A NOVEL TWO STAGE SCHEME UTILIZING THE TEST SET FOR MODEL SELECTION IN TEXT CLASSIFICATION

Bernhard Pfahringer (CS Dept. University of Waikato, Hamilton, New Zealand; bernhard@cs.waikato.ac.nz); Peter Reutemann (CS Dept. University of Waikato, Hamilton, New Zealand; fracpete@cs.waikato.ac.nz); and Mike Mayo (CS Dept. University of Waikato, Hamilton, New Zealand; mmayo@cs.waikato.ac.nz)

ABSTRACT

Text classification is a natural application domain for semi-supervised learning, as labeling documents is expensive, but on the other hand usually an abundance of unlabeled documents is available. We describe a novel simple two-stage scheme based on dagging which allows for utilizing the test set in model selection. The dagging ensemble can also be used by itself instead of the original classifier. We evaluate the performance of a meta classifier choosing between various base learners and their respective dagging ensembles. The selection process seems to perform robustly especially for small percentages of available labels for training.

1. INTRODUCTION

One of the novel non-standard learning trends emerging in recent years is so-called semi-supervised learning, where algorithms in addition to a standard labeled training set also have access to additional unlabeled data points. Various sophisticated schemes have been invented trying to extract useful information from this additional data. Such approaches include co-training [2, 10], transductive learning [5], and various methods based on extracting cluster structures, either explicitly [3] or implicitly [15, 4]. This paper investigates a rather different and simple idea in the context of text mining. Text mining is an obvious application area for semi-supervised learning, as there is an abundance of text available electronically, most of which does not come with explicit labels. Labeling text in itself is a costly procedure, usually requiring a human with the appropriate expertise. So any automatic gain achievable from unlabeled text is most welcome.

In this paper we investigate ways of utilizing a full test-set, for which predictions are sought, for selecting a well performing classifier. The next section will develop and explain a two-stage approach to semi-supervised classification. In section 3 the experimental design and hypothesis are described, and results of the experiments are presented and discussed. Finally Section 4 gives conclusions and directions for future work.

2. TWO STAGE ESTIMATION

The standard approach for model selection in Machine Learning uses estimated error rates for the various models and then chooses the best one (or alternatively the simplest model whose error is close enough to the error rate of the best model). Standard procedures for estimation are either splitting the labeled data into a train and a validation set, or cross-validation. The latter is the preferred approach for smaller datasets.

For small amounts of data (e.g. assuming only 5% of the given data is labeled) one might expect the cross-validation procedure to show a bias towards simpler models, which potentially might underfit the data, as more complex patterns may not by frequent enough to both be picked up by the learner in the training set and to be present in high enough numbers in the respective test fold simultaneously. Therefore in such situations cross-validation might not be able to robustly select good models. As an alternative we have devised the following two stage procedure:

1. Generate classifier C_1 using all the labeled data. Use C_1 to label the unlabeled data.
2. Generate classifier C_2 using all the data labeled by C_1 (i.e. the original test-set with estimated labels). Apply C_2 to the original training set and return this error rate as the estimate for the particular learning algorithm.

The rationale for this estimation procedure is as follows: for successful learning the same true patterns should be present in both the training and the test set, which is a fundamental assumption used generally in Machine Learning. So ideally one would expect the two classifiers C_1 and C_2 generated above to be rather similar, if not identical. And contrary to cross-validation all the labeled data can be used for inducing C_1. Furthermore C_2 is evaluated against the given labeled

data, which should lead to good estimates, as these labels should be more or less correct.

Unfortunately in preliminary tests we found that this procedure does not seem to work well, except for larger datasets and a large enough percentage of labeled data. The estimates seemed to have a rather high variance and as such could not be used to choose good models reliably. Additionally computational problems were encountered as well. C_2 is induced from a considerably larger set of data than C_1, e.g. 19 times as many instances are seen by C_2 if only 5% of the data carries original labels. As well-performing learning algorithms usually exhibit non-linear runtime complexity and potentially also non-linear space requirements, inducing C_2 can be rather time-consuming, and occasionally might not be feasible at all.

A third and more subtle problem arises when model selection involves parameter tuning of single algorithms as well. For some parameters the optimal value varies with the size of the training set (e.g. the bandwidth parameter used with RBF-kernels), and therefore a good value for C_1 might not perform as well for C_2 and vice versa.

