# Improving SVM Text Classification Performance through Threshold Adjustment

James G. Shanahan and Norbert Roma

Clairvoyance Corporation, 5001 Baum Boulevard, Suite 700,
Pittsburgh, PA 15213-1854, USA
{jimi,n.roma}@clairvoyancecorp.com

**Abstract.** In general, support vector machines (SVM), when applied to text classification provide excellent precision, but poor recall. One means of customizing SVMs to improve recall, is to adjust the threshold associated with an SVM. We describe an automatic process for adjusting the thresholds of generic SVM which incorporates a user utility model, an integral part of an information management system. By using thresholds based on utility models and the ranking properties of classifiers, it is possible to overcome the precision bias of SVMs and insure robust performance in recall across a wide variety of topics, even when training data are sparse. Evaluations on TREC data show that our proposed threshold adjusting algorithm boosts the performance of baseline SVMs by at least 20% for standard information retrieval measures.

## 1 Introduction

Generic support vector machines (SVMs) [19] provide excellent performance on a variety of learning problems including: handwritten character recognition [8], face detection [15] and most recently text categorization [6]. However, when generic SVMs are applied to text classification[1], their performance, while being competitive with other approaches (e.g., Rocchio, naïve Bayes) from a precision perspective, is not competitive from a recall perspective [6], [17].

Several attempts have been made to improve the recall of SVMs while not adversely affecting precision in a text classification context. The first category of such attempts falls under the label of uneven margin-based learning [12]. Here, a simple margin-based version of the perceptron learning algorithm is used to learn a model that has a pre-specified required positive and negative margin. The required positive and negative margins are heuristically determined using cross-validation on the training corpus. The second category of proposed SVM improvements for text classification is cost-based and is also incorporated into the SVM learning algorithm. To counter the imbalance of positive training documents to negative training documents,

[1] Text classification is a very active area of research and application in information management and is concerned with assigning a document to one or more pre-specified categories or classes.

a higher cost is associated with the misclassification of positive documents than with negative documents [20]. Tuning the asymmetric misclassification cost can provide significant improvement, though this process can be prohibitively expensive. The final category of proposed SVM improvements for text classification is based on post-processing or thresholding the output value (or margin/score) of the learnt SVM. This is generally an inexpensive one-dimensional optimization problem which can lead to significant improvement in performance measures.

This post-processing thresholding step is independent of the learning step. The critical step in thresholding is to determine the value, known as the threshold, at which a decision changes from labeling a document as positive to labeling a document as negative. Many of the approaches to thresholding that have been developed in other fields (such as information retrieval) can be applied directly in thresholding the score output of SVMs. Though thresholding has received a lot attention in the information retrieval sub-field of adaptive filtering, optimizing thresholds remains a challenging problem. The main challenge arises from a lack of labeled training data. Due to limited amounts of training data, standard approaches to information retrieval use the same data for both model fitting (learning) and threshold optimization. Consequently, this often biases the threshold to high precision, i.e., overfits the training data.

The following provides a brief overview of information retrieval-based thresholding approaches: Yang presents an empirical study of a variety of thresholding strategies for text categorization using *k* nearest neighbors [22]; Zhai. et al. present a beta-gamma thresholding algorithm for adaptive filtering, which has been adapted in the thresholding strategy proposed in this paper [23]; Zhang and Callan propose a maximum likelihood estimation of filtering thresholds [24]; Ault and Yang introduce a margin-based local regression approach for predicting optimal thresholds for adaptive filtering [2]; Arampatzis describe a score-distributional threshold optimization approach [1].

Some of these IR approaches have been adapted already for thresholding SVMs. Cancedda et al. report one such approach to adjusting the threshold of SVMs based upon a Gaussian modeling process of the SVM scores (output value) for positive and negative documents for each category [3]. This Gaussian model is then used to generate sample document scores and an optimal threshold is set to the score corresponding to maximum utility on the cumulative utility curve for the generated labeled scores. This approach, combined with asymmetric learning, has led to huge improvements in recall and precision, though it is hard to discern how much improvement can be attributed to the asymmetric cost learning strategy or to the thresholding strategy. This impact of adjusting the threshold will become clearer later in this paper when we show that it can boost significantly the performance of baseline SVMs for text classification.

