Heterogeneous Uncertainty Sampling for Supervised Learning

David D. Lewis and Jason Catlett
AT&T Bell Laboratories
Murray Hill, NJ 07974
lewis@research.att.com, catlett@research.att.com

Appeared (with same pagination) in William W. Cohen and Haym Hirsh, eds.,
Machine Learning: Proceedings of the Eleventh International Conference,
Morgan Kaufmann Publishers, San Francisco, CA, pp. 148–156.

Abstract

Uncertainty sampling methods iteratively request class labels for training instances whose classes are uncertain despite the previous labeled instances. These methods can greatly reduce the number of instances that an expert need label. One problem with this approach is that the classifier best suited for an application may be too expensive to train or use during the selection of instances. We test the use of one classifier (a highly efficient probabilistic one) to select examples for training another (the C4.5 rule induction program). Despite being chosen by this *heterogeneous* approach, the uncertainty samples yielded classifiers with lower error rates than random samples ten times larger.

1 Introduction

Machine learning algorithms have been used to build classification rules from data sets consisting of hundreds of thousands of instances [4]. In some applications unlabeled training instances are abundant but the cost of labeling an instance with its class is high. In the information retrieval application described here the class labels are assigned by a human, but they could also be assigned by a computer simulation [2] or a combination of both [30]. The terms *oracle* and *teacher* have been used for the source of labels; we will usually call it the *expert*.

Where one of the constraints on the induction process is a limit on the number of instances presented to the oracle, the choice of instances becomes important. Random sampling [5] may be ineffective if one class is very rare: all of the training instances presented may have the majority class. To make more effective use of the expert's time, methods that we collectively call *uncertainty sampling* label data sets incrementally, alternating between two phases: presenting the expert a few instances to label, and selecting (from a finite or infinite source) instances whose labels are still uncertain despite the indications contained in previously labeled data.

The type of classifier used in uncertainty sampling must be cheap to build and to use. At each iteration a new classifier is built (fortunately from a small sample) and then applied (unfortunately to a large sample). Our uncertainty sampling method also requires an estimate of the certainty of classifications (a class-probability value) [28]; not all induction systems provide this.

This paper examines a *heterogeneous* approach in which a classifier of one type selects instances for training a classifier of another type. It is motivated by applications requiring a type of classifier that would be too computationally expensive to use to select instances. Section 2 reviews research on uncertainty sampling. Section 3 points out that the class frequencies in uncertainty samples are severely distorted; the training algorithm should accept some parameter to correct for this. The experiments described in Section 6, on a large text categorization data set, showed our method for this correction to be effective and robust with respect to the particular parameter value used. Uncertainty samples chosen by a probabilistic classifier were found to be significantly better than random samples ten times larger when used by a modification of Quinlan's C4.5 algorithm. Section 9 lists several opportunites for future work.

2 Background

Theoretical analysis and practical experience have shown that a classifier can often be built from fewer instances if the learning algorithm is allowed to create artificial instances or *membership queries* that are given to an expert to label [1, 25]. Unfortunately such queries may create nonsensical examples: is a pregnant non-smoking male at high risk for heart disease? In applications where instances are images or natural language texts, arbitrary membership queries are also implausible.

Several algorithms have been proposed that base querying on filtering a stream of unlabeled instances rather than on creating artificial instances [6, 10, 20, 31]. The expert is asked to label only those instances whose class membership is sufficiently uncertain. Several definitions of uncertainty and sufficiency have been used, but all are based on esti-

```
1. Obtain an initial classifier
2. While expert is willing to label instances
   (a) Apply the current classifier to each unlabeled instance
   (b) Find the $b$ instances for which the classifier is least certain of class membership
   (c) Have the expert label the subsample of $b$ instances
   (d) Train a new classifier on all labeled instances
```

Figure 1: An algorithm for uncertainty sampling from a finite training set using a single classifier.

mating how likely a classifier consistent with the previously labeled data would be to produce the correct class label for a given unlabeled instance. These approaches can be viewed as a combination of stratified and sequential approaches to sampling [5, 32], so we refer to them as *uncertainty sampling*.

