ML Techniques for Labeling Legal Opinions

COMPSCI 281B Project Report

Christopher Hundt

May 6, 2008

1 Introduction

In this paper I discuss the application of machine learning techniques to the problem of labeling legal opinions with topic information, an important application domain that has received relatively little attention from the machine-learning community. I attempt to use both textual information from the opinion and structured information from the opinion's citations.

After laying out the problem in section 2, I apply standard machine-learning techniques to the textual information in a supervised fashion in section 3. Section 3.5 discusses experimental results from these techniques, and section 4 describes attempts to improve those results by boosting. Finally, section 5 gives an approach to using stochastic blockmodeling to make the citation information helpful in learning, and gives experimental results for the technique.

2 Labeling Legal Opinions

Topical classification of legal opinions is an important area of text classification. Legal opinions are written accounts of the reasoning behind the opinions of judges in making a legal decision. Judges frequently refer to other judges' opinions as guidance in resolving issues and, furthermore, in common-law jurisdictions like the United States, the decision of a higher court constitutes binding precedent that lower courts in its jurisdiction are expected to follow.

For this reason, legal opinions are collected into large collections for reference by those parties interested in finding the "case law" related to a given issue. The great quantity of legal opinions written means that the collections must have some structure. One type of structure is classification into various topics.

Legal opinions are in some ways different from other types of texts. As discussed in [Tho01], legal opinions may discuss many issues, some of which could not reasonably be called a focus of the opinion. Furthermore, legal opinions contain many specialized "terms of art" which tend not to appear in other types of texts. This, however, makes the task a little easier, as it lends a certain uniformity of phrasing to the opinions.

2.1 Features: Bag of Words

As the main concern of this project is to test various machine learning techniques, I did not dedicate a great amount of time to creating a textual feature structure for the opinions. I chose a simple "bag-of-words" representation, in which each opinion is represented as a vector consisting of all the unique words and bigrams appearing in the opinion and the number of times each appears. To reduce overfitting, I ignored all words that do not appear in at least four opinions. To aid in generalization I converted all words to their "stems" using a variation of Porter's stemming algorithm [Por97]. Furthermore, for each individual tagging task I ranked them all according to their "information gain": the entropy of the tag label minus the conditional entropy of the tag label given the feature value. All features ranked below 150 by this heuristic were discarded, as they were generally uninformative and tended to do more harm (through overfitting) than good when included.

2.2 Citation Structure

Another favorable feature of legal opinions is uniformity of citation format. The majority of jurisdictions follow the style laid out in "The Bluebook: A Uniform System of Citation," and those who do not often follow some close variant. Furthermore, most case law is published in periodic "reporters," and the location in the relevant reporter is cited when an opinion wishes to cite a previously written opinion. For example, United States Supreme Court decisions are published in the U.S. Reporter, which is cited as "x U.S. y," where x is the volume of the reporter containing the case of interest, and y is the number of the page where it starts.

This extremely simple format and great uniformity makes it quite easy to tell what other authorities are cited by a given opinion. This structure can then be used as additional information about the opinions: the authorities cited may help indicate what the opinions are about. I investigate this possibility in section 5.

3 Supervised Learning

The first task I undertook was to attempt to learn categorization by supervised learning. Since an opinion may have more than one label, my approach was to treat each label as creating a separate binary classification problem. Although the labels are not independent, I did not take advantage of any dependence that they might have; I trained an independent binary classifier for each tag.

In the next few sections I describe a few different classifiers that I evaluated, and then in section 3.5 I show experimental results for each type. I show results only for a few different tags, because those are the tags with a significant number of positive examples.

3.1 Naïve Bayes

In this simple model, the class is treated as a latent random variable and the features (word and bigram counts) are assumed to have been generated as multinomial random variables, each dependent only on the class (and thus independent from each other given the class). The simplifying assumptions of this model are obviously false in real text documents, but it nonetheless can be useful for text classification. (See, for example, [MN98].)

