STRIVE-D	0.6877	**0.7179**	**0.0429**	**0.1939**	**0.0742**	0.9383
STRIVE-D (norm)	**0.6908**	0.7178	0.0434	0.1949	0.0742	**0.9398**

The best performance in each column is given in bold.

Table 5. Results of incorporating reliability indicators directly into a base classifier for the Reuters corpus.

Method	Macro F1	Micro F1	Error	C(1,10)	C(10,1)	ROC area
Dnet	0.7846	0.8541	0.0242	0.0799	0.0537	0.9804
Dnet + RIV–	0.8429	0.8985	0.0167	0.0629	0.0444	0.9888
Dnet + RIV	0.8479	0.9074	0.0161	0.0460	0.0439	**0.9908**
Stack-D (norm)	**0.8636**	**0.9181**	**0.0153**	**0.0449**	0.0392	0.9893
STRIVE-D	0.8547	0.9106	0.0154	0.0472	**0.0358**	0.9903
STRIVE-D (norm)	0.8526	0.9085	0.0157	0.0468	0.0359	0.9897

The best performance in each column is given in bold.

Table 6. Results of incorporating reliability indicators directly into a base classifier for the TREC-AP corpus.

Method	Macro F1	Micro F1	Error	C(1,10)	C(10,1)	ROC area
Dnet	0.6015	0.5798	0.0065	0.0342	0.0079	0.9768
Dnet + RIV–	0.6716	0.6444	0.0056	0.0291	0.0078	0.9853
Dnet + RIV	0.7037	0.6734	0.0052	0.0315	0.0078	0.9854
Stack-D (norm)	**0.7357**	0.6982	0.0049	**0.0241**	0.0086	0.9856
STRIVE-D	0.7325	**0.6991**	**0.0048**	0.0276	0.0078	**0.9858**
STRIVE-D (norm)	0.7280	0.6973	0.0049	0.0271	**0.0077**	0.9855

The best performance in each column is given in bold.

reliability indicators directly into a base classifier versus combining them hierarchically with classifier outputs, as in STRIVE. Thus, this ablation experiment aims to empirically examine how much first-order information the reliability indicators have versus their information when conditioned on the other classifier outputs. The best performance in each column is given in bold.

The significance tests for this ablation experiment are summarized as an ordering over the systems in Table 7. For each performance measure, the systems are ordered from best (left) to worst (right). When *all* systems above a performance level significantly outperform *all* the lower performers, it is denoted in the following manner: when significant according to the macro sign test, a "|" is used to separate the columns; when significant according to the macro *t*-test a, "‖" is used as the column-separator; when significant according to both tests, a "‖|" is used. For micro F1, which uses only a *micro* sign test, a "|" is used.

4.6. Discussion

First, we note that the base classifiers are competitive and consistent with the previously reported results over these corpora (Zhang and Oles 2001, Dumais and Chen 2000, Dumais et al. 1998, Joachims 1998, Lewis 1995, Lewis and Gale 1994, McCallum and Nigam 1998).[1] Furthermore, the fact that the linear SVM *Smox* tends to be the best base classifier is consistent with the literature (Dumais et al. 1998, Joachims 1998, Yang and Liu 1999).

4.6.1. MSN Web Directory. Examining the main results for the MSN Web Directory corpus in Table 1 highlights 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 the performance of a very good learner (*Smox*) tends to be diminished 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 positively for a class. 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 compared to the best theoretical performance that could be achieved by a per-example selection model using these base classifiers (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, by the ROC area measure of performance, *STRIVE-D* beats the base classifiers and basic combiners on 13/13 classes, and it beats the stacking methods on 12/13 classes.

Table 7. An ordered comparison of classification models built with reliability indicators directly added to a base classifier versus hierarchical combination models.

	First	Second	Third	Fourth	Fifth	Sixth
Macro F1						
MSN Web	STRIVE-D (norm)	STRIVE-D	Dnet+RIV	Stack-D (norm) ‖	Dnet+RIV– ‖	Dnet
Reuters	Stack-D (norm)	STRIVE-D	STRIVE-D (norm)	Dnet+RIV	Dnet+RIV–	Dnet
TREC-AP	Stack-D (norm)	STRIVE-D	STRIVE-D (norm) ‖	Dnet+RIV ‖	Dnet+RIV– ‖	Dnet
Micro F1						
MSN Web	STRIVE-D	STRIVE-D (norm)	Dnet+RIV	Stack-D (norm)	Dnet+RIV–	Dnet
Reuters	Stack-D (norm)	STRIVE-D	STRIVE-D (norm)	Dnet+RIV	Dnet+RIV–	Dnet
TREC-AP	STRIVE-D	Stack-D (norm)	STRIVE-D (norm)	Dnet+RIV	Dnet+RIV–	Dnet
Error						
MSN Web	STRIVE-D	STRIVE-D (norm)	Stack-D (norm)	Dnet+RIV ‖	Dnet+RIV– ‖	Dnet
Reuters	Stack-D (norm)	STRIVE-D	STRIVE-D (norm)	Dnet+RIV	Dnet+RIV–	Dnet
TREC-AP	STRIVE-D ‖	STRIVE-D (norm) ‖	Stack-D (norm) ‖	Dnet+RIV ‖	Dnet+RIV– ‖	Dnet
C(1,10)						
MSN Web	STRIVE-D	STRIVE-D (norm) ‖	Dnet+RIV ‖	Stack-D (norm)	Dnet+RIV– ‖	Dnet
Reuters	Stack-D (norm)	Dnet+RIV	STRIVE-D (norm)	STRIVE-D	Dnet+RIV–	Dnet
TREC-AP	Stack-D (norm)	STRIVE-D (norm)	STRIVE-D	Dnet+RIV–	Dnet+RIV	Dnet
C(10,1)						
MSN Web	STRIVE-D	STRIVE-D (norm)	Dnet+RIV–	Dnet	Dnet+RIV	Stack-D (norm)
Reuters	STRIVE-D	STRIVE-D (norm)	Stack-D (norm)	Dnet+RIV	Dnet+RIV–	Dnet
TREC-AP	STRIVE-D (norm)	Dnet+RIV–	STRIVE-D	Dnet+RIV	Dnet	Stack-D (norm)
ROC Area						
MSN Web	STRIVE-D (norm) ‖	STRIVE-D	Dnet+RIV ‖	Dnet+RIV–	Stack-D (norm) ‖	Dnet
Reuters	Dnet+RIV	STRIVE-D	STRIVE-D (norm)	Stack-D (norm)	Dnet+RIV– ‖	Dnet
TREC-AP	STRIVE-D	Stack-D (norm)	STRIVE-D (norm)	Dnet+RIV	Dnet+RIV–	Dnet

For each performance measure, the systems are ordered from best (left) to worst (right). When *all* systems above a performance level significantly outperform *all* the lower performers, it is denoted as follows: when significant according to a sign test, a "‖" is used to separate the columns; when significant according to the macro *t*-test, a "‖" is used; when significant according to both tests, a "‖" is used.