A simple form of uncertainty sampling is possible for classifiers that operate by testing a numeric score against a threshold. A single classifier is trained, and those instances whose scores are closest to that classifier's threshold are good candidates to present to the expert. Where the set of instances is finite, the single instance with a score closest to the threshold can be found; where the stream of instances is effectively infinite, one can choose instances whose scores are within some distance of the threshold. The cycle is described in Figure 1 for the finite case.

Single classifier approaches to uncertainty sampling have been criticized [6, 20] on the grounds that one classifier is not representative of the set of all classifiers consistent with the labeled data: the *version space* [24]. The degree to which this is a problem in practice has not been established.

Single classifier approaches have successfully been used in generating arbitrary queries [16] and in sampling from labeled data [8, 25]. Uncertainty sampling with a single classifier can also be viewed as a variation on the heuristic of training on misclassified instances [15, 33, 35]. A familiar example of this is *windowing*, which appeared in Quinlan's first paper on ID3 [26], was questioned in [36] and re-examined in Chapter 6 of the C4.5 book [27]. As with uncertainty sampling, windowing builds a sequence of classifiers, selecting instances to add to the training set at each iteration. The key difference is its assumption that the class labels of all training instances are known: it examines them in order to choose misclassified examples to add.

A large scale test of uncertainty sampling with a single classifier approach [18] showed that uncertainty sampling could reduce by a factor of up to 500 the amount of data that had to be labeled to achieve a given level of accuracy.

3 Heterogeneous Uncertainty Sampling

Uncertainty sampling requires the construction of large numbers (perhaps thousands) of classifiers which are applied to very large numbers of examples. This suggests that the kind of classifier "in the loop" during sampling should be very cheap both to build and run.

Unfortunately, an uncertainty sample has strong connections to the classifier form used to select it: despite containing a disproportionately large number of instances from low frequency classes, it still yields an accurate classifier. Some of the characteristics of a sample that cause this to happen for one form of classifier are likely to have the same effect on others, such as overrepresentation of instances where different attribute values suggest different classes. However, this effect is unlikely to be perfect for classifiers of any form but the one used in selection. A new classifier trained on the uncertainty sample will then be unduly biased toward predicting low frequency classes.

Some mechanism to counterbalance this effect is needed. A feature of the CART [3] software for decision trees for specifying *priors* on classes could be used, but our application required decision rules. We used a version of C4.5 modified by Catlett to accept a parameter specifying the relative cost of two types of error: false positives and false negatives [9, Chapter 1]. We call this number the *loss ratio* (LR). A loss ratio of 1 indicates that the two errors have equal costs (the original assumption of C4.5). A loss ratio greater than 1 indicates that false positive errors (which a classifier built from a training set enriched with positive instances is more likely to make) are more costly than false negative errors (where a positive instance is classified negative). Setting the loss ratio above 1 will counterbalance the overrepresentation of positive instances, but exactly what figure should be used? This question motivates a sensitivity analysis of the effect of this parameter on the accuracy of classifiers produced.

The modifications to C4.5 left the selection criterion unchanged (in contrast to CART's treatment). When building trees the original C4.5 checks after each split that this decreases the error rate; otherwise it replaces the split with a leaf (`build.c`, line 347); if not disabled this preempts the construction of rules for classes with few examples. The class values at the leaves are determined not by majority vote but by comparison with a probability threshold of $LR/(LR+1)$ (or its reciprocal as appropriate). Pruning minimizes expected loss instead of estimated errors (simply multipling by LR, with the usual correction). A similar change is made to the minimum error rate required to drop a rule. The choice of default class is also based on expected loss, but the estimates of the number of examples left uncovered by any rule appeared too low, so an arbitrary factor was introduced to counterbalance this. The most problematic question is how to adapt the sifting of the rules for each class, which in C4.5 is guided by MDL principle. The current implementation simply multiplies the coding cost of either the false positives or the false negatives by

LR or $1/LR$ (as appropriate) to increase the coding cost of rulesets that make the more expensive error. Although this step lacks a theoretical justification, performance appeared satisfactory.

4 Task and Data Set

The applications motivating this research fall under the heading of *text categorization*: the classification of instances composed partly or fully of natural language text into pre-specified categories [7, 19]. We have found several business applications where categorizing text would aid its use, routing, or analysis.