3.2 Decision Stumps

A decision-stump learner must pick a single word or bigram and choose a threshold on that term's count as its decision boundary. Although this may seem overly simple (the entire document is reduced to a single discrete variable for classification purposes) it frequently meets with considerable success, as I will show, and is very amenable to boosting. (See section 4.) It is also resistant to overfitting: the decision stump's VC dimension grows only logarithmically in the number of features.

3.3 Decision Trees

Decision trees may make branching decisions, each of which is a threshold on a given word or bigram count. It has considerably more classification power than the decision stump (in fact, potentially unlimited classification power), and attempts to limit overfitting by restricting the tree generation in some way. I generate a C4.5 tree by the J48 algorithm. (See [Qui93] for more information.) Its principal ways of preventing overfitting are by requiring a minimum number of training examples that fall into each leaf and by pruning, the latter of which is controlled by a "confidence factor."

3.4 Support Vector Machines

Support Vector Machine algorithms create a linear decision boundary in the feature space. I used the Sequential Minimal Optimization implementation of the SVM [Pla98] with a Gaussian kernel $\kappa(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$. This kernel function creates a new, infinite-dimensional feature space, so a regularization term, scaled by a regularization constant, is used to avoid overfitting.

3.5 Experimental Results

In order to get a better idea of how different algorithms or different parameter choices affect the results, I wanted to create precision-recall curves. Since most of the algorithms did not have parameters that allow the precision or recall to be adjusted directly, I used MetaCost [Dom99], a metalearning algorithm which attempts to simulate cost-sensitive learning by iteratively re-weighting the training set and re-learning. This did not create smooth precision-recall curves, but allowed me to push the algorithms in one direction or the other, giving something a bit more informative than a single precision/recall pair.

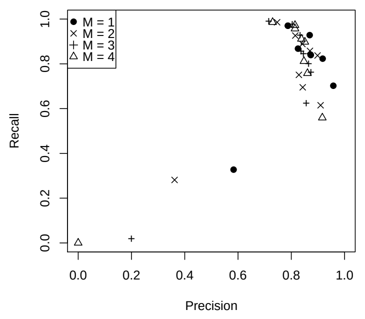
I first turn my attention to tuning the parameters for the algorithms that require it, and then to comparative results.

3.5.1 Decision Tree Parameters

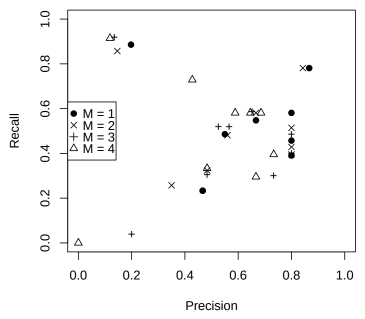
The J48 algorithm asks for two important parameters: M, the minimum number of examples that must be at each leaf, and C, a "confidence factor" used in pruning. C turned out not to have a great effect on classification results, but M had a significant effect. In figure 1 you can see that $M = 1$ seems to be the best choice, indicating that in this domain the risk of overfitting by allowing leaves with only one training example is minimal or is mitigated by other factors.

3.5.2 SVM Parameters

In training an SVM we must set the regularization coefficient C and the variance of the Gaussian kernel used. A smaller variance will tend to overfit, learning to classify as positive only those inputs which are very close to a



(a) Precision-recall curve for J48 on the Sentencing tag.



(b) Precision-recall curve for J48 on the Discrimination tag.

Figure 1: The effect of M in the J48 tree-learning algorithm.

positive training example. This is reflected in figures 3(a) and 3(b): the recall drops significantly as the variance drops. On the other hand, it can result in higher precision, as shown in 3(a).

This overfitting can be mitigated by regularization. In the above example the regularization coefficient is small, but in figure 2 we see that increasing C brings recall close to what it is for larger variances (cf. figure 3(a)). In the end, it appears that it is necessary to try various choices of the two parameters for each individual tag; no single choice seems to be optimal in this domain.

3.5.3 Comparison of algorithms

Once appropriate parameters have been chosen for the algorithms, we can compare them to each other. Figure 4 shows this comparison for two different tags. While it appears that both J48 and SMO are superior to the simpler Naïve Bayes and Decision Stump algorithms, picking between J48 and SMO is not as easy; which one is preferred depends on the particular tag in question and how one values precision and recall.