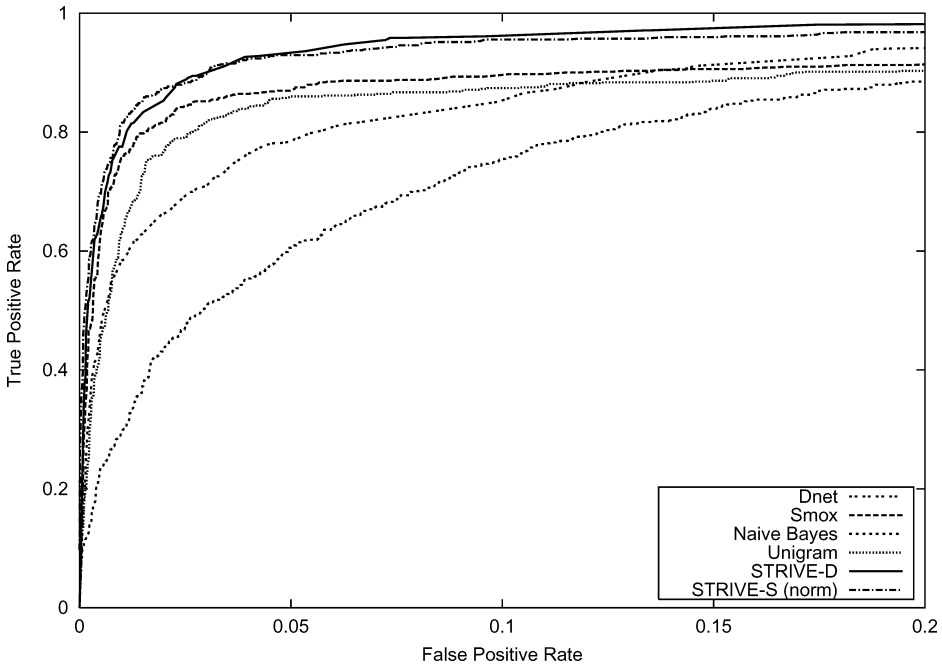

Figure 5. The ROC curve for the *Health & Fitness* class in the MSN Web Directory corpus.

The notable exception is the performance of *STRIVE-S* (*norm*) on ROC area; graphical inspection of the ROC curves suggests this result arises because too much weight is 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). The *Smox* 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 5, we can see that the two *STRIVE* variants dominate the 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. The base classifiers, in fact, do surpass *STRIVE-S* (*norm*). As a result, *STRIVE-D* is usually a more appropriate choice if the utility function penalizes false negatives significantly more heavily than false positives.

In some cases, we can develop an understanding of why the decision tree is more appropriate for tracking crossovers. In the case portrayed in figure 2, it appears that the tree establishes a score region for *Smox* and a score region for *Dnet* where the reliability

indicators give further information about how to classify an example. Since a linear SVM is a weighted sum over the inputs, it cannot represent crossovers that are dependent on breaking a single variable into multiple regions (such as this one); it has to use the information present in other variables to try to distinguish these regions. Higher-order polynomial kernels are one way to allow an SVM to represent this type of information.

Finally, we performed an ablation experiment to determine how *STRIVE* would behave without the presence of an extremely strong base classifier. Since *Smox* frequently outperforms the other base classifiers, we investigated the level of performance *STRIVE* could obtain if the output of *Smox* was omitted from the inputs given to the metaclassifier, *STRIVE-D* (*norm*, *omit Smox*). The results show that when we omit the base classifier *Smox*, the resulting combination improves by a large margin over the remaining base methods; however, the resulting classifier generally still fails to beat *Smox*'s individual performance. 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.

4.6.2. 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. Since it is common in the literature to show results for each class of the top ten largest classes in Reuters, we provide a detailed listing in Appendix B.

4.6.3. Additional experiments. Finally, we investigated whether the reliability indicators could be directly incorporated into the base classifiers. That is, we wanted to understand to what extent their information can be directly used to improve classification and to what extent it is conditional on the presence of classifier outputs. To examine these issues, we performed an experiment where we added all of the reliability indicators to the standard document representation and built a model using *Dnet*. The resulting system is denoted *Dnet + RIV*. Since some of the reliability indicators contain information equivalent to the output of other classifiers, we also built a model that uses almost all of the reliability indicators[2] and the standard document representation (denoted *Dnet + RIV–*). We compare these systems to their most similar counterparts presented in the earlier results: the base classifier (*Dnet*); the stacking method using the decision tree as a metaclassifier (*Stack-D* (*norm*)); and the normalized and standard version of *STRIVE-D*.

The expectation is that *Dnet* will be outperformed by all other systems, and *Dnet + RIV–* will be outperformed by all others except *Dnet*. A quick examination of the results indicates that this is generally the case according to nearly all of the performance measures. The amount of improvement of *Dnet + RIV–* over *Dnet* indicates the extent to which the reliability indicators give information that can be used directly to improve classification—which is quite large when compared to *Dnet*. However, the remaining methods still often show significant improvement beyond this when all the base classifier outputs are used. This experiment also highlights that because of the rarity of the classes, it is extremely challenging to produce any significant wins for the C(10,1)

function which places a heavy penalty on false positives, i.e., emphasizing extremely high precision.

We also tried integrating the reliability-indicator variables directly into the *Smox* classifier as well. We were surprised to observe a tendency for the test performance to decrease significantly below the level of the base *Smox* system. We believe this is the result of representational inhomogeneity: the majority of words that define support vectors are not present in any given document, but all the reliability indicators have values for documents. Thus, the reliability indicators dominate the influence of words by their sheer magnitude and not necessarily by the reliability information they carry. This highlights the value of hierarchical modeling for classifier combination; hierarchical modeling allows variables of the same type to be modeled in the same layers. This highlights that one major advantage to hierarchically combining systems is that variables of the same type can be modeled in the same layer.