To counter all these problems we resort to a rarely used ensemble method called dagging [13]. Dagging was originally invented for scenarios where training data was naturally coming from different sources, e.g. resulting from different locations or time-spans being used for data collection. In such scenarios dagging has been shown to perform well when data was plentiful. We utilize dagging in the following way: instead of generating one classifier C_2 using all the test data as labeled by C_1, the test data is split up into multiple batches of about equal size to the original training set. For each of this batches one classifier is trained. Contrary to [13] we use simple equal-weighted voting for prediction. The resulting ensemble is applied to the original training set to compute an estimate of the ensemble's predictive performance. This slightly more complex stage two classifier has the following theoretical advantages over the simpler approach described above:

1. As the batch sizes are more or less equal to the size of the original training set, computational complexity issues disappear. If it was feasible to induce a classifier on the original training set, then it should be feasible as well to induce one classifier for each of the equal-sized batches. The total complexity for stage two is simply *O(numberOfBatches * O(stageOne))*. Given that *numberOfBatches* is usually considerably smaller than the total number of test examples, there should be no problem with either memory consumption or runtime. Likewise the issue of tuning parameters whose optimal values vary with data size should vanish.
2. More importantly, as the predictions of the second stage are now votes across an ensemble of classifiers, one would expect to see reduction in variance. There is an additional reason why we should expect such a reduction: in the simple setup with only one stage two classifier every wrongly labeled test-example (i.e. all the errors that C_1 commits) can directly impact on the performance of the single stage two classifier C_2. In the dagging setup each such wrongly labeled example only impacts on exactly one classifier. Therefore bad performance of one these classifiers in one area of prediction can possibly be compensated for by the votes of all the classifiers which did not have to try to cover the mis-labeled example.

The experiments reported below seem to indicate that these expectations are fulfilled in practise. This is interesting from a dagging point of view as well, as so far dagging usually has only been employed with a small number of potentially quite diverse batches, whereas we use sometimes considerably larger number of batches of data, but on the other hand these batches are more uniform.

3. EXPERIMENTAL DESIGN AND RESULTS

To evaluate the dagging-based idea developed in the previous section we have conducted experiments using the well-known Reuters Corpus [1]. Only the ten largest categories were used and each experiment was a two-class problem, predicting whether a news wire article belongs to a respective topic or not. The textual data was preprocessed in a fairly standard way using the StringToWordVector filter supplied by WEKA [14]. The filter was used in the class-sensitive setting choosing the 1000 most frequent words for each class, no stemming was performed, the default English stop list supplied by WEKA was used, the generated attributes were counts of word-occurrences processed by the standard TFIDF procedure, and finally all counts across a document were normalized to average document length. This type of preprocessing has been found to perform well across a range of text classification tasks by various authors [6, 8]. As we use a class-sensitive approach, only the training set is used to build a filter, which is then applied to both the training and the test set. Otherwise class-information from the test set could leak through into the learning process resulting in too optimistic estimates.

Contrary to the usual predefined train/test splits we split each set randomly into a small train and a much larger test set. Training set sizes of 5%, 10% and 20% were used, and for each setting a 100 repetitions were performed. We were interested into the following questions:

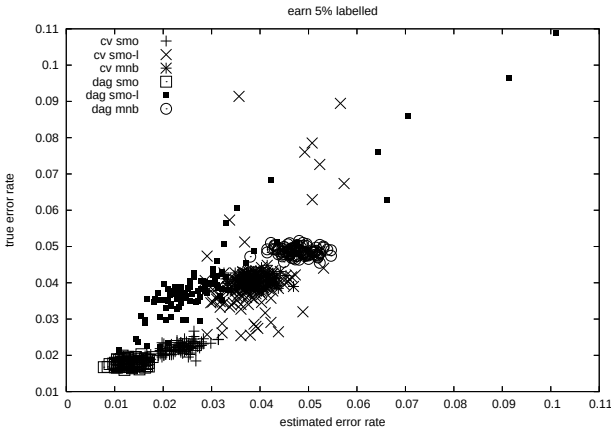

Figure 1. Correlation of estimated to true error rates for the "earn" topic with 5% labeled data.

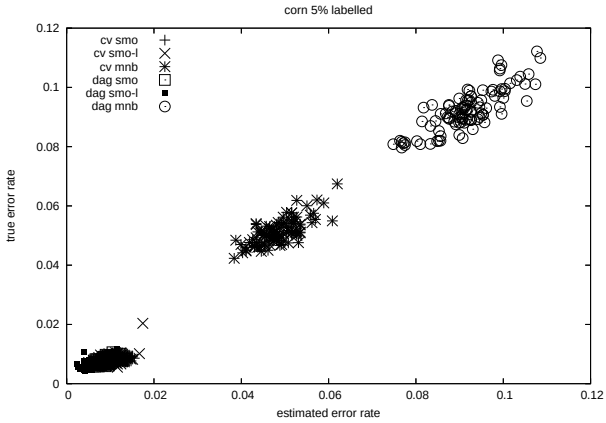

Figure 2. Correlation of estimated to true error rates for the "corn" topic with 5% labeled data.

