**Probabilistic approaches for classifier combination:**

We worked to keep the representations for the base classifiers analyzed in our experiments nearly identical so as to isolate the benefits gained solely from the probabilistic combination of classifiers with reliability indicators. We would expect that varying the representations (*i.e.*, using different feature selection methods or document representations) would only improve the performance as this would likely de-correlate the performance of the base classifiers. We selected four classifiers that have been used traditionally for text classification: decision trees, linear SVMs, na¨ıve Bayes, and a unigram classifier. For the decision-tree implementation, we employed the WinMine decision networks toolkit and refer to this as *Dnet* below [5]. Dnet builds decision trees using a Bayesian machine learning algorithm [4, 8]. While this toolkit is targeted primarily at building models that provide probability estimates, we found that Dnet models usually perform acceptably on error rate. However, we found that the performance of Dnet with regard to other measures is sometimes poor. For linear SVMs, we use the *Smox* toolkit which is based on Platt’s **S**equential **M**inimal **O**ptimization algorithm. After experimenting with a binary and a continuous model, we used a continuous model as it seemed to perform at approximately the same level. The *na¨ıve 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 na¨ıve Bayes classifier. Probability estimates are smoothed in a similar fashion to smoothing in na¨ıve Bayes classifier.

***Best combination methods****:* We performed experiments to explore a variety of combination methods. We considered several different combination procedures. The first combination method is based on selecting one classifier for each binary class problem, based on the one that performed best for a validation set. We refer to this method as the *Best By Class* method. Another combination method is based on taking a majority vote of the base classifiers. This approach is perhaps the most popular combination methodology. When performing a majority vote, ties can be broken in a variety of ways (*e.g.*, breaking ties by always voting for *in class*). We experimented with several variants of this method, but we only present results here for the method which relies on breaking ties by voting with the *Best By Class* classifier because it nearly always outperformed the other majority vote methods. We refer to this method as *Majority BBC*.

**Hierarchical Combination Methods:**

***Stacking***

Finally, we investigate several variants of the hierarchical models described earlier. As mentioned above, omitting the reliability indicator variables is equivalent to stacking [25, 29]. We refer to these classifiers below as Stack-X where X is replaced by the first letter of the classifier that is performing the meta-classification. Therefore, Stack-D uses a decision tree as the meta-classifier, and Stack-S uses a linear SVM as the meta-classifier. It should be noted that Stack-S is also a weighted linear combination method since it is based on a linear SVM and uses only the classifier outputs. It can be problematic to learn the weights for an SVM when the inputs have vastly different scales (in addition it may not be possible to pick good weights); therefore, we normalize the inputs to the meta-classifiers to zero mean and unit standard deviation. In order to perform consistent comparisons, we perform the same alteration for the meta-classifiers using Dnet. We also give for one of the Dnet variants the results without performing normalization; as would be expected the impact of normalization for decision-tree learners is relatively minimal (and has both positive and negative influences).

***Strive***

Similar to the notation described above, we add a letter to Strive to denote the meta-classifier being used. So, Strive-D is the Strive framework using Dnet as a meta-classifier. For comparison to the stacking methods, we evaluate Strive-D and Strive-S. Normalizations noted in the same way. The experiments reported here use a total of 49 reliability indicators (including those specific examples given in Section 3.1). These variables were simply our initial pass at representing appropriate information. In the future, we intend to publish an analysis of which variables are most useful, in addition to extending the set of variables currently employed.

**Recognition plausibility enhancement via a dictionary of samples:**

**Distance among strings:**