In this paper, we adapt a procedure of setting the threshold of the learnt SVM using the beta-gamma thresholding technique, developed previously for adaptive text filtering using information retrieval-based filters [22], a more challenging task than text classification. In addition, we present a novel and very cheap technique for selecting the parameters of the threshold adjustment strategy automatically, based upon cross fold validation. This paper is organized as follows: Section 2 describes the proposed threshold adjustment algorithm after a brief overview of generic linear SVM modeling; Section 3 describes the experimental setup, detailing the explored variables and datasets used to evaluate the proposed approach; Section 4 presents the results of evaluations of the proposed approach and compares these to other approaches; Section 5 presents some concluding remarks.

## 2 Proposed Thresholding Approach

The proposed threshold adjustment algorithm is performed immediately after learning an SVM. In this section, we first present some background material on SVMs and then present the proposed threshold adjusting algorithm.

### 2.1 Support Vector Machines

Though support vector machines (SVM) were originally introduced by Vapnik in 1979 [19], and have provided state-of-the-art performance for a variety of learning problems (and in some cases better than state-of-the-art), it is only recently that they have gained popularity in the text retrieval and classification community. Geometrically (for linear support vector machines), a learnt SVM model can be seen as a hyperplane that separates a set of positive examples (belonging to the positive class) from a set of negative examples (negative class). This is illustrated in Figure 1, where H is a hyperplane that separates positive class examples (denoted by "+") and negative class examples (denoted by "-"). Mathematically a hyperplane can be represented as follows:

$$\left(\sum_{i=1}^{n} w_i x_i\right) + b = 0 \qquad (1)$$

This can be written more succinctly in vector format as *<W,X>+b =0*. Here *W* is known as a weight vector and corresponds to the normal vector to the separating hyperplane, H, and *X* is an input vector or document. *b* denotes the perpendicular distance from the hyperplane to the origin. *n* represents the number of input variables, in the case of text, this can be viewed as the number of words (or phrases, etc.) that are used to describe a document. The classification rule for an unlabeled document, *X*, using a support vector machine with separating hyperplane *(W, b)*, is as follows:

$$Class(X) = Sign(\langle W, X \rangle + b) \qquad (2)$$

The distance from the hyperplane to the nearest positive or negative examples is known as the margin of the SVM. Learning a linear SVM can be simply thought of as searching for a hyperplane (i.e., the weights and bias values) that separates the data with the largest margin. As a result, learning for linearly separable data can be viewed as the following optimization problem:

$$\text{minimize}\left(\frac{\|w\|^2}{2}\right) \text{ subject to: } y_i(\langle W, X_i \rangle + b) \geq 1 \quad \forall\, i = 1,\ldots,n \qquad (3)$$

where $X_i$ is training example with label $y_i$ and $||W||$ is the $L^2$ norm of the weight vector (i.e, $\sqrt{(\Sigma^n_{i=1}(w_i * w_i))}$. In the case of non-linear separablity, two alternative formulations have been proposed: one is based upon slack variables; and the other is based upon using non-linear kernels (see [20]for more details). The slack variable or soft formulation of SVM learning [4] allows, but penalizes, examples that fall on the wrong side of the supporting hyperplanes ($H_+$ and $H_{-1}$ in Figure 1), i.e., false positives or false negatives. Different or asymmetric costs can be associated with false negatives and false positives. In practice, learning SVMs is more efficiently conducted in a dual space [19]. For our current study, two variations of the dual space Sequential Minimal Optimization (SMO) learning algorithm [16] were implemented and evaluated: SMOK1 and SMOK2, corresponding to modification 1 and modification 2, respectively, as proposed by Keerthi et al. [7]. Our current implementation caters only for symmetric false positive and false negative costs.

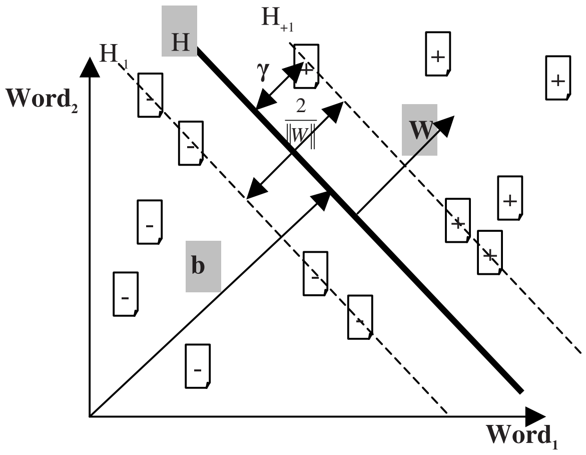

**Fig. 1.** A support vector machine in a two-dimensional input space, $Word_1 \times Word_2$, denoted by the hyperplane, $H$. Each document is associated with a category, ("+" or "-"). The support vectors correspond to the examples on the hyperplanes $H_{+1}$ and $H_{-1}$.