These texts often reside in large databases supporting *boolean queries* [29, pages 231–236], a restricted version of propositional logic. Because decision rules [27, 34] can be converted into this form (unlike probabilistic models requiring arithmetic), they make a good choice for the final classifier. Another important advantage is that they can more comprehensible to humans than decision trees [4]. Our databases contain hundreds of thousands of unlabeled instances, so uncertainty sampling is a natural approach. However, as discussed in Section 5, our current decision rule induction software cannot practicably be used for uncertainty sampling from large text databases. We therefore decided to test a heterogeneous approach to uncertainty sampling.

Given that a key aim of the research is to reduce the time spent by human experts categorizing texts, we could hardly ask them to label a hundred thousand instances for the sake of our experiments. Instead we used a data set with similar properties to those in our applications: the titles of stories categorized by a news agency. We collected in electronic form the titles of 371,454 articles that appeared on the AP newswire between 1988 and early 1993. We divided these randomly into a training set of 319,463 titles and a test set of 51,991 titles.[1]

Titles were converted to lower case and punctuation was removed. Each distinct word was treated as a binary attribute, resulting in 67,331 attributes. The data matrix was therefore extremely sparse, with each instance having an average of 8.9 nonzero attribute values.

The AP data is labeled with several types of subject categories. We defined ten binary categories of AP titles based on the "keyword slug line" from the article [13, page 317]. Frequency information on these categories is given in Table 1. The categories were chosen to resemble the applications of interest to us: approximately one in five hundred instances are positive; the classes are somewhat noisy, and cannnot be perfectly determined from the text.

	Training		Test	
Category	Number	Percent	Number	Percent
tickertalk	208	0.07	40	0.08
boxoffice	314	0.10	42	0.08
bonds	470	0.15	60	0.12
nielsens	511	0.16	87	0.17
burma	510	0.16	93	0.18
dukakis	642	0.20	107	0.21
ireland	780	0.24	117	0.23
quayle	786	0.25	133	0.26
budget	1176	0.37	197	0.38
hostages	1560	0.49	228	0.44

Table 1: The ten categories used in our experiments, with the number and percentage of positive occurrences on training and test sets.

5 Training C4.5 with Text Data

Although we used a modification of Quinlan's C4.5 software [27] to produce decision rules from the training data, using it to select examples is impractical for large text databases because it requires that training and test instances be presented as tuples specifying the values of all attributes. With 319,463 training instances and 67,331 attributes this would have required over 40 gigabytes. The extravagance of expanding such spase data was stressed in [22]. The C4.5 algorithm could be implemented in a manner suited to sparse data, but almost no machine learning software has this feature. Even eliminating attributes that take on the value *True* less than five times in the training data would still have left 24,052 attributes, at the cost of eliminating some useful attributes. Feature selection methods requiring class labels are not a solution because most labels are unknown.

5.1 Uncertainty Sampling with a Probabilistic Classifier

Methods for efficient training of probabilistic classifiers from large, sparse data sets are widely used in information retrieval [14]. We used this type of classifier to select instances in uncertainty sampling. The model is described in detail elsewhere [18], but in brief it gives the following estimate for the probability that an instance belongs to class C:

$$P(C|\mathbf{w}) = \frac{\exp(a + b\sum_{i=1}^{d} \log \frac{P(w_i|C)}{P(w_i|C)})}{1 + \exp(a + b\sum_{i=1}^{d} \log \frac{P(w_i|C)}{P(w_i|C)})}. \qquad (1)$$

C indicates class membership, and w_i is the ith of d attribute values in the vector **w** for an instance. The instance is assigned to class C if $P(C|\mathbf{w})$ exceeds 0.5.

The intuition behind the model is that

[1] Stories were randomly allocated to the test set with probability 0.14. Our goal was a test set with at least 40 to 50 positive instances of each category.

$$\prod_{i=1}^{d} \frac{P(w_i|C)}{P(w_i|\bar{C})} \qquad (2)$$

is a plausible approximation (exact if certain independence assumptions and class priors hold) to the likelihood ratio

$$\frac{P(\mathbf{w}|C)}{P(\mathbf{w}|\bar{C})} \qquad (3)$$

and so is a good predictor of class membership. However, it must be scaled to provide an explicit estimate of $P(C|\mathbf{w})$. One approach to this scaling is logistic regression [23].