4 Boosting

Next I considered the effect of boosting on the learning algorithms. I used an implementation of AdaBoost called AdaBoost.M1 [FS96], an algorithm that

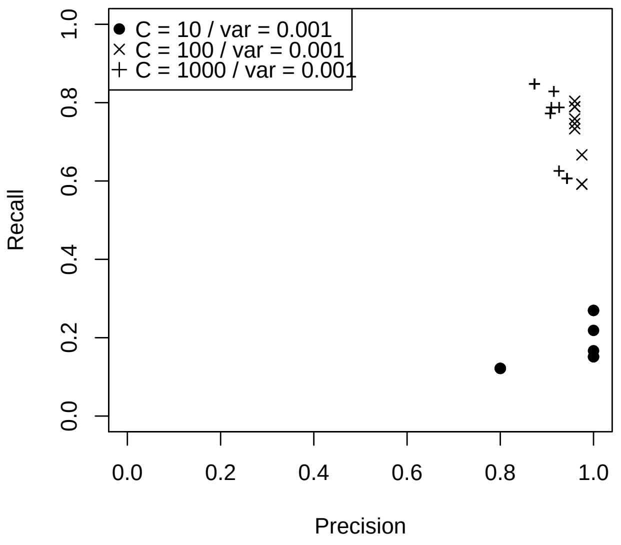
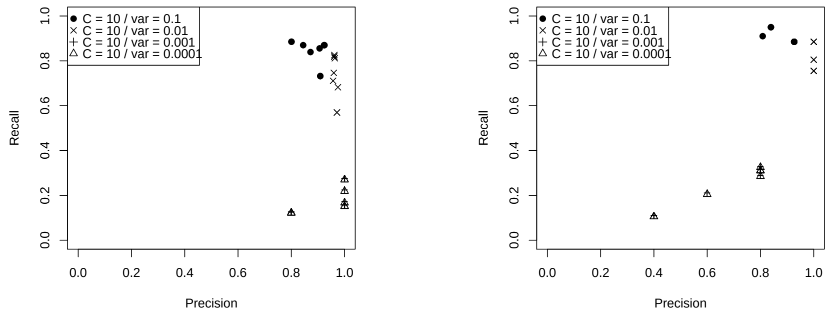


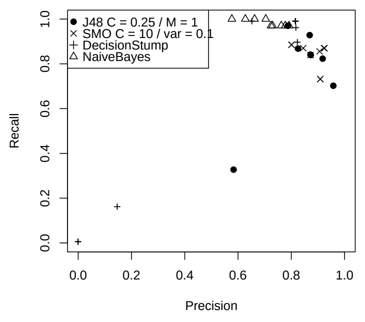
Figure 2: The effect of the regularization coefficient C on the SVM for the Sentencing tag.



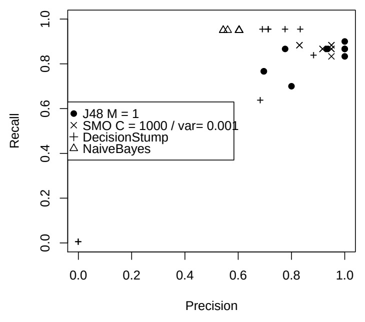
(a) Precision-recall curve for the SVM on the Sentencing tag.

(b) Precision-recall curve for the SVM on the Intellectual Property tag.

Figure 3: The effect of the variance in the SVM.



(a) Precision-recall curve on the Sentencing tag.



(b) Precision-recall on the Immigration tag.

Figure 4: A comparison of different algorithms

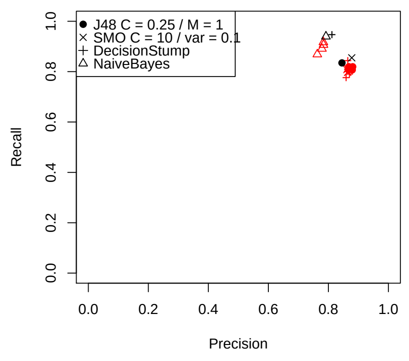
is sufficient for classifiers that do better than random guessing on binary classification problems, which is the situation here.

