A Redundant Covering Algorithm Applied to Text Classification

David Hsu and Oren Etzioni
Department of Computer Science
and Engineering, Box 352350
University of Washington
Seattle, WA 98195-2350
{hsud,etzioni}@cs.washington.edu

and

Stephen Soderland
Radiology Department
Mail Stop CH-69
Children's Hospital and Medical Center
4800 Sand Point Way NE
Seattle, WA 98105-0371
{soderlan}@cs.washington.edu

Abstract

Covering algorithms for learning rule sets tend toward learning concise rule sets based on the training data. This bias may not be appropriate in the domain of text classification due to the large number of informative features these domains typically contain . We present a basic covering algorithm, DAIRY, that learns unordered rule sets, and present two extensions that encourage the rule learner to milk the training data to varying degrees, by recycling covered training data, and by searching for completely redundant but highly accurate rules. We evaluate these modifications on web page and newsgroup recommendation problems and show recycling can improve classification accuracy by over 10%. Redundant rule learning provides smaller increases in most datasets, but may decrease accuracy in some.

Introduction

The covering algorithm is one of the most well-studied methods for learning sets of rules from examples. Covering algorithms repeatedly iterate through the following loop: generate a rule on the training data; remove the training data covered by the rule. These algorithms have performed well in many supervised learning tasks, and two such algorithms, SWAP-1 and RIPPER, have recently been successfully applied to problems in text categorization (Apté, Damerau, & Weiss 1994) (Cohen & Singer 1996).

This method, however, may not be well-suited to learning text classifiers. In text domains, documents are often represented as feature vectors, where every word in the training set has a corresponding feature. Therefore, the number of features in a dataset may number in the thousands, and can greatly outnumber the number of examples in the training set.

Because covering algorithms remove examples after they have been covered by a generated rule, the large feature-to-example ratio presents special problems. First, as the number of training examples a rule is generated on dwindles, the actual error rate of the generated rule tends to increase. This has been termed the small disjuncts problem (Holte, Acker, & Porter 1989). Pagello and Haussler make a similar observation on the importance of having a signifigant sample size for finding attributes to split a decision tree on (Pagello & Haussler 1990). In text domains, the problem is pronounced because the feature set is already large compared to the number of training examples, and therefore the chance that some noisy feature is prominent in a small sample is large.

In addition, the behavior of removing correctly classified examples from the training set leads covering algorithms to ignore features that don't appear to improve classification accuracy on the training set. A study by Joachims demonstrates that a classifier that only uses words with low information content can still greatly outperform random guessing (Joachims 1997). This indicates that a large number of the words in a document corpus may be relavent to the classifier to be learned, and including more words in a ruleset may improve the predictive accuracy of the ruleset.

We hypothesize that the covering algorithm can generate more accurate rulesets in many text domains by learning more complex rulesets from the available training data. We investigate this claim by creating a simple propositional covering algorithm, DAIRY, that milks the training data in order to produce more complex hypotheses. DAIRY milks the training data in two ways. First, in contrast to previous covering algorithms, DAIRY does not remove training examples once they have been covered. This extension allows DAIRY to more fully utilize the training data available since a training example can always affect the growth of any rule. Second, DAIRY engages in a search for completely redundant, but highly accurate, rules after it cannot learn rules that cover uncovered examples anymore. These two extensions lead DAIRY to generate more complex rule sets since DAIRY is no longer only concerned with learning only enough rules to classify the training examples. We test our extensions on several problems in text classification and find that DAIRY's ability to learn more from the dataset can signifigantly improve the accuracy of the rulesets it generates.

Figure 1: Standard Covering Algorithm

```
Theory ← ∅
while # of Positive Examples ≠ 0
    Rule ← FindBestRule(Examples)
    Covered ← Examples covered by Rule
    if Rule is not accepted
        break
    Examples = Examples - Covered
    Theory = Theory ∪ Rule
return (Theory)
```

A Motivating Example

This work is motivated by the authors' experience in applying a rule learning algorithm to classify web pages in a web history privacy manager (Lau 1997). This application allows a community of users to query other people's web histories. The problem was to learn a set of rules that predicts whether the user would like a keep a web page private in this shared environment. The rules were generated from classified pages in the user's browsing history where each page is represented by set valued features (Cohen 1996b) representing the URL, title, and document body.