Training proceeds as follows. The values $P(w_i|C)$ and $P(w_i|\bar{C})$, as well as $P(w_i)$ are estimated for every word w_i. This estimation uses a sparse representation of the data and requires only a few seconds for several hundred thousand training instances. Those w_i's with large values of $P(w_i) \times \log P(w_i|C)/P(w_i|\bar{C})$ were selected as features, a strategy found useful in other text classification problems [11]. The value $\prod_{i=1}^{d} \frac{P(w_i|C)}{P(w_i|\bar{C})}$ is then computed for each training instance, and the training data is used again to set a and b by a logistic regression.[2]

A classifier of this sort was trained on each iteration of uncertainty sampling, and was then applied to all unlabeled training instances. The two instances with the estimated $P(C|\mathbf{w})$'s closest to, but above, 0.5 were selected, as well as the two instances with $P(C|\mathbf{w})$'s closest to, but below 0.5. Using a subsample size of four rather than one was a compromise for efficiency. Selecting examples both above and below 0.5 was a simple way to halve the potential number of duplicate examples, and may also have benefits for training [16].

5.2 Initial Classifier

Without an initial classifier our sampling algorithm would commence with a long period of nearly random sampling before finding any examples of a low frequency class. Obtaining a plausible initial classifier is usually easy—it would be surprising if an expert were able to classify instances but could not suggest either some positive and negative instances, or some attribute values correlated with the class. For our experiments we instead generated initial classifiers from three positive instances of the category, selected randomly to avoid experimenter bias.

5.3 Feature Selection

The cost of specifying the values of all 67,331 attributes for even a small training set is so large that some feature selection was needed before presenting any data to C4.5. We used the union of the following sets of words as attributes:

1. all words occurring in at least 0.2% of the instances,
2. all words occurring in two or more positive instances, and
3. all words occuring in one or more of the three initial positive instances.

6 Experiment Design

Our experiment tested whether heterogeneous uncertainty sampling would produce decision rules with significantly lower error rates than those trained on random samples of the same or even larger size. We also wanted to determine the sensitivity of the rules' accuracy to the loss ratio used with C4.5. Sources of variability included the categories, quality of starting classifiers, and the vagaries of random sampling.

We repeated the uncertainty sampling process 100 times, 10 trials on each of 10 binary categories, each with a different random set of three initial positive instances. On each run, the three initial instances were used to build an initial classifier, after which uncertainty sampling with a subsample size of four was run for 249 iterations. This yielded 100 groups of 250 uncertainty samples of various sizes. We trained C4.5 rules on two uncertainty samples from each run, one with 299 instances and the final one (999 instances). Values of 1 to 20 for the loss ratio (that is, the ratio of loss incurred for false positives to loss incurred for false negatives) were tested.

As a comparison, C4.5 was applied to samples of size 1,000 and 10,000 produced by adding random instances to the same sets of three starting positive instances used to initialize uncertainty sampling. (The starting positive instances are retained to make the comparison more fair to random sampling.) The samples of size 10,000 were produced by adding additional random examples to the samples of size 1,000. We refer to all these samples as "random", though they are not completely random. Most of the analyses below use the samples of size 10,000.

We were also interested in the difference in accuracy compared to using C4.5 in the instance selection loop. Although it was not practicable to test this directly, we did train probabilistic classifiers on both the uncertainty and random samples to provide some comparison.

7 Results

Figure 2 shows average error rates for C4.5 rules trained with uncertainty samples of size 299 and 999 and various loss ratios, for each of nine categories. (The tenth category, *tickertalk*, resulted in degenerate classifiers—all instances classified as category nonmembers—under almost all conditions.) In all cases, error rates for uncertainty samples of size 999 are close to or better than those for a random sample of 10,000 instances, provided a loss ratio of three or more is used.

[2] Alternatively, one could view this as a one-node neural net with the input weights set via a probabilistic model rather than by error propagation.