## 2.2 Thresholding Adjusting Algorithm for SVMs

Optimizing thresholds is a challenging problem because the limited amount of training available is generally required for training the base model, thereby, resulting in a situation where it is rare to have an independent sample solely for threshold optimization. Standard approaches in text classification and retrieval use the same data for both model fitting (learning) and threshold optimization [22]. Consequently, this often biases the threshold to high precision, i.e., the threshold overfits the training data. SVM learning algorithms focus on finding the hyperplane that maximizes the margin since this criterion provides a good upper bound of the generalization error. Learning based on this criterion leads to models with very good ranking ability (demonstrated empirically by the results in Section 4). However, the resulting separating hyperplane tends to be too conservative (high precision oriented). The natural threshold value for SVM learning and classification is zero (see Equation 2). Here, we propose to combine the powerful ranking ability of SVMs with the beta-gamma thresholding algorithm [22] to reset the threshold of the learnt SVM in order to overcome this precision-oriented limitation. The powerful ranking ability of SVMs is only exploited for threshold adjustment, and is not used in classification (as each document is classified independently of each other). The beta-gamma thresholding algorithm relaxes the SVM threshold from zero, i.e., translates the SVM hyperplane towards the denser class (i.e., the class with more training data). In addition to adapting the beta-gamma algorithm for adjusting the SVM threshold, we propose a novel means for setting the parameters of this algorithm – beta and gamma – using a cheap cross validation mechanism.

We first present the core beta-gamma thresholding strategy, and subsequently describe how this can be used with cross validation to empirically determine the beta and gamma parameters. The beta-gamma thresholding strategy consists of the following steps and uses as input a category label, *C*, a labeled dataset, *T,* of documents consisting of both positive and negative examples of *C*, a learnt SVM, *M,* that models the category *C*, $\beta$, the threshold adjustment parameter, and *UtililtyMeasure*, a utility measure that models the user's expectations:

```
SetSVMThresholdUsingBetaGamma(C, T, M, $\beta$, UtililtyMeasure)
1. Rank the thresholding dataset, T, using the SVM, M, as scoring function, thereby yielding
   a ranked document list R consisting of tuples <$Document_i$, $SVMScore_i$>.
2. Generate the cumulative utility curve for R, i.e., for each document in the ranked list R
   compute the cumulative utility using the utility measure UtililtyMeasure.
3. Determine the rank or indices of the maximum utility point on the cumulative curve and
   the first zero utility point following the maximum utility point. Denote these respectively
   as $i_{Max}$, and $i_{Zero}$. Assign the variables $\theta_{Max}$ and $\theta_{Zero}$ the output scores of the SVM, M, for
   the documents associated with the maximum and zero utility points respectively, i.e., the
   SVM scores of the documents at rank $i_{Max}$, and $i_{Zero}$. (See Figure 2 for a graphic illustration
   of this step.)
4. Return the threshold, $\theta$, which is calculated as follows:
   $\theta = \beta\theta_{zero} + (1-\beta)\theta_{Max}$        (4)
```

In the procedure outlined above, $\beta$ is either provided heuristically or determined using the beta-gamma cross-validation procedure outlined below. The following is a more sophisticated version of this threshold adjustment algorithm (Equation 4) that takes into account the number of positive training examples used in *T*:

$$\begin{aligned}\theta &= \alpha\theta_{zero} + (1-\alpha)\theta_{Max} \\ \alpha &= \beta + (1-\beta)e^{-p\gamma}.\end{aligned} \tag{5}$$

In this equation, $p$ denotes the number of positive documents in the thresholding dataset, *T*. The $\gamma$ component of this threshold relaxation formulation provides a mechanism to *further* relax the threshold based entirely upon $\beta$ (Equation 4). This will have biggest impact on the threshold when there are very few documents. Once, again, as is the case for $\beta$, $\gamma$ is either provided heuristically or determined using the beta-gamma cross-validation procedure outlined below.