In this domain AdaBoost sometimes gave a significant improvement on the decision stump and naïve Bayes classifiers but seemed to offer little advantage when used with more advanced classifiers. In those cases it tended to improve either the precision or the recall, but rarely both, and sometimes neither. And where it did improve one it was often offset by a significant worsening in the other. See figure 5 for more details; AdaBoost.M1 was run with 5, 10, 15, and 20 iterations, and the results (red marks) compared to the base classifier (black marks).

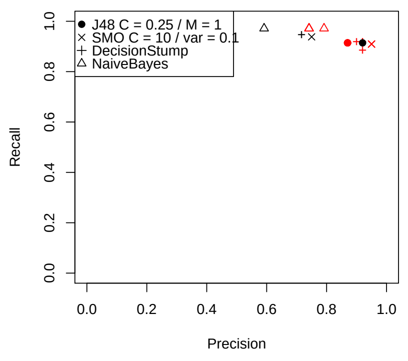
One possible reason for this is that AdaBoost is intended to work with weak classifiers, and even the more simple classifiers are often not really "weak" on this dataset. Observe that, when the base learner does have poor performance, the boosted learner improves significantly. In fact, on the Immigration and Intellectual Property tags, boosted learners are competitive with the best even when the base classifiers are not. In any case, the boosted learners can be considered among several candidates in the model selection process.

5 Using citation data

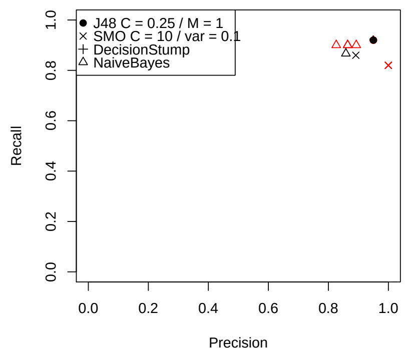
As discussed above in section 2.2, the citation structure of legal opinions is easy to extract, so it is worth considering whether this structure could be



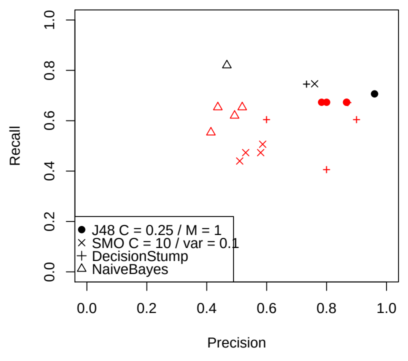
(a) Sentencing



(b) Immigration



(c) Intellectual Property



(d) Discrimination

Figure 5: The effect of boosting. Red marks are for the boosted classifiers.

used to strengthen the results gotten by the bag-of-words features.

Although the opinions in the present corpus are too close in time to cite each other, it is common for two or more opinions to cite the same other authority, such as a statute or an earlier opinion. One can thus create a "social network" from the opinions in which the relationship between two opinions is based on co-occurrence of citations between those opinions. It may then be possible to exploit this network structure in learning.

5.1 Stochastic blockmodeling

It is not obvious how the relationship between two nodes should influence their likelihood of receiving the same tag: many pairs of opinions which do not have any tag in common will still have citations in common, and many pairs which do have tags in common will not happen to cite the same other authority. One way of attempting to discover latent group structure in a social network is stochastic blockmodeling [NS01, KGT04]. This model of a social network assumes that each node belongs to a single group, and the probability of two nodes having a given relationship depends only on the groups that the nodes belong to. Relationships between different pairs of nodes are assumed to be conditionally independent given group memberships.

