Applying Support Vector Machines to the TREC-2001 Batch Filtering and Routing Tasks

David D. Lewis
Independent Consultant
858 W. Armitage Ave., #296
Chicago, IL 60614
Dave@DavidDLewis.com
www.daviddlewis.com

1. Introduction

My goal for TREC-2001 was simple: submit some runs (so that I could attend the conference), spend the minimum time necessary (since I've been busy this year with a large client project), and get respectable results (marketing!). The TREC batch filtering task was the obvious choice, since this year it was purely and simply a text categorization task.

2. Learning Algorithm

Given the large training set available for batch filtering, choosing a supervised learning algorithm that would make effective use of this data was critical. The support vector machine approach (SVM) to training linear classifiers has outperformed competing approaches in a number of recent text categorization studies, particularly for categories with substantial numbers of positive training examples. SVMs require little or no feature selection, since they avoid overfitting by optimizing a margin-based criterion rather than one based on number of features. This minimizes the complexity of the software and processing. Finally, Thorsten Joachims has made publicly available an efficient implementation of SVMs, *SVM_Light* [Joachims 1999]:

http://www.joachims.org/svm_light/

SVM_Light allows training of both linear and, via kernels, nonlinear classifiers. I used linear classifiers in all cases. Indeed, I left all *SVM_Light* options that affect learning at their default values except *-j*, which controls the relative weight of positive and negative training examples in computing the margin-based loss criterion that SVM's optimize.

I modified *SVM_Light* to accept a comment before each example specifying a document ID, and to output during classification records containing score, predicted class, true class (if present in the test data), and document ID.

3. Tuning the Weighting of Positive and Negative Examples

My experiments focused on the relative weighting of positive and negative training examples. This was due to two problems I anticipated with using SVMs:

Problem 1. Past text categorization experiments have suggested that SVMs are less dominant over competing algorithms on categories with very few positive training examples. A plausible explanation is that the orientation of the learned hyperplane is being determined almost completely by the negative examples. In some machine learning tasks, the positive and negative classes are equally coherent, and a classifier fit to either will produce good effectiveness on the binary classification problem. This is rarely true in text categorization, however. The positive class is typically a coherent subset (e.g. "Retail Sales") of all possible documents, but the negative class is the less well-defined "everything else". Therefore, telling *SVM_Light* to pay more attention to positive examples for low frequency classes seemed like a good idea.

Problem 2. *SVM_Light* by default optimizes a margin-based loss measure which gives equal weight to positive and negative examples. It has been proven that optimizing this measure will tend to lead to low error rate; error rate also gives equal weight to positive and negative examples. However, the TREC batch filtering task used two effectiveness measures, T10SU and T10F, which give unequal weights to positive and negative examples and, moreover, were likely to require two different classifiers to optimize. (See the TREC-2001 filtering track report in this volume for more on these measures.) This again suggested paying attention to the weighting of positive and negative examples.

A typical approach to Problem 2 would be to train using *SVM_Light*'s usual criterion, producing a linear model with a threshold of 0. In a second phase, one would search for a new threshold value that optimizes the TREC effectiveness measure on the training set, while leaving the rest of the parameters unchanged [Lewis, et al 1996]. Zhang & Oles recently used this approach with SVMs [Zhang & Oles, 2001]. This approach assumes, however, that the optimal orientation of the hyperplane is the same for all effectiveness measures, something which is not at all clear. Further, it does nothing about Problem 1. I therefore took the opposite approach, which was to leave the threshold at 0, and try to force the fitting process for the other parameters to adapt to the TREC filtering measures.

Happily, *SVM_Light* has a parameter that controls the relative weight of positive and negative examples in its loss function. Since the T10SU measure corresponds to a weighting of positive examples to be 2 times more important than negative examples, an obvious approach would be to use that same relative weighting in training, at least in producing classifiers for the T10SU measure. I rejected that approach for two reasons:

1. While it has been proven that equally weighting positive and negative examples in SVM Light's loss function leads to high accuracy (i.e. high utility with an equal weighting on positive and negative examples), this result has not been shown to carry

through to unequal weightings of positive and negative examples. It seemed possible, indeed likely, that the scaling factors for the two measures would be different. In any case, as with most computational learning theory results, those for SVMs are too loose to constrain parameter settings tightly.

2. Something had to be done about the F-measure, which does not correspond to any simple relative weighting of positive to negative examples. (Indeed, the F-measure can't be optimized by any predetermined threshold [Lewis 1995], but I ignored this problem for TREC-2001.)

I therefore took a brute force approach. I did multiple training runs for each topic with relative weightings of positive to negative examples of 0.4, 0.6, 0.8, 0.9, 1.0, 2.0, 4.0, and 8.0.

This gave me 8 classifiers for each topic, from which I needed to choose a single classifier for each of the two effectiveness measures. I considered three methods:

Method 1. Evaluate on training data: Using each classifier to classify the training data, comparing those classifications with the true labels, computing the effectiveness on the appropriate measure, and picking the classifier with best effectiveness. The obvious problem with this approach is overfitting: the effectiveness estimates will be too optimistic. If the estimates were *systematically* too optimistic, then the choice of classifier would not be affected. However, that seemed too much to hope for, particularly with a nonlinear effectiveness measure such as T10F.

