IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4791

Using Boosting in a Multilabel Text Categorization Problem

Jorge Muruzábal and Eduardo G. Souto
Statistics and Decision Sciences Group
University Rey Juan Carlos
28933 Móstoles, Spain
Tel: ++31916647474
Fax: ++34916647490
j.muruzabal@escet.urjc.es
edugs@telefonica.net

ABSTRACT

The issue of learning to predict the various *specific subjects or categories to which individual pieces of text belong is central in automatic classification. The Boosting scheme has been extensively tested and show an interesting record of success. In this paper we examine its performance in a well-known database of Medline research documents known as* cystic fibrosis. *This database is highly versatile for systematic experimentation. We empirically demonstrate this approach, discuss its overall usefulness and scope, and provide a detailed roadmap of further research in the area.*

1. INTRODUCTION

In a general text classification problem, data are available as a set of pairs $D=\{(x_i, y_i), i=1,...,n\}$, where each x contains a collection of text words and y encodes the correct categorization in each case. We denote by X and Y the spaces to which x and y belong. Each document's raw material typically needs to be preprocessed to yield the summary x (and hence the space X). On the other hand, $Y=\{1,...,k\}$, where k is the number of output categories; usually $k>2$. If each y_i corresponds to a single category in Y we are in the *single-label* or standard case. This paradigm is typically not so interesting in text categorization due to the inherent overlapping of output labels. Henceforth, we consider the *multilabel* case where each y_i corresponds to a subset of Y. Ordinarily, it is more difficult to approach this case because of the relative ambiguity in output label membership: different observers may classify items quite differently. The *cystic fibrosis* (CF) database (Shaw et al., 1991) provides an excellent environment for our purposes because *several* membership criteria are available. This database is rather flexible in that it can be readily adapted to formulate a variety of related problems.

In this paper we use this database and analyze the empirical behaviour and practical suitability of one of the most popular machine learning algorithms: Boosting.

The organization is as follows. In Sections 2 we outline some basic aspects of Boosting. Details of the CF database are provided in Section 3. Empirical results are discussed in Section 4. Finally, some conclusions are drawn in Section 5.

2. BOOSTING

Boosting is a general-purpose technique that has been extensively analyzed and successfully applied in a variety of flavours (Schapire and Singer, 1998), (Friedman, Hastie, Tibshirani, 2000), (Denison, 2001). Boosting is actually one of several methods for creating *committees* of rules. Committee-based methods have often been shown to improve over their single-run counterparts, see e.g. (Breiman, 1996).

A distinctive feature of Boosting is the iterated calling to a basic *weak learner* (WL). Each time the WL is invoked, it takes as input (D,Ω), where Ω is a weight distribution over D, and returns a classifier $h: X \to Y^*$, where Y^* is the WL's *prediction space*. The weight distribution changes over time, with the result that more weight is put on cases that turned out to be difficult for previous classifiers. The prediction space Y^* defines the type of search to be performed. In our case, Y^* may be either Y itself or $Y^*=\Re^k$. The latter choice is referred to as AdaBoost.MH (Schapire and Singer, 2000); it will be considered below together with some variants of interest. Note that the absolute numeric coordinates of $h(x)$ directly reflect "predictive confidence". The *strong classifier* H obtained after T training rounds is $H(x)=\sum_{t=1}^{T} h_t(x) \in \Re^k$. A *ranking* of output labels is thus defined by the coordinates of H, the largest entries always corresponding to the preferred labels.

Schapire and Singer (2000) have pioneered the use of Boosting techniques in text categorization. In their BoosTexter software they restrict attention to classifiers h_t based on *stumps*: a single question is posed about x, namely, whether certain pattern or word-gram e_t can be found, and two different set of values h_t^+ and h_t^- are provided. The strong classifier *H* thus becomes

$$H(x)=\sum_{t:e_t\in x} h_t^+(x)+\sum_{t:e_t\notin x} h_t^-(x).$$

The interested reader is referred to Schapire and Singer's original paper for all missing details in this abridged introduction.