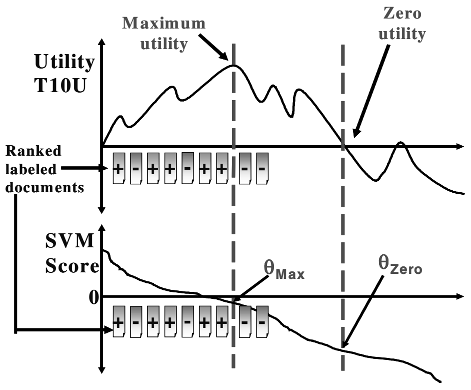

**Fig. 2.** Determining $\theta_{Max}$ and $\theta_{Zero}$ using a ranked list of training documents.

Now, we outline a procedure based upon n-fold cross validation to automatically determine the values of $\beta$ and $\gamma$ in the threshold relaxation procedure. It consists of the following steps and uses as input a category label, *C*, a labeled dataset, *T,* of documents consisting of both positive and negative examples of *C* (for example, *T* could be a subset or the complete training dataset), a learnt SVM, *M,* that models the category *C*, $\beta$, the threshold adjustment parameter, *UtililtyMeasure*, a utility measure that models the user's expectations, $\beta s$ (valid values for $\beta$ are positive or negative real numbers), the set of possible beta values, $\gamma s$, the set of possible gamma values, and *n,* the number of folds that will be used in parameter selection.

```
SelectOptimalSVMThreshold(C, T, M, UtililtyMeasure, βs, γs, n)
1. Partition the data into n non-overlapping subsets of the data ensuring that both positive
   and negative documents are present in each fold or subset.
2. Foreach each combination of β and γ values in βs and γs do steps 3 and 4
3. Foreach fold n
   • Set $T_n$ to the n-1 folds
   • Set $\theta$ = SetSVMThresholdUsingBetaGamma(C, $T_n$, M, β, γ, UtililtyMeasure)
   • Set $Utility_{\beta\gamma}$ = Calculate the utility for M and the threshold, $\theta$, over the fold n. See
     Equation 6 for an explanation of how to use an adjusted threshold in conjunction
     with an SVM.
4. Compute the average utility as follows: $Utility_{\beta\gamma} = Utility_{\beta\gamma}/n$
5. End Foreach
6. Calculate the optimal threshold, $\theta_{Opt}$, using the β and γ combination that has the highest
   average utility $Utility_{\beta\gamma}$ as follows: SetSVMThresholdUsingBetaGamma(C, T, M, β, γ,
   UtililtyMeasure)
7. Return $\theta_{Opt}$.
```

The SVM classification rule is altered slightly as follows to accommodate the adjusted threshold:

$$Class(X) = Sign\left(\langle W, X\rangle + b - \theta_{Opt}\right) \tag{6}$$

For our experiments, we adapted the *T10U* linear utility measure (see Table 2) for threshold optimization, as this provides an intuitive user utility model that generally leads to improved recall and precision when used as a cost function in learning [17].

## 3 Experimental Setup

This section describes the experimental variables, the experimental performance measures and the datasets used for this study. The parameters settings explored in the experiments reported in this paper are summarized in Table 1. All are pretty much self-explanatory, apart from how a document is represented. We represent a document as a vector of terms that is derived as follows: we replace all numerical and punctuation characters by spaces and eliminate stop-words such as articles and prepositions, etc.; each term is associated with a TFxIDF weight, where TF denotes the frequency of a term in a document, and IDF is calculated based on the distribution of the term in the training corpus [18]. In all experiments the document vectors were normalized to unit length.

In our analysis, we examined several information retrieval performance measures which are presented in Table 2 along with their definitions.

**Table 1.** Learning decision variables and explored values.

| Decision Variable | Explored Values |
|---|---|
| Learning Algorithm | SMOK2 |
| C (Upper bound for Lagrange multipliers) | 0.4, 0.8, 0.9, 1, 2, 5 |
| Tolerance | 0.001 |
| Type of kernel | Linear |
| Sampling Ratio | Used all training data |
| Number of terms k | Use all terms |
| Term types | White space delimited tokens with numbers, punctuation, and stopwords removed |
| Term weighting | TF_IDF |

For our current study, we have performed an evaluation of learning threshold adjusted SVM classifiers (TSVMs) on the following classification corpora: Reuters-21578 ModApte split collection [10] and TREC2001 corpus [17]. The main reasons for choosing these corpora include the following: these corpora are commonly used in benchmarking text classification problems; the Reuters-21578 corpus is a manageable size thereby enabling extensive experimentation (without being computationally prohibitive). The details of each corpus are presented below.