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. Also, we are interested in exploring the value of introducing more flexibility into the set of base classifiers that are being used. In the experiments we described, the classifiers were purposely held constant in an attempt to investigate the influence of reliability-indicator variables. In future studies, we will allow representations to vary to induce more variety among the base classifiers we have focused on. 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 (Nigam et al. 1999)) as classifiers performing better than random will be necessarily correlated.

In addition, we believe the *STRIVE* framework can be used to elucidate deeper, information-theoretic foundations of how classifiers leverage information in text classification. Currently, we build one metaclassifier for each binary topic problem in a corpus (e.g., *Acquisitions* vs. *not Acquisitions*). However, a promising abstraction of the *STRIVE* framework enables us to use model structure or parameters learned using data from one binary task for a separate binary task (even in a different corpus). This can be experimentally tested by coalescing metalevel training data from different binary discrimination tasks and building one, more general, metaclassifier. With this approach, we treat the metaclassifier as an abstraction moving the focus of the analysis from discriminating a specific topic (e.g., *Acquisitions* vs. *not Acquisitions*) to the problem of discriminating topic membership (i.e., *In-Topic* vs. *Out-of-Topic*). The base-level classifiers trained on a particular topic are used as the representation of topic-specific knowledge, while the metaclassifier provides information about how to leverage context across topic-classification in general. If the metaclassifier still improves performance of base classifiers, then the reliability indicators will have inductively defined informational properties for classifier combination across all of the text problems considered. Such an extension is only possible if we generalize the

reliability indicators away from linkages to the precise words in a document. Consider when "shares" occurs in a document in the *Acquisitions* discrimination task and "corn" occurs in a document in the *Corn Futures* discrimination task. One task-invariant representation of context at the metalevel might transform both of these to: *Is the word with maximum mutual information for the current task present in this document*? This representation enables the metaclassifier to use information about how document-specific context influences topic discrimination across a wide variety of text classification tasks. Recent experiments have demonstrated that this methodology can be valuable in enhancing the performance of classifiers (Bennett et al. 2003). We are continuing to pursue this promising extension.

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, informing a metaclassifier about the best way to integrate the outputs from base-level classifiers. 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.

We conducted experimental evaluations over three text-classification corpora (MSN Web, Reuters 21578, and TREC-AP) with a variety of performance measures. These measures were selected to determine the robustness of the classification procedures under different misclassification penalties. The empirical evaluations support the conclusion that a simple majority vote in situations where one of the classifiers performs strongly can weaken the best classifier's performance. In contrast, in all of these corpora across all measures, the *STRIVE* methodology was competitive, failing to produce the top performer in only two instances (the two skewed linear utility measures in the TREC-AP corpus). Furthermore, on a class-by-class basis, the *STRIVE* methodology produced receiver-operating characteristic curves that dominated the other classifiers in nearly every class of the MSN Web corpus—demonstrating that it provides the best choice for *any* possible linear utility function in this corpus. In conclusion, the experiments show that stacking and *STRIVE* provide robust combination schemes across a variety of performance measures.

Appendix A: Detailed descriptions of inputs to STRIVE

This appendix gives details for all of the inputs to STRIVE—including the base classifier outputs in addition to all 49 reliability-indicator variables.

A.1. Outputs of base classifiers

We considered the outputs of four base classifiers as inputs to *STRIVE*.

- *OutputOfDnet*
  This is the output of the decision tree built using the *Dnet* classifier (available as the WinMine toolkit (WinMine Toolkit v1.0 2001)). Its value is the estimated probability at the leaf node of belonging to the class.
- *OutputOfSmox*
  This is the output of the Linear SVM model built using the *Smox* toolkit. It is a probability estimate of class membership obtained via a monotonic transformation using a fitted sigmoid (Platt 1999) of the raw score of the SVM.
- *OutputOfNaïvebayes*
  This is the output of the *naïve Bayes* model built using a multivariate Bernoulli representation (i.e. only feature presence/absence in an example is modeled) (McCallum and Nigam 1998). It is the log-odds (or logistic) of the model's probability estimate of class membership, i.e. $\log \frac{P(c|d)}{1-P(c|d)}$. The direct probability estimate is not used because typically enough machine floating-point precision is not available to preserve the ranking induced by this model (they cluster too closely to zero and one).
- *OutputOfUnigram*
  This is the output of the *unigram* model (also referred to as a multinomial model (McCallum and Nigam 1998). It is the log-odds (or logistic) of the model's probability estimate of class membership, i.e. $\log \frac{P(c|d)}{1-P(c|d)}$. The direct probability estimate is not used because typically enough machine floating-point precision is not available to preserve the ranking induced by this model (they cluster too closely to zero and one).

A.2. Reliability indicator variables

Indicator variables are currently broken roughly into one of four types:

- Amount of information present in the original document (15/15);
- Information loss or mismatch between representations (12/12);
- Sensitivity of the decision to evidence shift (20/6);
- Basic voting statistics (2/2).

The first number in parentheses is the number of variables of this type that were present in the experimental evaluation. The second number is the number of those whose effect on the metaclassifier model appears to be non-negligible. We group the reliability indicators into their primary type (based on the main reasons we expect to see a link to classifier reliability). We note that this is only a soft clustering; some reliability indicators may provide context information in more than one way.

Several of the variables listed below have an instantiation for each class in a learning problem (the variable counts we report tallies each instance separately). For these variables

below we list only one entry and use "{Class}" in the name of the variable to denote that this variable has one instantiation per class. Since our methodology built a binary classifier for each topic, then our experiments have a *Positive* and a *Negative* class version. In a two-class problem, the values of the two instantiations may be redundant. We have, however, retained each since in polyclass (3 or more classes) discrimination they are more distinct.

After each bullet below, a number is given in parentheses, indicating the number of variables that this description includes. Thus, the total of the numbers in parentheses is 49.

A.2.1. Type 1: Amount of information present in the original document. There are 15 reliability indicator variables whose primary type is considered to be this type (11 not counting instantiations for each class).