In this domain, we noticed an interesting difference between the error rates induced rule sets would exhibit on test sets drawn from their respective user's web histories and error rates on pages that the user had not visited before. Even though an induced rule set would make predictions with very low error rates on test pages from the user's web history, it would exhibit a much higher error rate on pages the user had not visited before.

A representative example is the problem of generating rules to describe the concept of "computer games" from one user's web history. This user had browsed pages featuring a couple of game titles. However, a large portion of the game pages dealt with one particular game, "Return to Krondor." The resulting rule set was very representative of the user's history. It contained rules for the title, "Krondor," as well as rules which referred to other specific games or websites. However, there were very few rules which actually exploited common words among these games. Rules specific to a particular game, such as the rule mentioning "Krondor," while highly precise, do not make good predictors for computer game pages in general.

In the next section, we review the standard covering algorithm and show why it may learn the rule set described in the previous example.

Covering Algorithm and Improvements

Figure 1 describes a covering algorithm that learns rules in a two class problem. The algorithm repeatedly generates a single rule, computes the examples that are covered by the rule, and removes the covered examples from the training set before the next rule is generated. The latter step, removing covered examples, is the step that characterizes a covering algorithm. This step causes the algorithm learns just enough rules to classify the training data, because once the algorithm learns a rule that classifies an example, it removes the example from further consideration.

On the set of game pages described in the previous section, because the pages about "Return to Krondor" represent a large portion of the dataset, the covering algorithm learned a rule that mentioned the word "Krondor." Because of the removal of these pages, the covering algorithm cannot intentionally generate rules that exploit common words in both "Krondor" pages and other game pages. Therefore, the covering algorithm is more likely to learn rules specific to one particular game or website. In addition, the game pages about "Krondor," may contain many words that are strongly correlated with game pages, such as "dragon" or "puzzles." A classifier that learns rules that contain these words may provide a more accurate ruleset than a classifier that doesn't.

The key point is that the information to create accurate rule sets exists in the game pages from the user's web history. However, since the covering algorithm removes training examples once they are covered by a rule, it does take advantage of this information. In the introduction, we proposed two general ideas through which a covering style algorithm may take fuller advantage of the information in the dataset: recycling covered training examples, and learning redundant rules. We now describe these two ideas more fully, and how a covering algorithm that implements both ideas may learn a more predictive ruleset from the game page domain.

A rule learner that recycles covered training examples, as opposed to removing these examples, can still learn from examples that have been previously covered. With a recycling approach, covered training data remains in the pool of training data that the learner uses, but at a discounted weight. Therefore, recycled examples do not affect rule learning behavior as much as uncovered examples. Most importantly however, the information contained in covered examples can still be used by the covering algorithm to generate new rules. Therefore the fact that a certain word appears in many of the "Krondor" pages may improve its chance of being added to a new rule.

Redundant rule learning refers to the search for high quality rules regardless of whether the rules uniquely cover some training instances or not. In the computer game pages domain, the web pages about "Return to Krondor" may have many informative words. A redundant rule learner will ferret these words out even if they only appear in game pages about Krondor.

Figure 2: DAIRY Rule Learning Algorithm

```
Theory $\leftarrow \emptyset$
Default $\leftarrow$ class with most instances
While rule was accepted in the previous iteration
    For each Class that is not the Default
        (Covered, Uncov) $\leftarrow$
            Partition(Theory,Examples)
        Rule $\leftarrow$ FindRule(Class,Covered,Uncov)
        If Rule meets acceptance criterion
            Theory $\leftarrow$ Theory $\cup$ Rule
    (Covered, Uncov)$\leftarrow$Partition(Theory,Examples)
    Covered $\leftarrow$ RetainMisclassified(Covered)
    Rule $\leftarrow$ FindRule(Class,Covered,Uncov)
    If Rule meets acceptance criterion
        Theory $\leftarrow$ Theory $\cup$ Rule
Search for redundant rules
return Theory
```

The DAIRY Rule Learner

In this section, we describe a basic covering algorithm, DAIRY, that we implemented to test our redundant rule learning approach.

DAIRY, shown in Figure 2, is a propositional rule learner that learns an unordered list of rules, with a default classification appended at the end. The skeleton of the algorithm is very similar to the covering algorithm in Figure 1. However, we have adapted it slightly to work on multiclass problems.