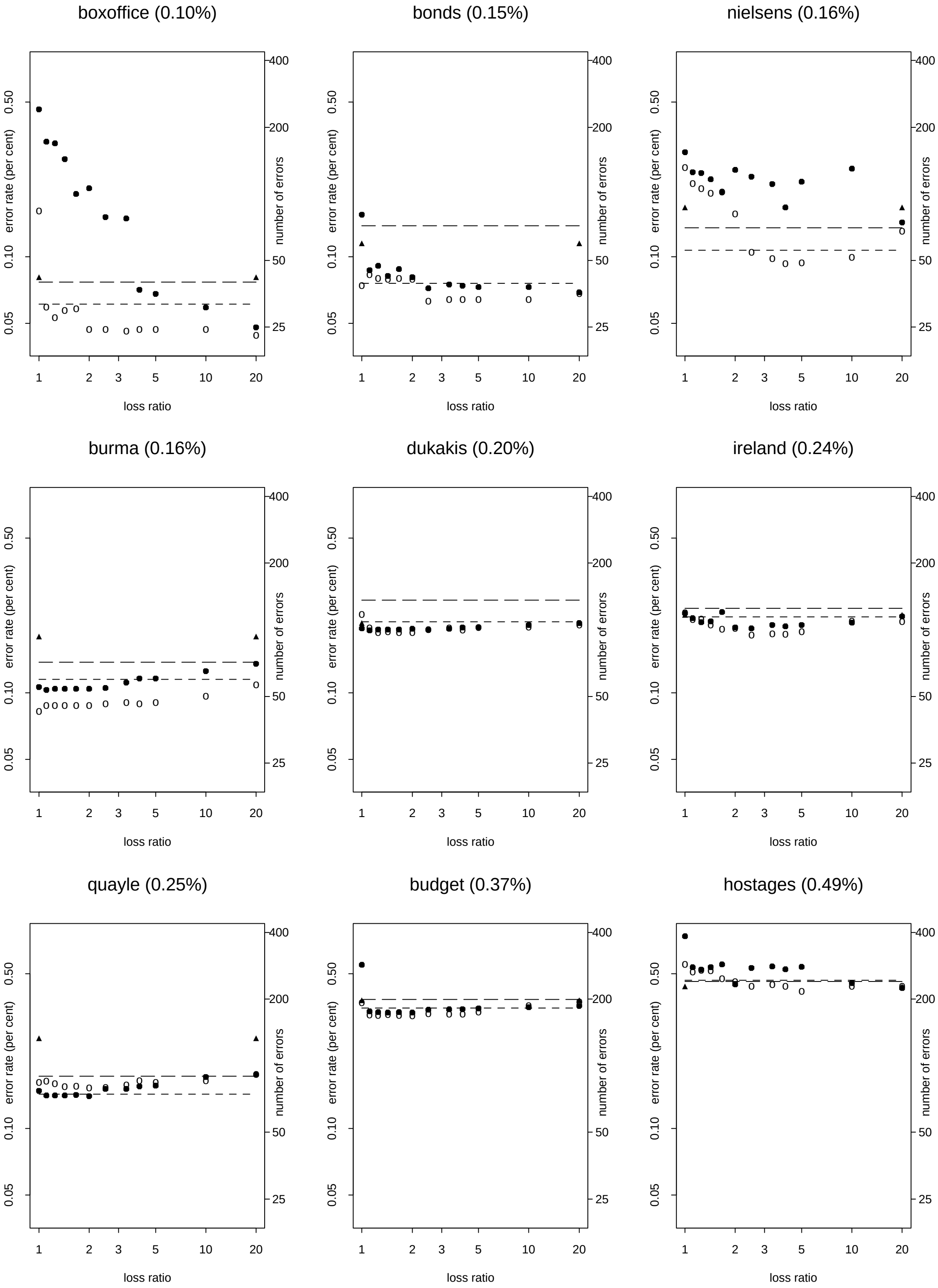

Figure 2: Average error rate for C4.5 rules trained on uncertainty samples of size 299 (black dots) and 999 (white dots), at various loss ratio values. The average error rates for C4.5 rules trained with random samples of size 1,000 (large dashes) and 10,000 (small dashes) are shown as dashed lines. The percentage of positive instances on the training set follows the category name; triangles indicate the percentage on the test set.