- (1) *DocumentLength*
  The number of words in a document before feature selection. Presumably longer documents provide more information to base a decision upon. Therefore, longer documents will lead to more reliable decisions (when *DocumentLength* is correctly modeled). Alternatively, models that do not correctly normalize for document length may be less reliable for extreme lengths (short or long) of documents.
- (1) *EffectiveDocumentLength*
  DocumentLength minus the number of out-of-vocabulary words in the document. Since a model cannot generalize strongly (other than smoothing) for features that were not seen in the training set, this variable may be a better indicator of information present in the document than *DocumentLength*.
- (1) *NumUniqueWords*
  Number of distinct tokens in a document, i.e. $|\{w \mid w \in document\}|$ (as opposed to length which counts repeats of a token in a document). The motivation is similar to *DocumentLength*, but here the variable is only counting each new word as an indicator of new information.
- (1) *EffectiveUniqueWords*
  NumUniqueWords minus the number of unique out-of-vocabulary words. This is the analogue of *EffectiveDocumentLength* and is included for similar reasons.
- (1) *PercentUnique*
  This is a measure of the variety in word choice in a document. It is equal to *NumUniqueWords*/*DocumentLength*. This can also be seen as 1/average number of times a word is repeated in a document. Close to 1 means very few words (if any) are repeated in the document; close to 0 means the documents consists of very few unique words (possibly repeated many times). This is essentially a normalized version of *NumUniqueWords*; however this variable will show high variance for short documents. The intuition here is that more complex documents, while providing more information, also might be more difficult to classify (since they may have many features each carrying some small weight).

- (1) *PercentOOV*
  The percentage of the words in a document which weren't seen in the training set. It is equal to the number of out-of-vocabulary words divided by *DocumentLength*. Similar to *PercentUnique*, this variable can show high variance for short values. The intuition here is that the more novel words a document contains the more likely a classifier is to incorrectly classify the document into the a priori prevalent class (typically unseen words slightly favor minority classes since we have less samples from them). This is a variable that essentially allows a global smoothing model to be induced. Its range is [0, 1]. Therefore, as it approaches 1, we would expect minority classes to be more likely than our base models might estimate.
- (1) *PercentUniqueOOV*
  The percentage of the words (not counting duplicates) in a document which weren't seen in the training set. This is the distinct token analogue for *PercentOOV*. Again, the motivations are similar to just using a different information model.
- (2) *PercentIn*{*Class*}*BeforeFS*
  Of all words occurring in the training set (i.e. out-of-vocabulary words are ignored), the percentage of words in a document that occurred at least once in examples belonging to the class. It is equal to the number of words that occurred in the class before feature selection divided by *EffectiveDocumentLength*. Similar to *PercentOOV*, this can be used to inductively learn smoothing behavior. The assumption is that if this variable is high, predictions that the example belongs to the class are more reliable. For the binary case with a negative class that effectively groups many classes together, this isn't quite expected with respect to *PercentInNegBeforeFS* (since predictions of "negative" would almost always expected to be more reliable under that assumption).
- (2) *UpercentIn*{*Class*}*BeforeFS*
  Of all words occurring in the training set (i.e. out-of-vocabulary words are ignored), the percentage of unique words in a document that occurred at least once in examples belonging to the class. This is the analogue to *PercentIn*{*Class*}*BeforeFS* using unique tokens as the basis for the information model.
- (2) *%Favoring*{*Class*}*BeforeFS*
  Of all words occurring in the training set, the percentage of words in a document that occurred more times in examples belonging to the class than in examples not belonging to the class. This is essentially a rough statistic for an *unnormalized* unigram model (tied slightly into the smoothing related variables discussed above) that gives a very rough sense of the evidential weight of the original document.
- (2) *U%Favoring*{*Class*}*BeforeFS*
  Of all words occurring in the training set, the percentage of unique words in a document that occurred more times in examples belonging to the class than in examples not belonging to the class. This is the analogue to *%Favoring*{*Class*}*BeforeFS*.

A2.2. Type 2: Information loss or mismatch between representations. There are 12 reliability indicator variables whose primary type is considered to be this type. While each of these variables are a measure of loss of information, they all generally have

a paired variable of Type 1 that together give a more direct measure of information loss.

- (1) *DocumentLengthAfterFS*
  The number of words in a document after out-of-vocabulary words have been removed and feature selection was performed. Similar to *DocumentLength*, this is the measure of information that the classifier actually sees with respect to this document. Currently, it's assumed the metaclassifier can combine this with *DocumentLength* to infer information loss.
- (1) *UniqueAfterFS*
  The number of unique words remaining in a document after out-of-vocabulary words have been removed and feature selection was performed. This is the distinct token analogue of *DocumentLengthAfterFS* and is similarly expected to be used in conjunction with *NumUniqueWords* as a gauge of information loss.
- (1) *PercentRemoved*
  The percentage of a document that was discarded because it was out-of-vocabulary or removed by feature selection. It can have high variance for short documents. The intuition is that reliability of a classifier is higher for low values of *PercentRemoved*.
- (1) *UniquePercentRemoved*
  The percentage of unique words in a document that were discarded because they were out-of-vocabulary or removed by feature selection. The distinct token analogue of *PercentRemoved* where the information model is unique words.
- (2) *PercentIn{Class}AfterFS*
  Of all words occurring in the training set, the percentage of words remaining in a document after feature selection that occurred at least once in examples in the class. Together with *PercentIn{Class}BeforeFS*, allows the model to inductively model shift in information content because of feature selection.
- (2) *UpercentIn{Class}AfterFS*
  Of all words occurring in the training set, the percentage of unique words remaining in a document (after feature selection) that occurred at least once in the class. Again, this is expected to be used in conjunction with *UpercentIn{Class}BeforeFS* to model information loss.
- (2) *%Favoring{Class}AfterFS*
  Of all words occurring in the training set, the percentage of words remaining in a document (after feature selection) that occurred more times in examples in the class than in examples not in the class. Like its BeforeFS counterpart, it is essentially like an unnormalized unigram model. We expect that it can be used in conjunction with *%Favoring{Class}BeforeFS* to measure how a feature selection method may have biased the information for a given document toward a particular class.
- (2) *U%Favoring{Class}AfterFS*
  Of all words occurring in the training set, the percentage of distinct words remaining in a document after feature selection that occurred more times in examples in the class than examples not in the class.