Prior to entering the main loop, the algorithm chooses the default class by picking the class with the most examples. The algorithm, then, repeatedly iterates through the main loop. In each iteration, the algorithm attempts to learn a rule for each class that is not the default class. Instead of removing covered examples, the algorithm separates covered examples from uncovered examples before learning a rule for a classification. In DAIRY's recycling approach, the status of whether an example is covered or uncovered does not matter for negative examples, but does matter for positive examples.

DAIRY handles rules for the default class specially. Since the default rule automatically matches examples from the default class, we did not see the need to learn many rules for the default class. The only rules with the default classification that DAIRY learns are rules that differentiate the members of the default class that have been previously misclassified. Therefore, the uncovered positive examples for the default class are examples that have been misclassified by a non-default rule, but not yet classified by a rule predicting the default class. In domains where the majority class represents a large portion of the training set, this special treatment can prevent rules of the majority class from overwhelming the rules of the minority class.

In order to find the best rule to add to the ruleset, DAIRY performs a greedy search which repeatedly adds words to an initially empty rule. Possible word expansions are evaluated via a scoring function. DAIRY currently uses the correlation heuristic as its scoring metric(Fürnkranz 1994). DAIRY ends rule growth once no new conjunct improves the m-level accuracy of the rule. M-level accuracy is a generalization of laplace accuracy in which the M value controls the influence of an a-priori probability on the accuracy measure of a rule.

$$accuracy = \frac{p + M * \frac{P}{P+N}}{M + p + n} \qquad (1)$$

P and N refer to the total number of positive and negative documents in the dataset. The p and n values refer to the positive and negative numbers covered by the rule.

The generated ruleset classifies unlabelled examples via a voting metric. Each rule contribute's a vote with the value of the rule's m-level accuracy for the class predicted by the rule. The rule set assigns the unlabelled example the classification with the highest vote total. If no rule matches an example, the example is assigned the default class. In addition to an unordered rule list representation, we have experimented with decision list hypotheses, but classification accuracy consistently suffers from these representations. Also the benefits of recycling and redundant rules are not apparent with decision list representations. This makes intuitive sense because, in a decision list, only one rule can contribute to the classification of a single example. Even though our algorithm may learn several rules, many of the rules may not have a chance of contributing to the classification of examples.

We now describe the implementation of recycling and redundant rule learning within the DAIRY algorithm.

Recycling Covered Training Examples

DAIRY recycles covered training examples in an effort to learn rules that cover more examples. Basically, DAIRY assigns a discounted weight, the recycling factor, to covered positive examples when growing new rules. This discount factor provides a tradeoff between learning rules that cover new examples, and rules that have a high coverage.

The recycling factor directly affects the scoring function used to evaluate candidate words to add to a rule. These scoring functions rely on some measure of the counts of positive and negative examples that contain the word, and positive and negative examples that do not contain the word. Negative examples always have the same weight in the example counts regardless of whether they have been previously covered or not. The recycling factor comes into play in the count of positive documents. Covered positive documents have a relative strength given by the recycling factor in comparison to uncovered documents.

For example, suppose the scoring function is Laplace accuracy, shown in the following equation.

$$\frac{1 + pos}{\#classes + pos + neg} \tag{2}$$

DAIRY transforms Equation 2 into a recycling formula by splitting the positive examples into previously covered positives and previously uncovered positives and multiplying the previously covered ones by the recycling factor. The result is Equation 3.

$$\frac{1 + recyclefactor * covpos + uncovpos}{\#classes + recyclefactor * covpos + uncovpos + neg} \tag{3}$$

Even if previously covered examples have a discounted weight, a candidate word may cover so many previously covered positives that it is chosen even though it covers no new positives. Therefore, words that do not cover any new positive documents are not considered in the rule expansion phase until the redundant rule learning phase.

Another straightforward approach to recycling would be to multiply an example's weight by the recycling factor each time a new rule covers it. Consider the case, however, where an article contains many informative words. Discounting the weight of the article in this way may decrease the chance that rules containing these new words are learned. However, further study needs to be done to analyze the effectiveness of different recycling policies.