For this application I use the model described by Nowicki and Snijders in [NS01], in which group memberships are assumed to be generated independently according to a common multinomial parameter, and relationships are generated independently as multinomial random variables given the group memberships. The model parameters are the group-membership multinomial parameter θ and the vector η, where θ_k is the prior probability of belonging to group k and η^b_{hk} is the probability that a node of group h and a node of group k have a relationship of type b. I chose to define the relationship between each pair of opinions as having type 0, 1, 2, or 3, with the types defined as follows:

- Type 3: The two opinions have an identical citation (both cite the same prior case, same section in a statute, etc.).
- Type 1 or Type 2: There is a pair of citations which refer to a similar source, but are not identical. For example, both opinions might cite from the same title in U.S. Code but not the same section. Type 2 indicates greater similarity than Type 1.
- Type 0: One opinion has no citation that is even similar to any citation in the other.

To take advantage of the blockmodel I used the following simple scheme: choose a subset of non-intersecting tags $t_1, \ldots, t_d$ and assign groups to the training set, placing a training example in group i if it is labeled with tag t_i and in group 0 if it is not labeled with any of the tags $t_1, \ldots, t_d$. Then find (Laplace-smoothed) maximum-likelihood parameter estimates for the model parameters $\{\eta^b_{hk} : 0 \leq h, k \leq d;\ 0 \leq b \leq 3\}$ and $\{\theta_k : 0 \leq k \leq d\}$. (I did not use an explicit prior.) Now these parameter estimates can be used to estimate the probability of membership in each group for a new, unlabeled example based on its relationships with the labeled examples.

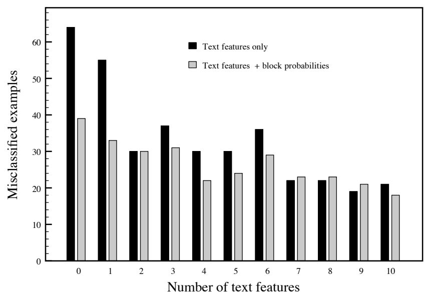
5.2 Experimental Results

To evaluate this method of using citation structure I tested it on the entire labeled data set with the Immigration, Sentencing, Discrimination, and Intellectual Property tags. For each opinion i of the 348 labeled examples I calculated group membership probabilities $\{p^i_k\}^5_{k=0}$ using the maximum-likelihood parameters derived from the other 347 labeled examples, where p^i_k is the probability (under the ML parameters) that opinion i has tag t_k (i.e., that it belongs to group k), and p^i_0 is the probability that opinion i does not have any of those four tags (belongs to group 0).

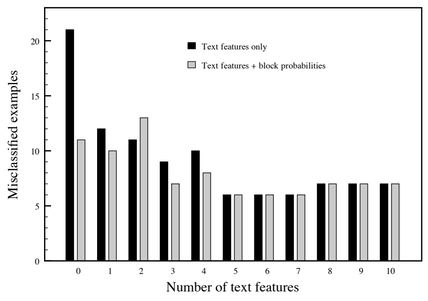
I then added these group membership probabilities as features and tested whether they help in classification. I did not use all the group membership probabilities in every tagging task, as that tended to cause overfitting. Instead, when creating the input for the classifier for tag t_k, I added the computed probabilities p_k and p_0, i.e., the probability of having tag t_k and the probability of having no tag.

In general, the word and bigram occurrences alone are very informative and adding citation information to the entire feature set helped little if at all. However, some insight into the interaction of the citation data with the textual information can be gained by showing how citation data helps (or does not help) classification when only some subset of the textual features are considered. To that end, I ran the following experiment: select the top ten word or bigram features by information gain, and use the J48 tree learner (with M set to 1; see section 3.5.1) to learn on various subsets of the ten word features with and without group probabilities from the blockmodel parameter estimation.