Method 2. Leave-one-out cross-validation: In cross-validation, one breaks the training data into *k* subsets. A classifier is trained on the union of *k-1* of the subsets, and evaluated on the *k*th subset. The process is repeated *k* times, using each of the subsets as the validation subset once. One then combines the results from the validation subsets to get an overall estimate of the effectiveness of the training procedure. The most extreme and most accurate version of cross-validation is *leave-one-out cross validation (LOOCV)*, i.e. doing *n*-fold cross validation when there are *n* training examples. Cross validation can be used to choose a parameter setting by making a cross-validated estimate of effectiveness at several values of the parameter, choosing the parameter value with highest estimated effectiveness, and then doing a final training run with all data using the chosen parameter setting. (Or one can train using all training data on all choices of parameter setting in advance, and then use cross-validation to pick the best of the already trained classifiers, as I did.)

Method 3. xi-alpha estimation: *SVM_Light* incorporates a highly efficient approximation to LOOCV called *xi-alpha estimation* [Joachims 2000].

It is important to stress that none of these three methods make any use of the test data.

Given the computational expense of Method 2, I first investigated Methods 1 and 3. I had high hopes for the xi-alpha estimate but found its predictions of effectiveness seemed

both unrealistically pessimistic, and varied with the example weighting parameter in ways that seemed intuitively wrong. On the other hand, the estimates of Method 1 seemed unrealistically optimistic in many cases, as well as disagreeing strongly with the Method 3 estimates.

I therefore used the more expensive but accurate Method 2. SVM_Light includes support for LOOCV which, in the case of TREC 2001 batch filtering, meant 23,307-fold cross validation. Using this to choose among 8 values of a weighting parameter in theory meant training $8 \times 23{,}307$ classifiers, each on 23,306 examples, for each of 84 categories.

Fortunately, the properties of the SVM algorithm are such that many of the results of LOOCV folds can be predicted from a run on the full training data, without actually doing the training on the subset. *SVM_Light* incorporates an optimization that prunes away the folds that do not need to be explicitly run, and it meant that typically only a few hundred to a few thousand of the LOOCV folds were actually run for each setting, rather than 23,307 folds. In addition, I used a slightly aggressive version of pruning (the options -x 1 and -o 1) known to work well on text classification problems [Joachims, 2000] instead of the exactly correct version of pruning (options -x 1 and -o 2). Still, several weeks of computing time on a 700MHz PC were required to generate the results for 8 parameter settings and 84 categories.

A minor complexity was that *SVM_Light* output LOOCV and xi-alpha estimates of recall, precision, and error, but I really wanted estimates of T10F and T10U. I therefore wrote code to work backwards from the estimates that were printed (to only 4 digits of accuracy) to the actual contingency table entries, taking into account knowledge of the number of training examples and the number of positive examples for a topic. The code then computed estimates for T10F and T10U from the LOOCV contingency table entries.

These LOOCV-based effectiveness predictions were used to choose two sets of 84 classifiers, one set submitted for T10SU (run *DLewis01bfUa*) and one for T10F (run *DLewis01bfFa*).

I also used these two sets of classifiers to rank the test documents and submitted rankings of the top 1000 documents for each topic (runs *DLewis01rUa* and *DLewis01rFa*) for the routing evaluation.

4. Text Representation

SVMs, like most approaches in both machine learning and IR, require text to be reduced to vectors of numeric feature values.

Our text processing was minimal, consisting of downcasing the text, replacing numbers and punctuation with whitespace, and breaking the text into tokens at whitespace. No stemming was used. I discarded words on the SMART stoplist.

One innovation was motivated by the robustness of SVMs to large feature sets. It is common in IR to give additional prominence in a document representation to words that occur in the document title, for instance by counting them twice. It seems likely that this is a good strategy for some words but a poor one for others. To allow the learning algorithm to choose, I created two sets of features:

1. A set of binary features corresponding to the presence or absence of each word in the title.
2. A set of features for the total number of words in both the title and body of the document. Log TF weighting was applied to this feature set.

These two sets of features were combined into a single feature set used to represent documents. With respect to this feature set, there are linear models that give a variety of ranges of prominence to title words vs. words in the document body.

No IDF weighting or other corpus weighting was used. I did, however, apply cosine normalization to the feature vectors, as the default settings of *SVM_Light* are designed with this in mind.

5. Results

See the Appendix to these proceedings for the raw data. Comparing each of my runs to the other 18 submitted runs as evaluated by the measure I submitted that run for, I found:

- *DLewis01bfUa*, my T10SU run, had greater than or equal to median T10SU on 82 of 84 topics, and equaled the maximum T10SU score on 61 of 84 topics.

- *DLewis01bfFa*, my T10F run, had greater than or equal to median T10F on 81 of 84 topics, and equaled the maximum T10F score on 49 of 84 topics.