Learning Redundant Rules

As a post processing step, DAIRY conducts a large scale search for redundant rules, rules that do not cover any uncovered examples, but are still of high quality as defined by some accuracy, coverage, and depth bounds. The post procesing step resembles the preceding portion of the DAIRY algorithm. However, for each non-default class, DAIRY tries to approximate a massive search for all good rules as opposed to a search for the single best rule.

The beam search that DAIRY conducts is similar to the beam search procedure used in CN2 with a few minor adjustments (Clark & Niblett 1989). Candidate rules are evaluated via the m-level accuracy measure and rules that specialize or are equivalent to existing rules are not generated. The beam width, maximum depth, minimum accuracy requirements, and minimum coverage requirements for candidate rules are all user defined parameters.

DAIRY uses a beam search because a purely greedy search for the single best rule may result in a rule that does not meet the predetermined accuracy or coverage criterion. Therefore the beam search is necessary to prevent the redundant rule phase from ending too early. This beam search continues for each class until DAIRY can no longer find redundant rules that meet the depth, accuracy, and coverage criteria.

For the default class, DAIRY follows the same procedure as in the main DAIRY body, only learning rules that cover at least one uncovered positive instance. An uncovered positive instance corresponds to instances that have been misclassified by a generated rule, but are not yet covered under a rule that predicts the default class.

Instead of learning redundant rules in a post processing phase, we could have integrated redundant rule learning in DAIRY's normal rule learning phase. However, there is a potential for the redundant rules to starve out rules that cover uncovered examples. This can happen when redundant rules all qualify with a higher m-level accuracy than rules covering new examples. If none of the rules meets the acceptance criterion for redundant rules, then the algorithm terminates without learning rules that cover positive instances.

Table 1: Datasets used in the tests

Name	#classes	#doc's
newsgroups	20	20000
e-mail	2	750
game web pages	4	4000
movie/music/cs/games/tv	5	8000

Extensions for Text

In regard to text domains, DAIRY attempts to learn keyword spotting rules, rules that test whether certain words appear in the document or not. Since feature vectors make very inefficient use of space, DAIRY uses set valued features (Cohen 1996b), and in the experiments below, represents each document as a single set valued feature. Therefore each rule is a conjunction of several tests, each test being whether the word is in the document set or not.

Experimental Evaluation

In the following section, we report results on experiments that test the effect of recycling and redundant rule learning on the accuracy of rulesets that DAIRY learns in datasets drawn from domains on the web, e-mail, and newsgroups.

Methodology

For our experiments, we draw some text datasets from a variety of real world domains. We use a 20 newsgroups datasets drawn from the CMU text learning archive. The talks dataset attempts to identify talk announcement e-mail messages from other messages. This dataset had been previously used in (Cohen 1996a). Finally, we construct two datasets from categories or subcategories in the Yahoo hierarchy.

Figure 3: With recycling, DAIRY performs better with 360 examples than with over 5000 examples in the Music/Movie/TV/CS/Games domain.

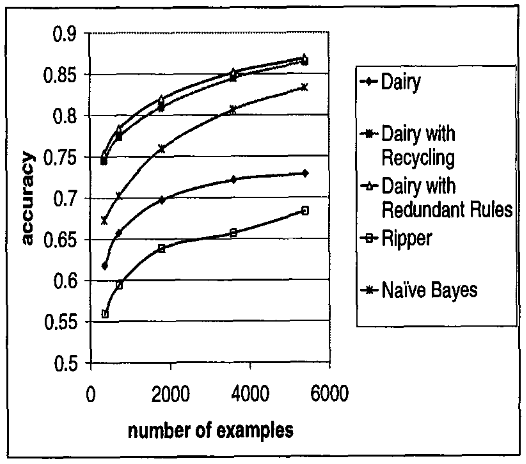
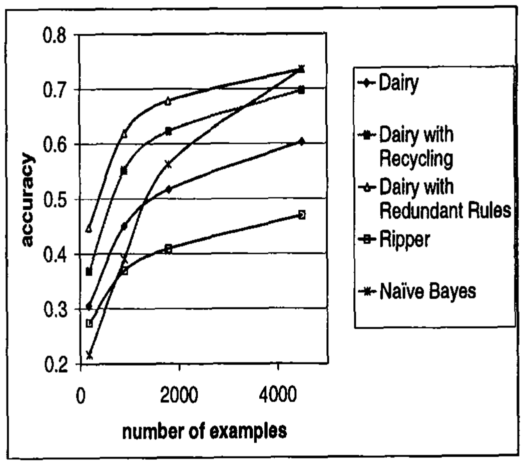


Figure 4: Recycling and redundant rule learning improve DAIRY's accuracy by over 10% on the 20 newsgroups dataset.