		3 + 996 uncertainty				3 + 9997 random			
	Reject	C4.5 (LR=5)		prob. (LR=1)		C4.5 (LR=1)		prob. (LR=1)	
Category	All	Average	SD	Average	SD	Average	SD	Average	SD
tickertalk	0.077	0.077	(0.000)	0.078	(0.001)	0.078	(0.003)	0.109	(0.044)
boxoffice	0.081	0.047	(0.002)	0.048	(0.008)	0.061	(0.018)	0.077	(0.021)
bonds	0.115	0.064	(0.002)	0.069	(0.006)	0.076	(0.020)	0.145	(0.069)
nielsens	0.167	0.094	(0.011)	0.062	(0.005)	0.107	(0.006)	0.100	(0.026)
burma	0.179	0.090	(0.008)	0.098	(0.006)	0.115	(0.040)	0.193	(0.046)
dukakis	0.206	0.197	(0.014)	0.208	(0.020)	0.210	(0.039)	0.235	(0.036)
ireland	0.225	0.188	(0.005)	0.189	(0.011)	0.220	(0.024)	0.228	(0.016)
quayle	0.256	0.161	(0.009)	0.222	(0.012)	0.143	(0.010)	0.263	(0.035)
budget	0.379	0.336	(0.010)	0.361	(0.009)	0.350	(0.014)	0.392	(0.016)
hostages	0.439	0.415	(0.024)	0.360	(0.016)	0.466	(0.039)	0.431	(0.018)

Table 2: Average and standard deviation of percentage error of various classifiers. *Reject all* is a classifier that deems all instances non-members of the category. Two types of training set were used: an uncertainty sample of size 999 and a random sample of size 10,000. Two types of classifier are built from each training set: a decision rule classifier trained using C4.5, and the probabilistic classifier described in the text. When C4.5 was used on the uncertainty sample, a loss ratio of 5 was used; for the random sample a loss ratio of 1 was used (original C4.5). Figures are averages over 20 runs for classifiers built from random samples using the probabilistic method, and over 10 runs for the other three combinations.

			3 + 996 uncertainty				3 + 9997 random			
	Reject All		C4.5 (LR=5)		prob. (LR=1)		C4.5 (LR=1)		prob. (LR=1)	
Category	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN
tickertalk	0.0	40.0	0.0	40.0	1.3	39.3	0.8	39.7	18.3	38.5
boxoffice	0.0	42.0	5.5	19.0	12.6	12.6	5.0	26.8	10.8	29.3
bonds	0.0	60.0	3.6	29.8	7.9	28.3	4.7	34.9	33.6	41.9
nielsens	0.0	87.0	6.0	42.8	9.9	22.2	11.5	44.0	10.6	41.4
burma	0.0	93.0	3.0	43.9	6.0	44.8	5.0	54.6	14.1	86.6
dukakis	0.0	107.0	14.4	88.0	9.5	98.5	68.8	40.1	21.0	101.1
ireland	0.0	117.0	4.8	93.1	16.2	81.9	12.4	101.8	13.8	104.7
quayle	0.0	133.0	23.3	60.2	19.0	96.6	42.3	32.1	17.2	119.4
budget	0.0	197.0	10.6	164.2	29.0	158.5	57.1	124.7	25.7	177.9
hostages	0.0	228.0	30.1	185.6	44.7	142.6	78.3	164.3	25.3	199.0

Table 3: Average number of false positives (FP) and false negatives (FN) for each of 10 categories and 5 conditions. Experiment conditions are the same as for Table 2.

Table 2 lists error rates for both C4.5 and the probabilitistic classifier used during uncertainty sampling. C4.5 figures are for a loss ratio of 5 for uncertainty samples and 1 (the unmodified C4.5) for random samples. The probabilistic classifier uses a loss ratio of 1.0 in both cases. Table 3 shows how the errors divide into false positives and false negatives.