- *DLewis01rFa*, my T10F run as submitted for the routing evaluation, had greater than or equal to median SAP (simple average precision) on 83 of 84 topics, and equaled the maximum SAP score on 61 of 84 topics.

What I had intended to be quick-and-dirty runs using OPS (other people's software) were unusually dominant in a relatively mature TREC track. I believe the most important factors in this performance were:

1) The good fit of the SVM approach to text classification problems,
2) Exploring a range of relative weightings of positive and negative examples as a uniform way of addressing both the scaling and thresholding problems, and
3) Use of the computationally expensive but very accurate LOO approach for choosing a distinct relative weighting for each topic.

Future work will test the relative importance of these factors versus, for example, choice of text representation.

6. Afterword: High Frequency Topics for Dinner? Yum!

Prior to the submission of TREC Filtering runs, I made two dinner bets on the outcome of the routing evaluation. The outcome of one of these has been determined as of the writing of this paper. Here's the history:

1. Avi Arampatzis wrote (15-August-2001) to the TREC filtering mailing list, worrying that using only the top 1000 docs in the routing evaluation wouldn't be meaningful, because there were too many positive test documents.

2. As part of the discussion of Avi's obvservation, I wrote: "While I have not looked at the test data labels, I'll go out on a limb and predict that many groups will have have [sic] test set precision @ 1000 over 90% for a nontrivial number of topics. That suggests that any interesting differences between systems will only kick [sic] among documents well below rank 1000..."

4. Chris Buckley wrote "I'd be surprised with P @ 1000 of over 90% for any topics except those that are defined by a single keyword. That's a comment about reliability of the target categorization, not on system performance. Ie, the system may find 950 documents that should be in the category, but only 850 of those were actually assigned the category."

5. I wrote Chris off the list betting dinner that some system would get P @ 1000 of over 90% for some topic that was not defined by a single keyword. We discussed a bit how "single keyword" would defined and he accepted. (Basically, if Chris can write a single word query that gets P @ 1000 of 90% or more, the topic doesn't count.)

6. Separately, Paul Kantor wrote me and the list that "I will buy you a dinner if any system gets 90% @ 1000 for any topic." These were much looser terms than I'd already proposed to Chris, so I happily accepted.

7. Paul conceded on the list on September 6, 2001, after the preliminary results were released and several groups reported 90% @ 1000 results on 30 or so of the topics. I had a nice dinner with Paul at TREC 2001.

8. I am eagerly awaiting the result of Chris Buckley's single word queries to see who buys dinner, perhaps in Finland at SIGIR 2002.

Precision of 90% at 1000 documents seems to conflict with that fact that interindexer consistency rates are often 60% or lower [Cleverdon, 1991]. If two people can't agree more often than that, how can a machine do better? However, interindexer consistency is measuring agreement on an entire collection of documents. (In practice it is typically estimated by sampling techniques.) I made the above bets based on a suspicion that high scoring documents are different from collections as a whole. That is, if a well-trained statistical classifier is very confident a document belongs to a class, then it must be an "easy" document that almost any human indexer would assign to the class. The interindexer consistency, and thus the possible machine/indexer consistency, would be

much higher than average. Since some routing topics in TREC 2001 were clearly going to have 10's of thousands of relevant documents, the top 1000 would consist of documents with very high scores.

In addition, at the time I made the bets I already knew that *SVM_Light*'s LOOCV estimates of precision on the *entire* training set were above 90% for some topics. Since LOOCV estimates are (almost) statistically unbiased, I was confident that results on the *entire* test set would be similar, and that on the top 1000 would be even better. As it turned out, some groups had precision of 100% on their top 1000 test documents for some topics.

On a serious note, it's clear that Avi Arampatzis' original worry was on the mark: evaluating routing runs on only the top 1000 documents meant that there was very little distinction among systems for high frequency topics. I will go further, and suggest that any ranking-oriented evaluation is of questionable interest when one has both substantial training data and thousands of relevant documents.

References

[Cleverdon 1991] C. Cleverdon. The Significance of the Cranfield Tests on Index Languages. In A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan, editors, *SIGIR '91: Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 3-12, New York, 1991. Association for Computing Machinery.

[Joachims 1999] T. Joachims. Making Large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[Joachims 2000] T. Joachims. Estimating the Generalization performance of an SVM Efficiently. *International Conference on Machine Learning (ICML)*, 2000.

[Lewis 1995] D. Lewis. Evaluating and optimizing autonomous text classification systems. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR '95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246-254, New York, 1995. Association for Computing Machinery.

[Lewis, et al 1996] D. Lewis, R. Schapire, J. Callan, and R. Papka. Training algorithms for linear text classifiers. In Hans-Peter Frei, Donna Harman, Peter Schauble, and Ross Wilkinson, editors, *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298-306, Konstanz, 1996. Hartung-Gorre Verlag.

[Zhang and Oles, 2001] T. Zhang and F. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, v. 4, pp. 5--31, 2001.