### 3.1 Reuters-21578 (ModApte Split)

The Reuters-21578 collection contains 12,902 newswire stories that had been classified into 118 categories (e.g., corporate acquisitions, earnings, money market, grain, and interest) [10]. We followed the ModApte split in which 75% of the stories (9603 stories) are used to build classifiers, while the remaining 25% (3299 stories) are used to test the accuracy of the resulting models in reproducing the manual category assignments. Only 90 categories are modeled in our experiments. These 90 categories were selected based upon having at least one training and one testing example.

Though only 90 categories were modeled, the examples belonging to the non-modeled categories were used for training and testing.

### 3.2 TREC 2001 Corpus

The TREC 2001 Corpus, officially known as "Reuters Corpus, Volume, English Language, 1996-08-20 to 1997-08-19", contains one year of Reuters newswire stories in English, corresponding to 1.5 GB of data, or 810,000 news stories taken from the period August 1996 – August 1997. Each story has been assigned one or more category labels from 84 possibilities. The training dataset is limited to the last 12 days of August 1996 (corresponding to approximately 23,000 examples); the remaining 11 months are designated as test data. More information about this corpus can be found at http://about.reuters.com/researchandstandards/corpus [17].

**Table 2.** Evaluation measures and their definitions, where $R^+$, $N^+$, $R^-$, and $N^-$ are true positives, false positives, false negatives and true negatives respectively.

| **Evaluation Measure** | **Definition** |
|---|---|
| Precision | $p = \frac{R^+}{R^+ + N^+}$ |
| Recall | $r = \frac{R^+}{R^+ + R^-}$ |
| $F_\beta$ | $F_\beta = \frac{(\beta^2 + 1) * p * r}{(\beta^2 * p) + r}$ |
| T10U/ *T11U* | *T10U= T11U = 2R⁺-1N⁺* |
| T10SU | $T10SU = \frac{\max(T10U, MinU) - MinU}{MaxU - MinU}$ where $MaxU = 2*(R^+ + R^-)$ and $MinU = -100$ |
| T11SU | $T11SU = \frac{\max(T11NU, MinNU) - MinNU}{1 - MinNU}$ where $T11NU = \frac{T11U}{MaxU}$ and $MinNU = -0.5$ |

## 4 Results and Empirical Observations

In the case of all examined corpora, a topic-specific binary classifier was learned from the training data that models the topic (positive examples) and the not-topic (negative examples). The values explored for β were restricted to the following list: {-0.05, 0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8}, while γ was set to 100 (effectively disabled). The γ parameter was disabled after noticing no discernable improvement from using it in the context of classification, though this parameter proved to be crucial in an adaptive text filtering context [22], where a topic is defined differently and its definition is adapted over time; usually a topic is defined in terms of a focused query and a small number of explicitly labeled documents; and its definition is refined over time upon receiving user feedback.

The proposed thresholding approach is compared against the following approaches: baseline (unthresholded) SVMs; other threshold adjusting SVM approaches; asymmetric (misclassification costs) SVMs; and traditional IR approaches.

Figure 3 compares the results for the Reuters-21578 corpus between the threshold adjusted SVMs and baseline SVMs for each topic with respect to the T11SU evaluation measure. For this graph of results, and for subsequent graphs of results, the horizontal axis represents the topics (considered in a corpus), ranked in decreasing order of the number of positive training data available for that topic. This graph has two primary vertical or *y* axes; the left vertical axis corresponds the log (base 10) of number of training documents; the right vertical axis corresponds to the difference in performance for the indicated measure (T11SU in the case of Figure 3) between the threshold adjusted SVM (denoted as *SVMThresh*) and the baseline SVM. Positive bars for this measure correspond to an improvement in performance when threshold adjustment is used. Table 3 presents the macro-average results for precision, recall, Fbeta, and T11SU for the Reuters-21578 corpus.