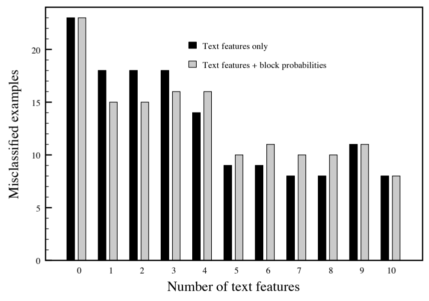
The cross-validation errors from these experiments are shown in figure 6. For the Sentencing and Immigration tags (6(a) and 6(b)) it is clear that the blockmodel features help significantly by themselves or with only a single word or bigram feature. But in the presence of two or more of the



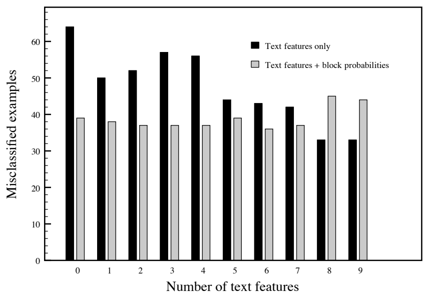
(a) Sentencing (64 positive examples) with the top ten text features



(b) Immigration (21 positive examples) with the top ten text features



(c) Discrimination (23 positive examples) with the top ten text features



(d) Sentencing (64 positive examples) with text features 11-20

Figure 6: The effect of adding group probabilities from blockmodeling to the text features. See the text for more details.

most informative text features, the blockmodel information appears to be redundant.

For the Discrimination tag (figure 6(c)) it appears that blockmodel information does not help much, but the data does show an interesting feature: by itself the blockmodel information is useless; the tree simply classifies all examples as negative. But in the presence of a single text feature the blockmodel reduces error significantly. That is, the number of correctly classified examples for one text feature with blockmodel is more than the sum of the number of correctly classified examples by the blockmodel alone and by the text feature alone. This suggests that the blockmodel information combines well with the textual information.

Finally, figure 6(d) shows the result of combining blockmodel probabilities with the text features ranked 11 through 20 in terms of information gain for Sentencing. Here we see that the blockmodel probabilities help more consistently, indicating that this method may be of particular use in situations where the text features provide less information.

6 Conclusion

The results from this project suggest that simple machine learning techniques apply well to the problem of labeling legal opinions with topics. Either J48 or SMO is able to make good use of simple bag-of-words features and get high classification accuracy.

More simple algorithms did not perform as well by themselves, but with the help of boosting they frequently improved significantly. This verifies that boosting is helpful in cases where the base classifier is not getting all the information it can from the training data, but in general (as in the cases of J48 and SMO) may not create a significant improvement.

Furthermore, stochastic blockmodeling was shown to make effective use of the citation structure in the opinions. Although the complete bag-of-words feature information generally seemed to render the citation information mostly redundant, blockmodeling on citation structure provided clear gains when only limited textual information was available.

Directions for possible future investigation include the use of more advanced textual features and learning algorithms. Furthermore, the data used for learning here was limited and had some errors; more and better-quality data may improve results. Finally, in a future paper (Master's project report, to be completed in the upcoming weeks) I will compare stochastic blockmodeling to other ways of using the citation structure.

References

[Dom99] Pedro Domingos, *Metacost: A general method for making classifiers cost-sensitive*, Knowledge Discovery and Data Mining, 1999, pp. 155–164.

[FS96] Yoav Freund and Robert E. Schapire, *Experiments with a new boosting algorithm*, International Conference on Machine Learning, 1996, pp. 148–156.

[KGT04] Charles Kemp, Thomas L. Griffiths, and Joshua B. Tenenbaum, *Discovering latent classes in relational data.*

[MN98] A. McCallum and K. Nigam, *A comparison of event models for naive bayes text classification*, 1998.

[NS01] Krzysztof Nowicki and Tom A. B. Snijders, *Estimation and prediction for stochastic blockstructures*, Journal of the American Statistical Association **96** (2001), no. 455, 1077.

[Pla98] J. Platt, *Sequential minimal optimization: A fast algorithm for training support vector machines*, 1998.

[Por97] M. F. Porter, *An algorithm for suffix stripping*, 313–316.

[Qui93] J. Ross Quinlan, *C4.5: programs for machine learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Tho01] Paul Thompson, *Automatic categorization of case law*, ICAIL '01: Proceedings of the 8th international conference on Artificial intelligence and law (New York, NY, USA), ACM, 2001, pp. 70–77.