1. How well does a 10fold cross-validation estimate on the training set correlate with the true error on the test set?
2. How well does the error of the dagging ensemble on the original training set correlate with the true error on the test set?
3. Can we use these estimates (10fold cross-validation and training set error of the dagged test ensemble) to reliably choose between multiple classifiers and also between the training set version of a classifier and its dagged test ensemble version?

To answer these questions we have employed some standard text classification algorithms: a multinomial Naive Bayes learner optimized for sparse data (Weka's NaiveBayesMultinomial), and a linear support vector machine optimized for sparse data (WEKA's SMO algorithm), either with the "-M" option on or off. If this option is active, the raw output of the support vector machine is used as the sole input for a logistic regression which allows for proper probability estimation [11]. Additionally this second stage can also de facto move the decision threshold of the support vector machine, a property we have found useful in a lot of experiments involving non-textual data.

For a given train/test split, the following computations were performed:

- Compute a cross-validation estimate for each classifier on the training set.
- Compute a single classifier from the full training set and record its error rate on the test data.
- Use the single classifier to label the test set, induce the dagging ensemble from this now labeled test set, and record the error rate of the dagging ensemble on the original training set.
- Compute the error of the dagging ensemble with respect to the true labeling of the test set.

Using these measurements we can answer the correlation questions 1 and 2 formulated above, as well as Question 3, which is the most interesting one and also of most practical value. Regarding the first two questions, we usually find a good positive correlation. Figures 1, 2, and 3 illustrate example correlations. Figure 1 depicts the performance of all six classifiers for the "earn" topic, which is the largest one. On the x axis we plot the respective estimate, and on the y axis we plot the true error rate on the test set for each of the 100 random train/test splits. We can observe a very good correlation, and also notice that for this category "dag smo" seems to be the best choice on average. Figure 2 repeats the same setup for the smallest category, which is "corn". This picture is much more extreme, and clearly shows that for this topic a support vector machine outperforms multinomial naive Bayes, and also that the dagging variant of multinomial naive Bayes performs worst. All the support vector machine results are too close to each other in this figure, therefore we have enlarged this area in Figure 3: dagged SMO with logistic post-processing is clearly dominating all other variants.

To answer Question 3, we compute the following simple meta-classifier: assume that for all three classifiers (multinomial Naive Bayes, and linear support vector machine with or without logistic regression-based post-processing) both a cross-validation estimate as well as the dagging-ensemble estimate have been computed. Then simply select the algorithm and setup with the smallest estimated error; ties are broken at random. In Tables 2,3, and 4 we compare the performance of this algorithm to the performance of the best single algorithm, and also to the second best. Notice that the meta classifier is a reasonable setup for practical prediction, but that the "best" and

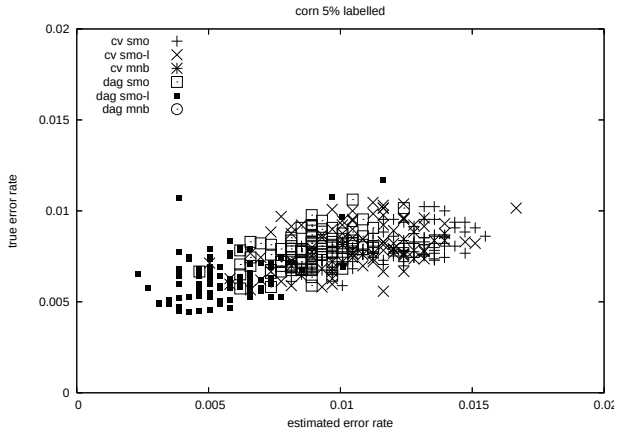

Figure 3. Correlation of estimated to true error rates for the "corn" topic with 5% labeled data, zoomed in into the best region.

Table 1. The percentage of documents for each topic, which is also the default accuracy achievable by always predicting *not(Topic)*.

Topic	Percent	Topic	Percent
acq	18.36	interest	3.70
corn	1.84	money-fx	5.56
crude	4.48	ship	2.22
earn	30.72	trade	3.77
grain	4.51	wheat	2.19

only, which would not be available in practise. Most differences are significant using a sign test over the 100 iterations each. We are using a sign test for lack of better alternatives. Corrected t-tests like the one described in [9] unfortunately only apply to cross-validation and training set sizes between 50% and 90%. Their corrections therefore do not apply to the more extreme settings used here.