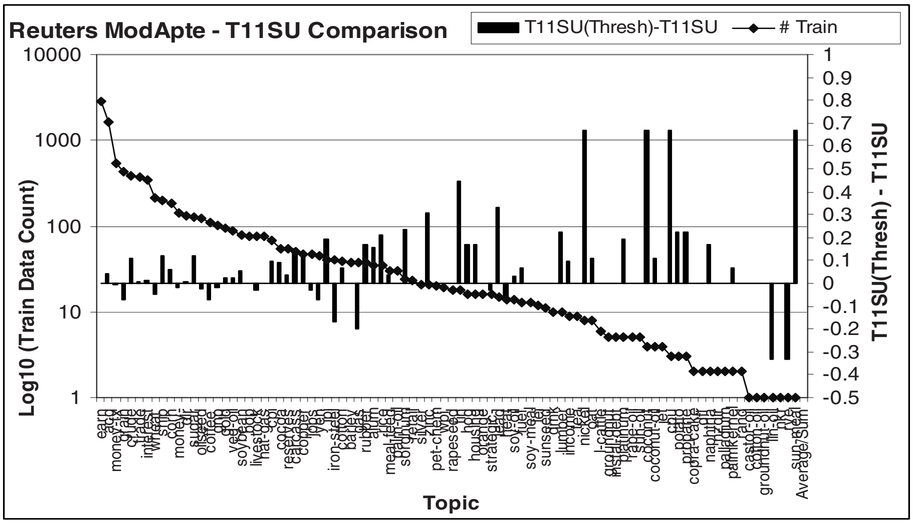

**Fig. 3.** The difference in T11SU performance for the Reuters-21578 corpus.

**Table 3.** A results comparison for the Reuters 87 ModApte corpus.

| Approach | T11SU | $F_{\beta=0.5}$ | Precision | Recall |
|---|---|---|---|---|
| CC Thresholded SVMs | 0.61 | 0.57 | 0.64 | 0.48 |
| Linear SVM | 0.54 | 0.48 | 0.58 | 0.33 |

Overall, we can see that adjusting the threshold using the beta-gamma procedure boosts the performance of the baseline SVM on all examined evaluation measures at a macro level for the Reuter-21578 corpus (Table 3). Examining each topic from a T11SU perspective (Figure 3), we notice that the biggest improvement in T11SU performance comes from topics that have fewer than *fifty* positive training documents, topics that have traditionally being very difficult to model. Overall, 80% of the topics have improved or have not been adversely affected by this procedure.

Figure 4 compares the results for the TREC 2001 corpus between the threshold adjusted SVMs and baseline SVMs for each topic with respect to the T10SU evaluation measure. Table 4 presents the macro-average results for precision, recall, Fbeta, and T11SU for the TREC 2001 corpus. The K-NN result in Table 4 corresponds to a *k* nearest neighbor approach [2]. The IR result is achieved using traditional information retrieval filters [1]. The RBF SVM result in Table 4 was achieved using SVMs and radial basis kernels [13].