One dataset contains four subcategories of the computer games category, each subcategory containing a genre of computer games. The second dataset is a top level dataset that contains music, movie, TV, game, and computer science web pages. The datasets are summarized in table 1.

We also gather results on two well-known algorithms that have been applied to text categorization. The naive Bayesian classifier learns the posterior probabilities for class membership given a set of words. Although naive Bayes relies on an unrealistic assumption that words occurrences are independent of one another, it performs well in several text domains. We use a variant of the naive Bayesian classifier that is described in (Mitchell 1997). In our experiments we did not perform feature selection since we have found that it reduced performance in at least one domain, the newsgroups domain.

The second classifier used was RIPPER, a propositional rule learner that uses a combination of pruning techniques, heuristic evaluations, and an optimization phase in order to generate a compact decision list rule set(Cohen 1995).

In our tests, we employ a 10-fold incremental cross validation scheme. We first split the dataset into 10 folds, and use 9 for training and the 10th for testing. Instead of training on the entire training set, we train on a succession of subsets of each training group, each subset containing a larger portion of the training set. Each subset is created by randomly selecting training examples with a certain percentage. We run each learner on the same subsets of training data, and average the results for each training set size.

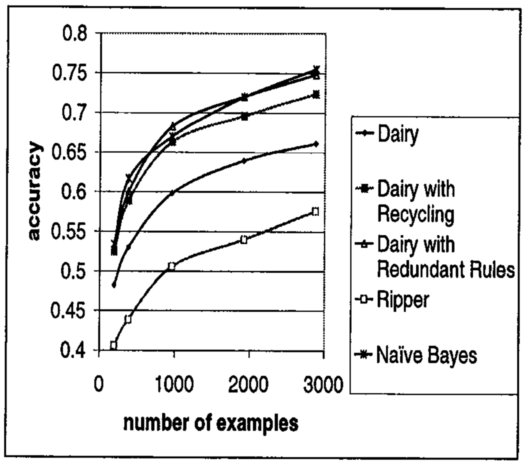
Figure 3, Figure 4, Figure 5, and Figure 6 show a sample of the results that we have gathered on the four datasets. We illustrate three variations of DAIRY. The first variant does not recycle and does not learn redundant rules. The second variant uses a 75% recycling factor when weighing covered examples. The third variant uses the 75% recycling factor and learns redundant rules of at most length 2 and with a minimum accuracy threshhold of 60%. In the newsgroup domain, the minimum coverage for redundant rules is set to 3, and in all other domains, it is set to 5. Since the number of articles per newsgroup is especially small on small amounts of data, the smaller coverage threshhold for newsgroup facilitated the learning of more redundant rules. We set the maximum rule depth to 2 since almost all generated rules that we have observed in the non-redundant learning phase took 3 or fewer conjuncts.

Results and Discussion

We immediately see that recycling consistently improves the accuracy of the resulting rule set. This improvement holds across domains and across variations in training set size. In fact, in each individual run in the cross validation, using a recycling factor greater than 0 outperforms running DAIRY without recycling on each dataset except for the talks dataset. Experiments in which we vary the scoring function, stopping criterion, and recycling level all show an increase in accuracy due to recycling in the domains tested.

DAIRY itself is competitive turning in consistently

Figure 5: On the computer game genres domain, recycling and redundant rule learning noticeably improve DAIRY's accuracy.



more accurate rulesets than RIPPER, but not performing as well as naive Bayes on these domains. The poor performance of RIPPER seems surprising especially in light of its performance on standard text collections such as the reuters dataset. The difference in performance lies in the type of classifier RIPPER learned. In measurements on the reuters datasets, RIPPER learned a separate classifier for each class, while in these tests, RIPPER learned one classifier for the entire dataset.

The effect of including redundant rules in the training data also leads to general increases in the accuracy of the resulting rule set. Again, this increase seems to hold across several dataset sizes. However, the benefits are much more muted. The benefit of finding redundant rules containing informative words may be tempered by the amount of overfitting that redundant rule learning invariably causes. Still, redundant rule learning does produces useful improvements in accuracy in two of the domains.