A2.3. Type 3: Sensitivity of the decision to evidence shift. There are 20 reliability indicator variables that fall into this class. Ten of them are based on a unigram model (where only the probabilities of the words present are modeled), and the other ten are based on a naïve Bayes multivariate Bernoulli model (where a document is modeled in terms of the probabilities of each word's presence/absence).[3]

- (2) *UnigramVariance*, *NaïveBayesVariance*
  In a binary class problem, the weight each word contributes to the unigram model's decision is $\log \frac{P(w|c)}{P(w|\neg c)}$. Similarly, each word's presence/absence contributes a weight of $\log \frac{P(w=\{\text{present,absent}\}|c)}{P(w=\{\text{present,absent}\}|\neg c)}$ to the naïve Bayes model. If this ratio is greater than 0, the word gives evidence to the positive class (c), and if it is less than zero, the word gives evidence to the negative class ($\neg c$). The reliability indicators are the variance of these weights for the feature values (word occurrences or presence/absence) in a specific document. If the variance is close to zero, that means all of the words tended to point toward one class. As the variance increases, this means there was a large skew in the amount of evidence present by the various words (possibly strong words pulling toward two classes). The intuition is that the reliability of naïve Bayes related classifiers will tend to decrease as this variable increases. To apply this to polyclass learning problems, there should be one value per class and the weight should be either $\log \frac{P(w|c)}{1-P(w|c)}$ or $\log \frac{P(w|c)}{\max_{c' \neq c} P(w|c')}$ with the second being preferred.
- (4) *UnigramVarianceOfLogOfWordGiven*$\{Class\}$, *NaïveBayesVarianceOfLogOfWord - Given*$\{Class\}$
  These variables represent the variance of the log of the conditional probabilities of a word given a class. For example *UnigramVarianceOfLogOfWordGivenPositive* is the variance of $\log P(w \mid Class = Positive)$ over the words in the document.
- (4) *UnigramMeanOfWordGiven*$\{Class\}$, *NaïveBayesMeanOfWordGiven*$\{Class\}$
  These variables represent the mean of the conditional probabilities of a word given a class. For example *UnigramMean OfWordGivenPositive* is the mean of $P(w \mid Class = Positive)$ over the words in the document. None of these variables had much impact on the metaclassifier model and have been eliminated from future study as a result.
- (4) *UnigramMeanOfLogOfWordGiven*$\{Class\}$, *NaïveBayesMeanOfLogOfWordGiven-*$\{Class\}$
  These variables represent the mean of the conditional probabilities of a word given a class. For example *UnigramMeanOfLogOfWordGivenPositive* is the mean of $\log P(w \mid Class = Positive)$ over the words in the document. This essentially provides a log scaling of the variables (*UnigramMeanOfWordGiven*$\{Class\}$, *NaïveBayesMeanOfWordGiven*$\{Class\}$) described above and is only a factor if the metaclassifier is sensitive to the scaling. None of these variables had much impact on the metaclassifier model and have been eliminated from future study as a result.
- (4) *UnigramVarianceOfWordGiven*$\{Class\}$, *NaïveBayesVarianceOfWordGiven*$\{Class\}$
  These variables represent the variance of the log of the conditional probabilities of a word given a class. For example *UnigramVarianceOfWordGivenPositive* is the variance of $P(w \mid Class = Positive)$ over the words in the document. This essentially provides a

log scaling of the variables (*UnigramVarianceOfLogOfWordGiven*{*Class*}, *NaïveBayes-VarianceOfLogOfWordGiven*{*Class*}) described in the Type 3 section and is only a factor if the metaclassifier is sensitive to the scaling. We have found that the metaclassifiers have not been sensitive to this scaling, and these variables have been eliminated from future study while their counterparts are being retained.

- (2) *UnigramMeanOfLogOfRatioOfWordGivenClass*, *NaïveBayesMeanOfLogOfRatioOf-WordGivenClass*
  In a binary class problem, the weight each word contributes to the unigram model's decision is $\log \frac{P(w|c)}{P(w|\neg c)}$. Similarly, each word's presence/absence contributes a weight of $\log \frac{P(w=\{\text{present,absent}\}\,|\,c)}{P(w=\{\text{present,absent}\}\,|\,\neg c)}$ to the naïve Bayes model. If this ratio is greater than 0, the word gives evidence to the positive class (c), and if it is less than zero, the word gives evidence to the negative class ($\neg c$). The reliability indicators are the mean of these weights for the feature values (word occurrences or presence/absence) in a specific document. These variables are being eliminated from future study because their information is highly correlated with the output scores of the unigram and naïve Bayes classifiers.

A.2.4. Type 4: Basic voting statistics. There are 2 reliability indicator variables whose primary type is considered to be this type. Both of these were introduced mainly to reduce the data required to learn m-of-n rules in the decision tree metaclassifier.

- (1) *PercentPredictingPositive*
  We refer to this in the main text as *NumVotingForClass*. This variable is the percentage of base classifiers (out of all base classifiers) that vote for membership in the class. In our experimental evaluation, we only used one instantiation of this variable. This was added to help the search space since learning this m-of-n type of feature can require significant data for a decision tree learning algorithm (unless it is specifically altered for this).
- (1) *PercentAgreeWBest*
  This variable is referred to as *PercentAgreement* in the main text. For polyclass problems, *PercentAgreement* can be used to indicate how many classes the classifiers fracture their votes among. Since there are only two classes here, we altered it to indicate the percent agreement with the best base classifier (the classifier that performed best over the training data).

Appendix B: Detailed results for reuters

Since it is common in the literature to provide detailed performance results for the top ten most frequent classes in Reuters 21578, we present a breakdown by class here. For the reader's benefit, we also give after the class name the number of training documents and testing documents with membership in the class (*Train/Test*). The best result per class per performance measure (excluding *BestSelect*) is in bold.

Table 8. Details for Reuters by class.