8 Discussion

As Figure 2 shows, an uncertainty sample of 999 instances was in most situations as good for training C4.5 rules on a random sample of 1,000 or even 10,000 instances. At a loss ratio of 5, it was even significantly better (p=.03) than a random sample of 10,000 instances.[3] In some cases, an uncertainty sample of 299 instances is also as good, though this was less reliable. As expected, it is often necessary to use a loss ratio greater than 1 in training rules. Fortunately, there is some leeway in choosing the loss ratio—good error rates are produced for values from 3 to 20 (the highest value we tried) for our data. These results show that heterogeneous uncertainty sampling can indeed be effective. Table 2 presents the data for the larger uncertainty samples and random samples in tabular form.

To point out the extremely low category frequencies, Figure 2 and Table 2 also indicate the error rate of a strategy that classifies all instances as nonmembers. While such a strategy has a low error rate, it is not useful. In most cases the classifiers did manage to beat this error rate, and an evaluation measure that penalized false negatives would show an even greater advantage for the trained classifiers.

Table 2 also shows error rates for the probabilistic classifier, both on the samples it selected and on random samples of size 10,000. C4.5 is significantly better (p=.01) than the probabilistic classifier on the random sample, but only insignificantly better (p=.30) on the uncertainty sample. This suggests that C4.5 is actually more suitable for this text categorization task than the probabilistic classifier, and that there is some penalty in accuracy for heterogeneity in uncertainty sampling.

Table 3 is similar to Table 2 but shows false positives and false negatives separately. This shows that while the total numbers of errors produced by our classifiers were sometimes not substantially smaller than the total number for a strategy that rejects all instances, the errors were more balanced between false positives and false negatives.

9 Future Work

In this section we discuss a few unexplored directions in what we believe is a rich area for study.

We believe uncertainty sampling and other sequential, active, or exploratory approaches to learning [12, 25] enable both learning research and learning applications on large, complex, real-world data sets where fixed training sets are impracticable. Natural language processing, where there is great interest in inducing knowledge to support tagging, parsing, semantic interpretation, and other forms of analysis, is a particularly fruitful application area.

Heterogeneous approaches are likely to become common, in response to both resource limitations and the desire to train new algorithms on previously generated uncertainty samples. A better understanding of how to minimize the problems caused by a heterogeneous approach would be desirable.

Note that we treated our large but finite set of instances as if it were infinite. By adapting results from sequential sampling [32] it may be possible both to improve uncertainty sampling and to tell when additional iterations are no longer providing any benefit—when all the juice has been squeezed out of a data set.

Finally, in contrast to the assumptions made in most theoretical work on querying, our categories are stochastic rather than deterministic. A classifier may indicate that the probability of category membership is 0.5 not because the classifier is incompletely trained, but because the expert may really classify such instances as category members 50% of the time. Indeed, we have seen some evidence of such instances being selected in the later iterations of an uncertainty sampling run.

These murky instances are not the best ones for training [17, 20]. This suggests a goal of producing a classifier that estimates $P(C|\mathbf{w})$ accurately rather than simply classifying accurately. The variance of this estimate becomes important, and it may be more appropriate to treat the problem as one of regression or interpolation [21, 25] rather than classification.

10 Summary

Using partially formed classifiers to select training data incrementally can reduce the number of instances the expert must label to achieve a given error rate. Our experiments show that some reduction is possible even if this uncertainty sampling is heterogeneous: the classifiers used to select instances were of a very different type from the one built from the final sample. The decision rules C4.5 produced from uncertainty samples of roughly 1,000 instances chosen by a probabilistic classifier were significantly more accurate than those from random samples ten times larger. The ability to use cheap classifiers to select data for training expensive classifiers makes uncertainty sampling even more attractive for a variety of applications where large amounts of unlabeled data are available.

[3] Significance by t-score. The null hypothesis was that differences in average error rate across the 10 runs for each category were normally distributed with mean zero and a category-specific variance.

Acknowledgements

We thank William Cohen, Eileen Fitzpatrick, Yoav Freund, William Gale, Trevor Hastie, Doug McIlroy, Robert Schapire, and Sebastian Seung for advice and useful comments on this work, and Ken Church for help with his text processing tools.