To help interpreting the results we provide a listing of the default accuracies for each topic in Table 1. Using simple accuracies is somewhat unusual in text classification, as quite often misclassification costs are non-uniform; for instance in case of spam classification false-positives are a lot worse than false-negatives. Ideally one would use a threshold-independent measure like AUC [12] for comparison, but this is currently not well integrated in Weka. Therefore we have resorted to simple error rates, which seem to be sufficient for this experiment. Proper AUC evaluation is part of our future work plan.

Inspecting table 2 which lists the results for the 5% training and 95% test split, we notice that for all topics a dagging ensemble over the test-set yields the theoretically best performance. The best base level learner is always a support vector machine, and except for one topic it is the naive version, which contradicts our experience with non-textual data. Even though NaiveBayesMultinomial does not win for any of the topics, and therefore does not show up in Table 2, it is still the best classifier occasionally for a specific train/test split. The meta-classifier can usually select a good if not the best classifier, therefore we note that the meta-classifier always performs better than the second best single classifier, and even outperforms the best single classifier for five of the ten topics. If we look at the number of times the meta-classifier does not select the best single classifier for a specific train/test split, we notice that this number is weakly correlated with the relative size of the topic. The best result of only one wrong selection is achieved for "earn" which is the largest topic, and the worst

Table 2. For each of the topics and for 100 random 5% training, 95% test splits, we list how often the best classifier was missed (Wrong), the actual error rate of the chosen classifier (Error), the (theoretical) error rate of the second best classifier (2nd), and of the best classifier (1st), and the name of the best classifier.

Topic	Wrong	True Error	2nd Est.Error	1st Est.Error	Best Algorithm
acq	16	2.920	3.763	2.882	dag SMO
corn	11	0.628	0.778	0.635	dag SMO-L
crude	26	1.321	1.499	1.331	dag SMO
earn	1	1.791	2.199	1.788	dag SMO
grain	5	0.840	0.967	0.839	dag SMO
interest	13	1.824	2.052	1.786	dag SMO
money-fx	5	2.020	2.355	2.027	dag SMO
ship	42	0.995	1.058	1.054	dag SMO
trade	14	1.506	1.745	1.499	dag SMO
wheat	23	0.637	0.727	0.643	dag SMO

Table 3. For each of the topics and for 100 random 10% training, 90% test splits, we list how often the best classifier was missed (Wrong), the actual error rate of the chosen classifier (Error), the (theoretical) error rate of the second best classifier (2nd), and of the best classifier (1st), and the name of the best classifier.

Topic	Wrong	True Error	2nd Est.Error	1st Est.Error	Best Algorithm
acq	0	3.048	3.980	3.048	dag SMO
corn	76	0.987	0.994	0.909	cv SMO-L
crude	60	1.866	1.987	1.777	cv SMO
earn	84	3.039	2.318	2.137	dag SMO
grain	25	1.199	1.294	1.285	cv SMO-L
interest	32	2.129	2.275	2.233	cv SMO
money-fx	78	2.831	2.809	2.597	cv SMO
ship	87	1.190	1.167	1.030	cv SMO-L
trade	8	1.542	1.939	1.543	dag SMO-L
wheat	11	0.640	0.732	0.640	dag SMO-L

result (42 wrong selections out of 100) is achieved for "ship", one of the smaller topics. But even in these mis-selection cases the meta-classifier usually seems to at least choose a reasonable "runner-up" algorithm, thus still achieving high overall performance.

Tables 3 and 4 list the same results for the 10% and 20% training size cases respectively. These results are subtly different to the ones shown in Table 2. Suddenly the sole classifier trained on the full training set (as indicated by "cv" in these tables) is the theoretically best one for six of the ten topic. Also post-processing SMO outputs by logistic regression (indicates as "SMO-L") seems much more important here. Looking at the mis-selection rates for the meta-classifier we note that these are much higher now, reaching 87 out of 100 in the worst case. Consequently, the performance of the meta-classifier is also worse, for three topics it even cannot outperform the theoretically second best classifier.

Even more concerning is the fact that the results actually seem to degrade for larger percentages of known labels. For instance, the meta-classifier's performance on the "acq" topic is 2.920% for 5% training, 3.048% for 10% training, and 3.788% for 20% training. We have not yet found a satisfactory explanation for this anomaly. Our current hypothesis is two-fold: smaller training percentages lead to smaller batch sizes for the dagging ensemble, and therefore to more classifiers in the dagging ensemble (19 for 5%, 9 for 10%, and only 4 for 20%). Voting might perform

Table 4. For each of the topics and for 100 random 20% training, 80% test splits, we list how often the best classifier was missed (Wrong), the actual error rate of the chosen classifier (Error), the (theoretical) error rate of the second best classifier (2nd), and of the best classifier (1st), and the name of the best classifier.