3. THE CYSTIC FIBROSIS DATABASE

The cystic fibrosis (CF) database provides a nice environment for systematic comparative experimentation. The output categories in this database correspond to questions (queries), and the judgement of four experts is available on the relevance of each article with regard to each question. For the complete database (containing 1239 documents initially), the average number of documents relevant to a category is about 48. After stopword removal, training documents contain 17911 different terms in total. The input vectors passed to the learning algorithm are thus obtained by scanning the document and retaining these terms in order.

As regards the output $y_i \subset Y,$ we have begun our research with a relatively simple criterion. In the CF database, each question Q_j is an output class, $j = 1, \ldots, 100$, and document x_i is linked to question Q_j via four expert ratings r_{ijl}, $l = 1, \ldots, 4$, each taking values 0, 1 or 2 from irrelevant to fully relevant. We proceed in two steps: all four r_{ijl} are collapsed into a single $R_{ij} = \sum_l r_{ijl} \in \{0, 1, \ldots, 8\}$. Then we set a threshold ρ and split according to $M_{ij} \equiv R_{ij} \geq \rho$. Two choices for ρ are considered: 1 and 6. If $\rho = 6$, then matrix M becomes sparse and the problem is more like standard classification (a portion of documents having 0 rows are removed from the analysis in this case). The average size of the y_i in this case is about 1.4, whereas for $\rho = 1$ it jumps to about 4. Two different versions for each value of ρ are considered. The first version considers only title, keywords and abstract, the second adds author names, references and journal where the article was published. Thus, there are 4 versions of the multilabel problem in total, which we label as $\rho CF \delta$, with $\rho = 1$, 6 and $\delta = 1$ (less detail), 2. Fixed training and test samples, with 755 and 484 documents respectively in the $\rho = 1$ case (485 and 321 for $\rho = 6$), were selected at random once and used systematically for training and evaluation in all runs below.

4. EXPERIMENTAL WORK

In this section we first review the various performance measures that substantiate our experiment. Then we discuss the different variants of the software tools that we have tested.

4.1 Performance Measures

Recall that BoosTexter attempts to learn a sensible *ranking* of output labels H. There are several ways to probe a given H. The simplest approach involves the top rank $l(x) = \arg\max_{1 \leq j \leq k} H^j(x)$, the most confident output label for document x. Indeed, in BoosTexter the primary performance measure provided is one-error, the fraction of the time $l(x) \notin y$ over the test sample (Schapire and Singer, 2000). These authors also consider the average rank that must be descended before all true test labels are checked up, namely, the Average Coverage Rank or $ACR(x)$.

4.2 Algorithm Variants

The BoosTexter software has been used. We have considered standard adaboost.MH as well as the abstaining adaboost.MH and the (discrete) adaboost.MR algorithms. Since the abstaining variant (where all h_t^- are set to 0) nearly always outperforms adaboost.MH in the CF trials, only results from the last two algorithms will be presented here. It is convenient to label these algorithms after their BoosTexter codes (R and L respectively). Abstaining is a frequently considered option in Boosting algorithms (Cohen and Singer, 1999). For details on adaboost.MR, see (Schapire and Singer, 1998, 2000); in this case, $H^j(x) > 0$ for all j and x.

BoosTexter builds on the idea of seeking *contiguous* text patterns or *word-grams* e_t to guide the optimization process. Besides the type of learning scheme R or L, we can select independently the length W and style N of the word-grams e_t. While we have tested the range W≤ 4, it turns out that W$= 1, 2$ yield the best results and are less demanding computationally, so we analyze them in greater detail below. Furthermore, besides N$=$ngram, we have also tried the sgram and fgram options. Sgram allows *sparse* word-grams such as human#*#disease (W$= 3$), whereas fgram enforces the specified limit W in all e_t. Again, for fixed W$= 2$ sgram is nearly always identical to ngram, so it will not be considered. On the other hand, fgram is usually much worse than ngram, so it is not considered either. Hence, we focus on the BoosTexter variants that we label R-N1, R-N2, L-N1 and L-N2. These refer to the abstaining or .MR basic engines, N$=$ngram (fixed) and W$= 1, 2$ respectively.