One way to measure the effectiveness of recycling and redundant rule learning is to measure how many training examples these techinques are worth. That is, how many more training examples does DAIRY need to see to match the performance of recycling and redundant rule learning? In the 20 newsgroups domain, DAIRY with a 75% recycling factor, and learning redundant rules with a minimum 60% accuracy achieves about 60% accuracy around 900 examples. In contrast, naive bayes achieves the same accuracy after about 1000 more examples, and DAIRY with a 0% recycling factor achieves that accuracy at around 5000 examples. In the Music/Tv/Movie/Games/CS domain, the benefit of recycling and redundancy is even more pronounced with DAIRY providing better performance with recycling and redundancy on 360 examples, then DAIRY without recycling on over 5000 examples.

Finally, we turn our attention to the potential for overfitting. Since recycling and redundant rule learning lead DAIRY to generate more complex hypotheses, overfitting becomes a very real possibility.

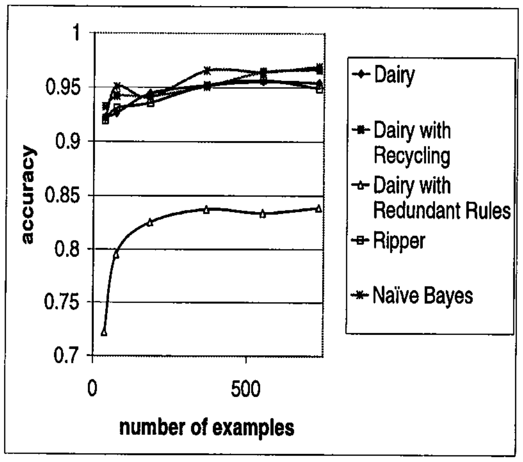
The results on the talks dataset may shed some light on how much recycling and redundant rule learning contribute to this phenomenon. The talks dataset is an example of a domain where the underlying concept is quite simple and therefore the recycling and redundant rule learning approaches employed in DAIRY may overfit the training data. RIPPER, for example can learn an 8 rule hypothesis that classifies the test data with 96% accuracy.

The results from the talks dataset show that recycling did not lead to a decrease in ruleset accuracy, but redundant rule learning did. We hypothesize that since DAIRY learns redundant rules with an accuracy threshhold lower than the DAIRY can provide already, this hurts performance considerably. In addition, the number of low accuracy rules for the "talk-announcement" class far outweighed the number for the default class, so the voting metric is skewed towards the minority class. On tests, where we set the accuracy threshhold to 100%, DAIRY learns far fewer redundant rules and achieves accuracies around 95%, which still indicates a slight amount of overfitting taking place.

Examination of rulesets that were generated with redundant rule learning, and recycling to a lesser extent, do show signs of overfitting. For example, the rule learner may learn several rules which only differ in the last conjunct and that last conjunct may not be an important word. In fact, at least in the web domains, the improvement given by redundant rule learning increases when redundant rules are restricted to length 1. Also, with the parameter settings reported in this section, many of the rules returned in the redundant learning phase seem noisy. More careful evaluation of the parameter settings need to be done to gauge how effective redundant rule learning is.

In this light, the relative lack of decline in predictive accuracy from overfitting on these datasets is somewhat surprising. One reason for this phenomenon may be the interpretation of hypotheses as a list of voting rules. Rules that have small coverage are more likely to overfit the training data. With the m-level accuracy formula, these rules will naturally have smaller weight than rules that cover more examples. Therefore, larger coverage rules will tend to overshadow low coverage rules in the vote. If only low coverage rules match a training instance, that instance may not have a very clear classification. As we mentioned before, decision lists learned with recycling and redundant rule learn-

Figure 6: In recognizing talk announcement e-mails, recycling does not degrade DAIRY's performance, but redundant rules do.



ing perform worse than decision lists learned without these techniques.

Related Work

A few rule learners have already been applied to the field of text classification. Two of them, SWAP-1, and RIPPER fall under the category of covering algorithms. The CHARADE algorithm performs a breadth first search to find good rules (Moulinier, Raskinis, & Ganascia 1996). Like DAIRY, CHARADE naturally incorporates redundancy and will continue to search for rules until no good rule can be found or each example is covered by some user-defined number of rules. However, CHARADE's search strategy leads to the need for automatic feature selection before running the algorithm, whereas DAIRY can run on the full text dataset.