Class/Method	F1	Error	C(1,10)	C(10,1)	ROC area
ACQ (1650/719)					
Dnet	0.8475	0.0634	0.1773	0.1910	0.9758
Smox	0.9280	0.0306	0.1176	0.0837	0.9891
Naïve Bayes	0.8554	0.0597	0.1658	0.1716	0.9796
Unigram	0.9249	0.0433	0.0943	0.1494	0.9900
Best By Class	0.9280	0.0306	0.1176	0.0837	N/A
Majority BBC	0.9341	0.0306	0.0818	0.1376	N/A
Stack-D (norm)	0.9431	0.0312	0.0891	0.0967	0.9924
Stack-S (norm)	0.9416	0.0285	0.0888	0.0888	0.9934
STRIVE-D	0.9354	0.0273	0.0994	0.0806	0.9938
STRIVE-D (norm)	0.9362	0.0270	0.0994	**0.0776**	**0.9939**
STRIVE-S (norm)	**0.9492**	**0.0218**	**0.0791**	0.0852	0.9935
STRIVE-D (norm, omit Smox)	0.9274	0.0309	0.1115	0.1000	0.9924
BestSelect	0.9804	0.0082	0.0145	0.0424	N/A
CORN (181/56)					
Dnet	0.9180	0.0030	**0.0033**	0.0170	0.9987
Smox	0.8421	0.0045	0.0255	0.0152	0.9813
Naïve Bayes	0.5481	0.0149	0.0612	0.0164	0.9844
Unigram	0.6115	0.0115	0.0391	0.0164	0.9910
Best By Class	0.9180	0.0030	**0.0033**	0.0152	N/A
Majority BBC	0.9060	0.0024	0.0082	0.0167	N/A
Stack-D (norm)	0.9106	0.0033	**0.0033**	0.0091	0.9994
Stack-S (norm)	**0.9402**	0.0024	0.0036	0.0088	0.9995
STRIVE-D	0.9204	0.0027	**0.0033**	0.0091	0.9995
STRIVE-D (norm)	0.9204	0.0027	**0.0033**	0.0091	0.9995
STRIVE-S (norm)	0.9381	**0.0021**	0.0055	0.0124	0.9996
STRIVE-D (norm, omit Smox)	0.9286	0.0024	**0.0033**	**0.0045**	**0.9997**
BestSelect	0.9825	0.0006	0.0006	0.0027	N/A
CRUDE (389/189)					
Dnet	0.8429	0.0200	0.0570	0.0603	0.9896
Smox	0.8587	0.0161	0.0464	0.0409	0.9928
Naïve Bayes	0.6220	0.0358	0.1622	0.0576	0.9644
Unigram	0.8421	0.0191	0.0421	0.0509	0.9942
Best By Class	0.8587	0.0161	0.0421	0.0509	N/A
Majority BBC	0.8632	0.0149	0.0361	0.0497	N/A
Stack-D (norm)	0.8655	0.0167	0.0379	0.0543	0.9952
Stack-S (norm)	0.8706	0.0158	0.0270	0.0388	0.9966

(*Continued on next page*.)

Table 8. (*Continued*).

Class/Method	F1	Error	C (1,10)	C (10,1)	ROC area
STRIVE-D	0.8365	0.0206	0.0367	0.0449	0.9927
STRIVE-D (norm)	0.8365	0.0206	0.0367	0.0449	0.9927
STRIVE-S (norm)	**0.8737**	**0.0145**	**0.0252**	**0.0239**	**0.9976**
STRIVE-D (norm, omit Smox)	0.8333	0.0170	0.0309	0.0443	0.9944
BestSelect	0.9764	0.0027	0.0079	0.0306	N/A
EARN (2877/1087)					
Dnet	0.9529	0.0315	0.0867	0.0479	0.9959
Smox	0.9790	0.0139	0.0458	0.0500	0.9951
Naïve Bayes	0.9348	0.0373	0.3383	0.0415	0.9687
Unigram	0.9617	0.0249	0.1949	0.0394	0.9837
Best By Class	0.9790	0.0139	0.0458	0.0500	N/A
Majority BBC	0.9751	0.0158	0.0549	0.0373	N/A
Stack-D (norm)	0.9794	0.0136	0.0512	0.0336	0.9978
Stack-S (norm)	**0.9798**	**0.0133**	0.0476	0.0491	0.9978
STRIVE-D	0.9786	0.0142	0.0527	**0.0312**	0.9982
STRIVE-D (norm)	0.9750	0.0167	0.0530	**0.0312**	0.9981
STRIVE-S (norm)	0.9793	0.0136	**0.0455**	0.0367	**0.9989**
STRIVE-D (norm, omit Smox)	0.9665	0.0221	0.0643	0.0470	0.9969
BestSelect	0.9954	0.0030	0.0070	0.0130	N/A
GRAIN (433/149)					
Dnet	0.8797	0.0085	0.0388	0.0258	0.9842
Smox	0.9342	0.0061	0.0264	0.0197	0.9853
Naïve Bayes	0.7205	0.0261	0.0737	0.0436	0.9877
Unigram	0.8148	0.0164	0.0446	0.0330	0.9900
Best By Class	0.9342	0.0061	0.0264	0.0197	N/A
Majority BBC	0.9161	0.0061	0.0215	0.0239	N/A
Stack-D (norm)	**0.9384**	**0.0055**	**0.0158**	0.0197	0.9919
Stack-S (norm)	0.9338	0.0061	0.0161	0.0173	**0.9950**
STRIVE-D	0.9129	0.0076	0.0285	0.0197	0.9916
STRIVE-D (norm)	0.9129	0.0076	0.0224	0.0197	0.9917
STRIVE-S (norm)	0.9352	0.0058	0.0164	0.0152	0.9882
STRIVE-D (norm, omit Smox)	0.9054	0.0085	0.0318	**0.0139**	0.9914
BestSelect	0.9703	0.0012	0.0061	0.0055	N/A
INTEREST (347/131)					
Dnet	0.5249	0.0318	0.1337	0.0452	0.9645
Smox	0.7470	0.0224	0.0997	0.0330	0.9597
Naïve Bayes	0.5059	0.0358	0.1367	0.0412	0.9635

(*Continued on next page*.)

Table 8. (*Continued*).