4.3 Empirical Results

We begin with the one-error measure alone, see Table 1. BoosTexter seems to have difficulty priming correct labels at the top rank $l(x)$. The abstaining variant R is better than the alternative L, particularly in the high overlap ($\rho = 1$) case. We also note that, rather surprisingly, the added flexibility in W$= 2$ does not mean a clear advantage. As regards CF versions, note that the richer setting ($\delta = 2$) helps substantially Note also that the sparse case ($\rho = 6$) seems to be harder for the four variants.

In Table 2, we look at the ACR measure. We avoid using the usual mean ACR because of the long, distorting right tail; order statis-

Table 1. *One-error test figures for various database settings (rows) and algorihtm variants (columns). See text for details.*

	One-error			
	R-N1	R-N2	L-N1	L-N2
$1CF1$	45.9	47.1	56.4	57.5
$1CF2$	42.5	45.0	52.1	50.3
$6CF1$	64.5	62.6	68.1	68.9
$6CF2$	59.2	55.8	62.5	61.8

Table 2. *First and second quartiles of the* ACR *distribution over documents. The notation "2,6" means that in 50% of all documents the correct labels were among the top 6 labels, whereas in 25% of all documents the correct labels were among the top 2 labels.*

	ACR	
	R-N1	L-N2
$1CF1$	9,24	10,26
$1CF2$	9,22	9,20
$6CF1$	2,6	2,7
$6CF2$	1,5	2,4

tics provide a more robust picture. In the high overlap ($\rho = 1$) case, for example, we would need to link each document to the top 20 labels (out of 100). There is clearly room for improvement in as far as documents would be retrieved quite often in this case.

5. DISCUSSION

We have investigated one of the most popular machine learning algorithms in the CF multilabel text categorization problem. A number of factors concerning the particular BoosTexter design can be distinguished. For one thing, stumps may not have enough expressive power by themselves to effectively separate out the output categories. Stumps indeed are not recommended in situations where a great deal of the game is played by predictor *interactions* (word co-occurrences in our case), see (Friedman et al., 2000). A natural alternative to stumps are trees of depth 2 (typically involving 4 terminal nodes).

Another improvement would be the inclusion within the Boosting approach of the word frequency information. Preliminary results with algorithms using this kind of information (SVMs or Bayesian schemes) show that a given term's classification *relevance* appears directly concerned with its turnover rate. This should not mean a substantial increase in complexity and deserve further investigation.

ACKNOWLEDGMENT

The authors are partially supported by grant TIC2001-0175-C03-03. We are much grateful to Javier M. Moguerza for his helpful comments.

REFERENCES

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24(2), 123-140.

Cohen, W. W. & Singer, Y. (1999). A Simple, Fast, and Effective Rule Learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. AAAI Press.

Denison, D. G. T. (2001). Boosting with Bayesian Stumps. *Statistics and Computing*, 11, 171-178.

Freund, Y., Mansour, Y. & Schapire, R. E. (2001). Why Averaging Classifiers Can Protect Against Overfitting. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*.

Friedman, J., Hastie, T. & Tibshirani, R. (2000). Additive Logistic Regression: a Statistical View of Boosting (with discussion). *The Annals of Statistics*, 28(2), 337-407.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. S. (1998). Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5), 1651-1686.

Schapire, R. E. & Singer, Y. (1998). Improved Boosting Algorithms using Confidence-rated Predictions. *Machine Learning*, 37(3), 297-336.

Schapire, R. E. & Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3), 135-168.

Shaw, W. M., Wood, J. B., Wood, R. E. & Tibbo, H. R. (1991). The Cystic Fibrosis Database: Content and Research Opportunities. *Library & Information Science Research*, 13, 347-366.