References

[1] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[2] I. Bratko, I. Mozetic, and N. Lavrac. *KARDIO: a study in deep and qualitative knowledge for expert systems*. MIT Press, Cambridge, Massachusetts, 1989.

[3] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[4] J. Catlett. Megainduction: a test flight. In *Machine Learning: Proceedings of the Eigth International Workshop*, pages 596–599, San Mateo, CA, 1991. Morgan Kaufmann.

[5] William G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, 3rd edition, 1977.

[6] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with self-directed learning, 1992. To appear in *Machine Learning*.

[7] Stuart L. Crawford, Robert M. Fung, Lee A. Appelbaum, and Richard M. Tong. Classification trees for information retrieval. In *Eighth International Workshop on Machine Learning*, pages 245–249, 1991.

[8] Daniel T. Davis and Jenq-Neng Hwang. Attentional focus training by boundary region data selection. In *International Joint Conference on Neural Networks*, pages I–676 to I–681, Baltimore, MD, June 7–11 1992.

[9] James P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press, New York, 1975.

[10] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Information, prediction, and query by committee. In *Advances in Neural Information Processing Systems 5*, San Mateo, CA, 1992. Morgan Kaufmann.

[11] William A. Gale, Kenneth W. Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439, 1993.

[12] B. K. Ghosh. A brief history of sequential analysis. In B. K. Ghosh and P. K. Sen, editors, *Handbook of Sequential Analysis*, chapter 1, pages 1–19. Marcel Dekker, New York, 1991.

[13] Norm Goldstein, editor. *The Associated Press Stylebook and Libel Manual*. Addison-Wesley, Reading, MA, 1992.

[14] Donna Harman. Ranking algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 363–392. Prentice Hall, Englewood Cliffs, NJ, 1992.

[15] Peter E. Hart. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, IT-14:515–516, May 1968. Reprinted in Agrawala, *Machine Recognition of Patterns*, IEEE Press, New York, 1977.

[16] Jenq-Neng Hwang, Jai J. Choi, Seho Oh, and Robert J. Marks II. Query-based learning applied to partially trained multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2(1):131–136, January 1991.

[17] Igor Kononerko, Ivan Bratko, and Esidija Roskar. Experiments in automatic learning of medical diagnostic rules. Technical report, Jozef Stefan Institute, Ljubljana, Slovenia, 1984.

[18] David D. Lewis and William A. Gale. Training text classifiers by uncertainty sampling. In *Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994. To appear.

[19] David D. Lewis and Philip J. Hayes. Editorial. *ACM Transactions on Information Systems. Special Issue on Text Categorization*, 1994. To appear.

[20] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4:720–736, 1992.

[21] David J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1992.

[22] Michel Manago. Knowledge intensive induction. In *Machine Learning: Proceedings of the Sixth International Workshop*, pages 151–155, 1989.

[23] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall, London, 2nd edition, 1989.

[24] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[25] Mark Plutowski and Halbert White. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, 4(2):305–318, March 1993.

[26] J. R. Quinlan. Discovering rules by induction from large collections of examples. In *Expert systems in the micro-electronic age*, Edinburgh, UK, 1979. Edinburgh University Press.

[27] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[28] J.R. Quinlan. Decision trees as probabilistic classifiers. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 31–37, Irvine, California, 1987.

[29] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1989.

[30] Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. In *Ninth International Workshop on Machine Learning*, pages 385–393, 1992.

[31] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.

[32] Bikas Kumar Sinha. Sequential methods for finite populations. In B. K. Ghosh and P. K. Sen, editors, *Handbook of Sequential Analysis*, chapter 1, pages 1–19. Marcel Dekker, New York, 1991.

[33] Paul E. Utgoff. Improved training via incremental learning. In *Sixth International Workshop on Machine Learning*, pages 362–365, 1989.

[34] Sholom M. Weiss, Robert S. Galen, and Prasad V. Tadepalli. Maximizing the predictive value of production rules. *Artificial Intelligence*, 45(1–2):47–71, September 1990.

[35] P. H. Winston. Learning structural descriptions from examples. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 157–209. McGraw-Hill, New York, 1975.

[36] J. Wirth and J. Catlett. Costs and benefits of windowing in ID3. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, Michigan, 1988. Morgan Kaufmann.