Class/Method	F1	Error	C (1,10)	C (10,1)	ROC area
Unigram	0.6471	0.0315	0.0773	0.0388	0.9840
Best By Class	0.7470	0.0224	0.0997	0.0412	N/A
Majority BBC	0.6807	0.0261	0.0846	0.0382	N/A
Stack-D (norm)	0.7470	0.0255	0.0843	0.0382	0.9806
Stack-S (norm)	0.7530	0.0227	0.0967	0.0364	0.9827
STRIVE-D	0.6933	0.0236	0.0906	0.0409	0.9804
STRIVE-D (norm)	0.6933	0.0236	0.0906	0.0382	0.9804
STRIVE-S (norm)	**0.7737**	**0.0188**	**0.0655**	**0.0303**	**0.9888**
STRIVE-D (norm, omit Smox)	0.6553	0.0273	0.1115	0.0385	0.9759
BestSelect	0.8755	0.0155	0.0155	0.0282	N/A
MONEY-FX (538/179)					
Dnet	0.5779	0.0412	0.1531	0.0591	0.9478
Smox	0.7756	0.0267	0.0815	0.0661	0.9788
Naïve Bayes	0.5526	0.0473	0.2143	0.0682	0.9425
Unigram	0.7077	0.0358	0.0727	0.0597	0.9823
Best By Class	0.7756	0.0267	0.0727	0.0661	N/A
Majority BBC	0.7627	0.0318	0.0809	0.0600	N/A
Stack-D (norm)	0.7754	0.0255	0.0712	**0.0570**	0.9844
Stack-S (norm)	0.7807	0.0246	0.0764	0.0606	0.9868
STRIVE-D	0.7403	0.0276	0.0867	0.0573	0.9839
STRIVE-D (norm)	0.7403	0.0276	0.0867	0.0597	0.9839
STRIVE-S (norm)	**0.8100**	**0.0206**	0.0573	0.0776	**0.9910**
STRIVE-D (norm, omit Smox)	0.7283	0.0294	**0.0558**	0.0879	0.9796
BestSelect	0.9070	0.0106	0.0233	0.0343	N/A
SHIP (197/89)					
Dnet	0.7284	0.0133	0.0418	0.0270	0.9800
Smox	0.8506	0.0079	0.0276	0.0318	0.9950
Naïve Bayes	0.7485	0.0136	0.0436	0.0318	0.9853
Unigram	0.8391	0.0085	0.0279	0.0333	0.9954
Best By Class	0.8506	0.0079	0.0276	0.0318	N/A
Majority BBC	**0.8706**	**0.0067**	**0.0182**	0.0264	N/A
Stack-D (norm)	0.8506	0.0079	0.0327	0.0318	0.9898
Stack-S (norm)	0.8686	0.0070	0.0197	0.0233	**0.9968**
STRIVE-D	0.8508	0.0082	0.0212	**0.0215**	0.9862
STRIVE-D (norm)	0.8508	0.0082	0.0212	**0.0215**	0.9862
STRIVE-S (norm)	0.8556	0.0079	0.0203	0.0267	0.9921
STRIVE-D (norm, omit Smox)	0.8343	0.0088	0.0233	0.0291	0.9857
BestSelect	0.9605	0.0024	0.0058	0.0082	N/A

(*Continued on next page*.)

Table 8. (*Continued*).

Class/Method	F1	Error	C (1,10)	C (10,1)	ROC area
TRADE (369/117)					
Dnet	0.6764	0.0249	0.1000	0.0418	0.9688
Smox	0.7034	0.0212	0.0755	0.0321	0.9520
Naïve Bayes	0.4348	0.0303	0.1828	0.0339	0.9380
Unigram	0.6496	0.0291	0.0773	0.0333	0.9808
Best By Class	0.7034	0.0212	0.0755	0.0333	N/A
Majority BBC	0.7100	0.0191	0.0527	0.0346	N/A
Stack-D (norm)	0.7289	0.0185	0.0555	0.0333	0.9694
Stack-S (norm)	0.7364	0.0191	0.0570	**0.0294**	0.9826
STRIVE-D	**0.7692**	0.0164	0.0461	0.0385	0.9843
STRIVE-D (norm)	0.7631	0.0176	0.0482	0.0385	0.9715
STRIVE-S (norm)	0.7449	**0.0155**	0.0640	0.0376	**0.9900**
STRIVE-D (norm, omit Smox)	0.7568	0.0158	**0.0446**	0.0349	0.9852
BestSelect	0.9283	0.0039	0.0133	0.0261	N/A
WHEAT (212/71)					
Dnet	0.8974	0.0048	0.0076	0.0215	0.9982
Smox	0.8613	0.0073	0.0339	0.0173	0.9855
Naïve Bayes	0.6517	0.0188	0.0440	0.0215	0.9891
Unigram	0.6466	0.0142	0.0424	0.0215	0.9859
Best By Class	0.8974	0.0048	0.0076	0.0173	N/A
Majority BBC	0.9054	0.0064	0.0094	0.0215	N/A
Stack-D (norm)	0.8974	0.0048	0.0076	0.0179	0.9916
Stack-S (norm)	**0.9150**	**0.0036**	0.0121	0.0121	0.9992
STRIVE-D	0.9091	0.0058	0.0070	0.0139	0.9919
STRIVE-D (norm)	0.8974	0.0058	0.0067	0.0188	0.9987
STRIVE-S (norm)	0.8889	0.0048	**0.0030**	**0.0070**	**0.9995**
STRIVE-D (norm, omit Smox)	0.8974	0.0058	0.0067	0.0249	0.9985
BestSelect	0.9524	0.0015	0.0058	0.0112	N/A

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. We would also like to thank the anonymous reviewers for the useful suggestions they provided.

Notes

1. While the results reported for Reuters are not directly comparable to those reported by Yang and Liu (1999) as these investigators report results over all 90 classes and do not give a breakdown for the ten most frequent

categories, others (Zhang and Oles 2001, Dumais et al. 1998, Joachims 1998, McCallum and Nigam 1998, Platt 1999b) provide published baselines over the ten largest classes.

2. The omitted variables are the Type 4 variables and the unigram and naïve Bayes variants of the Type 3 variable *MeanOfLogOfRatioOfWordGivenClass* discussed in Appendix A. We left in other variables such as *Unigram-Variance* that are related to other classifier models but are not directly correlated with their outputs.
3. Our current analysis of the metaclassifier models has restricted our attention to a set of only 35 of the 49 total variables used in the experiments above. The remaining variables have been eliminated from future study because they are highly correlated with other reliability indicators or have had little impact on the metaclassifier. All 14 eliminated variables are Type 3 variables and are listed last in this section.

References

Al-Kofahi K, Tyrrell A, Vacher A, Travers T and Jackson P (2001) Combining multiple classifiers for text categorization. In: CIKM '01, Proceedings of the 10th ACM Conference on Information and Knowledge Management, pp. 97–104.

Bartell BT, Cottrell GW and Belew RK (1994) Automatic combination of multiple ranked retrieval systems. In: SIGIR '94, Proceedings of the 17th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 173–181.

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

Bennett PN, Dumais ST and Horvitz E (2002) Probabilistic combination of text classifiers using reliability indicators: Models and results. In: SIGIR '02, Proceedings of the 25th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 207–214.

Bennett PN, Dumais ST and Horvitz E (2003) Inductive transfer for text classification using generalized reliability indicators. In: Working Notes of ICML '03 (The 20th International Conference on Machine Learning), Workshop on The Continuum from Labeled to Unlabeled Data, pp. 72–79.