Other approaches to rule learning may make more use of the training data than covering algorithms. For example, RISE combines the strength of instance based learning with rule learning and generates rules in a bottom-up strategy (Domingos 1996). Other rule learners such as Brute perform a massive search to generate a good set of rules (Riddle, Segal, & Etzioni 1994). Since these techniques are polynomial in the number of attributes, they are ill-suited in text domains without some form of feature selection.

Boosting is another method by which a learner can milk the training data. The Adaboost algorithm generates a set of classifiers by repeatedly training a base learning algorithm on different distributions of examples from a training set. Examples are reweighted after each classifier is learned so that harder to classify examples become more prominent in the training set (Freund & Schapire 1996). This philosophy is different from recycling in that easy to classify examples will soon have little impact in the training set without regard to the number of informative features the example contains.

Conclusion

This paper demonstrates that in text classification domains, allowing a covering algorithm to milk the training data may lead to improved accuracy in the final ruleset. In DAIRY, the accuracy of the covering algorithm can be greatly improved by allowing it to recycle previously covered training data. Redundant rule learning may also be beneficial but the jury is still out. Although these techniques were developed in a simple covering algorithm, we expect them to transfer well to more complex covering methods that can learn unordered rulesets. In turn, covering algorithms that can create more predictive rulesets from smaller amounts of data may be more attractive for use in information filtering tasks or active learning applications of text classification where the ability to generalize well from small amounts of data is important (Lewis & Catlett 1994) (Pazzani & Billus 1997).

Acknowledgements

We wish to thank Corin Anderson, Anhai Doan, Erik Selberg, Jared Saia, and the reviewers from the workshop for helpful comments. This research is funded in part by Office of Naval Research grant 92-J-1946, by ARPA / Rome Labs grant F30602-95-1-0024, by a gift from Rockwell International Palo Alto Research, and by National Science Foundation grant IRI-9357772.

References

Apté, C.; Damerau, F.; and Weiss, S. M. 1994. Automated learning of decision rules for text categorization. *ACM Transactions on Office Information Systems* 12(3).

Clark, P., and Niblett, T. 1989. The CN2 induction algorithm. *Machine Learning* 3(4).

Cohen, W., and Singer, Y. 1996. Context sensitive learning methods for text categorization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Cohen, W. 1995. Fast effective rule induction. In *Proceedings of the Twelth International Conference on Machine Learning.*

Cohen, W. 1996a. Learning rules to classify e-mail. In *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning for Information Access.*

Cohen, W. 1996b. Learning trees and rules with set valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*

Domingos, P. 1996. Unifying instance-based and rule-based induction. *Machine Learning* 24.

Freund, Y., and Schapire, R. E. 1996. Experiments with a new boosting algorithm. In *Proceedings of the Eleventh International Conference on Machine Learning.*

Fürnkranz, J. 1994. Fossil: a robust relational learner. In *Proceedings of the Seventh European Conference on Machine Learning (ECML-94).*

Holte, R. C.; Acker, L. E.; and Porter, B. W. 1989. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89).*

Joachims, T. 1997. Text categorization with support vector machines: learning with many relavent features. Technical Report LS8-Report, University of Dortmund.

Lau, T. 1997. Privacy in a collaborative web browsing environment. Master's thesis, University of Washington.

Lewis, D., and Catlett, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceeding of the Eleventh International Conference on Machine Learning.*

Mitchell, T. 1997. *Machine Learning.* McGraw Hill.

Moulinier, I.; Raskinis, G.; and Ganascia, J.-G. 1996. Text categorization: a symbolic approach. In *Symposium on Document Analysis and Information Retrieval.*

Pagello, G., and Haussler, D. 1990. Boolean feature discovery in empirical learning. *Machine Learning* 5.

Pazzani, M., and Billus, D. 1997. Learning and revising user profiles: the identification of interesting web sites. *Machine Learning* 27.

Riddle, P.; Segal, R.; and Etzioni, O. 1994. Representation design and brute-force induction in a boeing manufacturing domain. *Applied Artificial Intelligence* 8.