Topic	Wrong	True Error	2nd Est.Error	1st Est.Error	Best Algorithm
acq	6	3.788	4.435	3.745	dag SMO
corn	21	1.257	1.442	1.254	dag SMO-L
crude	38	2.226	3.111	2.331	cv SMO
earn	82	2.988	2.948	2.649	cv SMO
grain	63	2.679	2.985	2.044	cv SMO
interest	71	2.575	2.698	2.524	cv SMO
money-fx	61	3.256	3.163	3.066	cv SMO
ship	10	1.348	2.170	1.699	cv SMO
trade	23	1.962	2.542	2.314	dag SMO-L
wheat	30	1.088	1.254	1.084	dag SMO-L

more robustly for the larger ensembles. Additionally the single classifier constructed from the full training set might be becoming more competitive to the dagging ensemble for larger percentages. With estimates being closer to each other the selection process might commit more errors as well.

4. CONCLUSIONS

We have introduced a simple new scheme to exploit the test set for model selection in text classification. The new scheme is a two stage process which first trains a classifier on the training set, uses this classifier to label the test set, then induces a dagging ensemble on the labeled test set and evaluates this ensemble with respect to the training set. In the experiments conducted this estimation appears to be reliable enough to be able to select between different algorithms and also between the original classifier and the dagging ensemble as the final classifier to be used, at least in the more extreme setup where only 5% of the labels are available for learning. The higher percentage cases exhibited a few anomalies which are currently being investigated.

There are quite a few more directions for future work. First and most importantly more different and larger text datasets need to be explored, a particular promising source should be the new well-processed corpus of news wire articles available from Reuters [7] again. Secondly the hypothesis that it is actually the larger number of batches that are used in the 5% setup than in the 10% or 20% setups which causes better overall performance needs to be evaluated. Thirdly we want to switch to AUC [12] as a measure for performance evaluation, as AUC is independent of the setting of particular thresholds in classification, an important property for learning with skewed class distributions commonly encountered in text classification.

5. ACKNOWLEDGMENTS

This work has been funded by a Marsden Grant of the Royal Society of New Zealand.

6. REFERENCES

[1] Chidanand Apté, Fred Damerau, and Sholom M. Weiss, "Automated learning of decision rules for text categorization," *Information Systems*, 12(3):233-251, 1994.

[2] Avrim Blum and Tom Mitchell, "Combining labeled and unlabeled data with cotraining," In *COLT: Proceedings of the Workshop on Computational Learning Theory*, Morgan Kaufmann Publishers, pages 92-100, 1998.

[3] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," volume 15 of *NIPS*, 2003.

[4] T. Joachims, "Transductive learning via spectral graph partitioning," In *Proceedings of The Twentieth International Conference on Machine Learning*, 2003.

[5] Thorsten Joachims, "Transductive inference for text classification using support vector machines," In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML- 99, 16th International Conference on Machine Learning*, pages 200-209, Bled, SL, Morgan Kaufmann Publishers, San Francisco, US, 1999.

[6] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, "Multinomial naive bayes for text categorization revisited," In *Seventeenth Australian Joint Conference on Artificial Intelligence*, pages 488-499, 2004.

[7] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research* (JMLR), 5:361-397, 2004.

[8] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[9] Claude Nadeau and Yoshua Bengio, "Inference for the generalization error," *Mach. Learn.*, 52(3):239-281, 2003.

[10] Kamal Nigam and Rayid Ghani, "Analyzing the effectiveness and applicability of co-training," In *CIKM*, pages 86-93, 2000.

[11] J.C. Platt, "Probabilities for SV machines," In *Advances in Large Margin Classifiers*, pages 61-74, 1999.

[12] Foster Provost and Tom Fawcett, "Robust classification for imprecise environments," *Machine Learning*, 42(3):203-231, 2001.

[13] Kai Ming Ting and Boon Toh Low, "Model combination in the multiple-databatches scenario," In *European Conference on Machine Learning*, pages 250-265, 1997.

[14] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2005.

[15] D. Zhou, O. Bousquet, T. Lal, J.Weston, and B. Schölkopf, "Learning with local and global consistency," In *18th Annual Conf. on Neural Information Processing Systems*, 2003.