Chickering D, Heckerman D and Meek C (1997) A Bayesian approach to learning Bayesian networks with local structure. In: UAI '97, Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, pp. 80–89.

Dietterich T (2000) Ensemble methods. In: MCS '00, Proceedings of the 1st International Workshop on Multiple Classifier Systems, Springer, pp. 1–15.

Duda R, Hart P and Stork D (2001) Pattern Classification. John Wiley & Sons, Inc., New York, NY.

Dumais ST and Chen H (2000) Hierarchical classification of web content. In: SIGIR '00, Proceedings of the 23rd Annual International ACM Conference on Research and Development in Information Retrieval, pp. 256–263.

Dumais ST, Platt J, Heckerman D and Sahami M (1998) Inductive learning algorithms and representations for text categorization. In: CIKM '98, Proceedings of the 7th ACM Conference on Information and Knowledge Management, pp. 148–155.

Gama J (1998a) Combining classifiers by constructive induction. In: ECML '98, Proceedings of the 10th European Conference on Machine Learning, pp. 178–189.

Gama J (1998b) Local cascade generalization. In: ICML '98, Proceedings of the 15th International Conference on Machine Learning, pp. 206–214.

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

Hersh W, Buckley C, Leone T and Hickam D (1994) OHSUMED: An interactive retrieval evaluation and new large test collection for research. In: SIGIR '94, Proceedings of the 17th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 192–201.

Horvitz E, Breese J and Henrion M (1988) Decision theory in expert systems and artificial intelligence. International Journal of Approximate Reasoning, Special Issue on Uncertain Reasoning, 2:247–302.

Horvitz E, Jacobs A and Hovel D (1999) Attention-sensitive alerting. In: UAI '99, Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, pp. 305–313.

Hull D, Pedersen J and Schuetze H (1996) Method combination for document filtering. In: SIGIR '96, Proceedings of the 19th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 279–287.

Joachims T (1998) Text categorization with support vector machines: Learning with many relevant features. In: ECML '98, Proceedings of the 10th European Conference on Machine Learning, pp. 137–142.

Kargupta H and Chan P, Eds. (2000). Advances in Distributed and Parallel Knowledge Discovery. Cambridge, Massachusetts: AAAI Press/MIT Press.

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

Kessler B, Nunberg G and Schütze H (1997) Automatic detection of text genre. In: ACL '97, Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, pp. 32–38.

Klein LA (1999) Sensor and data fusion concepts and applications. 2nd edition. Society of Photo-Optical Instrumentation Engineers.

Lam W and Lai KY (2001) A meta-learning approach for text categorization. In: SIGIR '01, Proceedings of the 24th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 303–309.

Larkey LS and Croft WB (1996) Combining classifiers in text categorization. In: SIGIR '96, Proceedings of the 19th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 289–297.

Lewis DD (1995) A sequential algorithm for training text classifiers: Corrigendum and additional data. ACM SIGIR Forum, 29(2):13–19.

Lewis DD (1997) Reuters-21578, distribution 1.0. http://www.daviddlewis.com/resources/testcollections-/reuters21578 (visited 2002).

Lewis DD and Gale WA (1994) A sequential algorithm for training text classifiers. In: SIGIR '94, Proceedings of the 17th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 3–12.

Lewis DD, Schapire RE, Callan JP and Papka R (1996) Training algorithms for linear text classifiers. In: SIGIR '96, Proceedings of the 19th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 298–306.

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

McCallum A and Nigam K (1998) A comparison of event models for naive bayes text classification. In: Working Notes of AAAI '98 (The 15th National Conference on Artificial Intelligence), Workshop on Learning for Text Categorization, pp. 41–48.

Nigam K, Lafferty J and McCallum A (1999) Using maximum entropy for text classification. In: Working Notes of IJCAI '99 (The 16th International Joint Conference on Artificial Intelligence), Workshop on Machine Learning for Information Filtering, pp. 61–67.

Platt JC (1999a) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges C and Smola A, Eds. Advances in Kernel Methods—Support Vector Learning. MIT Press, pp. 185–208.

Platt JC (1999b) Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Smola AJ, Bartlett P, Scholkopf B and Schuurmans D, Eds. Advances in Large Margin Classifiers. MIT Press, pp. 61–74.

Provost F and Fawcett T (2001) Robust classification for imprecise environments. Machine Learning, 42:203–231.

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

Sahami M, Dumais S, Heckerman D and Horvitz E (1998) A bayesian approach to filtering junk e-mail. In: Working Notes of AAAI '98 (The 15th National Conference on Artificial Intelligence), Workshop on Learning for Text Categorization, pp. 55–62.

Schapire RE and Singer Y (2000) BoosTexter: A boosting-based system for text categorization. Machine Learning, 39:135–168.

Sebastiani F (2002) Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1–47.

Shaw J and Fox E (1995) Combination of multiple searches. In: TREC-3, Proceedings of the 3rd Text Retrieval Conference, pp. 105–108.

Ting K and Witten I (1999) Issues in stacked generalization. Journal of Artificial Intelligence Research, 10:271–289.

Toyama K and Horvitz E (2000) Bayesian modality fusion: Probabilistic integration of multiple vision algorithms for head tracking. In: ACCV 2000, Proceedings of the 4th Asian Conference on Computer Vision.

van Rijsbergen CJ (1979) Information Retrieval. Butterworths, London.

Weiss S, Apte C, Damerau F, Johnson D, Oles F, Goetz T and Hampp T (1999) Maximizing text-mining performance. IEEE Intelligent Systems, 14(4):63–69.

WinMine Toolkit v1.0, http://research.microsoft.com/~dmax/WinMine/ContactInfo.html (visited 2002). Microsoft Corporation.

Wolpert DH (1992) Stacked generalization. Neural Networks, 5:241–259.

Yang Y, Ault T and Pierce T (2000) Combining multiple learning strategies for effective cross validation. In: ICML '00, Proceedings of the 17th International Conference on Machine Learning, pp. 1167–1182.

Yang Y and Liu X (1999) A re-examination of text categorization methods. In: SIGIR '99, Proceedings of the 22nd Annual International ACM Conference on Research and Development in Information Retrieval, pp. 42–49.

Zhang T and Oles FJ (2001) Text categorization based on regularized linear classification methods. Information Retrieval, 4:5–31.