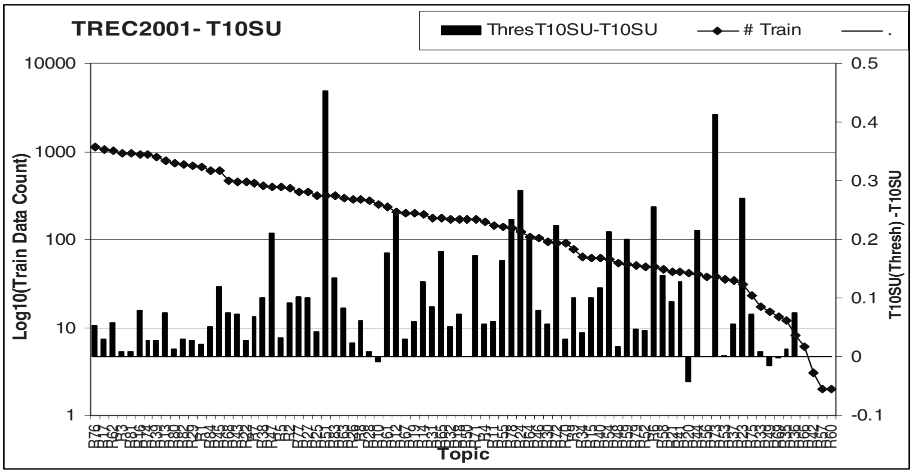

**Fig. 4.** The difference in T10SU performance for the TREC2001 corpus.

**Table 4.** A results comparison for the TREC2001 corpus.

| Approach | T10SU | $F_{\beta=0.5}$ | Precision | Recall |
|---|---|---|---|---|
| Asymmetric SVM [11] | 0.41 | 0.60 | 0.75 | 0.45 |
| CC Thresholded SVMs | 0.40 | 0.56 | 0.64 | 0.50 |
| K-NN [2] | 0.32 | 0.49 | 0.63 | 0.36 |
| Linear SVM | 0.31 | 0.50 | 0.75 | 0.31 |
| IR [1] | 0.31 | 0.51 | 0.57 | 0.41 |
| RBF SVM [13] | 0.28 | 0.46 | 0.55 | 0.44 |

Adjusting the threshold of the SVM for the TREC2001 topics has boosted recall and therefore led to over 20% improvement in terms of T11SU performance over baseline SVMs (linear SVM), while not effecting precision. This performance is comparable with the best performer for this text classification task that was prepared by Lewis [11]. Lewis's submission was generated using asymmetric SVMs. The following observations can be made when we compare evaluation measures for our threshold adjusted experiment and Lewis's asymmetric run: first of all, due to the expensive cross fold validation required for determining the asymmetric costs of the SVM learning, training Lewis's asymmetric SVMs took two orders of magnitude more time to learn than our threshold adjusted SVMs (i.e., 500 hours for Lewis's experiment versus 5 hours for our experiment); Lewis's experiment with asymmetric SVMs provides 14% better precision than our threshold adjusted run; our threshold adjusted run provides 11% better recall than Lewis's run; this would seem to suggest that asymmetric SVMs and adjusting the threshold are addressing two independent aspects of the problem, which if combined could boost performance even further.

## 5 Conclusions

We have presented a novel SVM threshold adjusting algorithm. It uses cross validation to automatically determine the optimal parameters for the beta-gamma algorithm, which are subsequently used to relax the threshold of the class model. The proposed approach boosts the recall performance of baseline SVMs for text classification, while not adversely affecting precision. The gain in performance for examined TREC corpora is over 20% for standard information retrieval measures when compared to baseline SVMs. The extra cost of performing this threshold adjustment is small, in that it is a one-dimensional optimization problem. Adjusting the threshold of SVMs is just one technique for boosting the performance of SVMs. Combining our threshold adjustment algorithm with other techniques, such as asymmetric cost-based learning of SVMs, should lead to even better performance. This is part of ongoing work. A more detailed comparison between the proposed approach and other thresholding approaches that have or can be applied to the task of threshold adjustment for SVMs is currently being carried out. In addition, since the proposed thresholding approach is independent of the learnt model, using it in conjunction with other types of models will also form an interesting aspect of future work.

## Acknowledgements

We thank David Evans, David Hull and Diane Kelly for their useful comments.

## References

1. Arampatzis A., Unbiased S-D Threshold Optimization, Initial Query Degradation, Decay, and Incrementality, for Adaptive Document Filtering, *Tenth Text Retrieval Conference (TREC-2001)*, (2002), 596-605
2. Ault T., Yang Y., kNN, Rocchio and Metrics for Information Filtering at TREC-10, *Tenth Text Retrieval Conference (TREC-2001)*, (2002), 84–93
3. Cancedda N. et al., Kernel Methods for Document Filtering, Eleventh Text Retrieval Conference (TREC-2002), 2003
4. Cortes, C., Vapnik, V., Support vector networks. *Machine Learning,* (1995) 20:273-297
5. Evans, D.A., Shanahan, J., Tong, X., Roma, N., Stoica, E., Sheftel, V., Montgomery, J., Bennett, J., Fujita, S., Grefenstette, G. Topic Specific Optimization and Structuring. *Tenth Text Retrieval Conference (TREC-2001)*, (2002), 132–141

6. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings 10th European Conference on Machine Learning (ECML),* Springer Verlag, (1998)
7. Keerthi, S.S., Shevade, S.K., Bhattacharyya, C., Murthy, K.R.K. *Improvements to Platt's SMO algorithm for SVM classifier design*. Technical report, Dept of CSA, IISc, Bangalore, India, (1999)
8. LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P. and Vapnik, V. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks: The Statistical Mechanics Perspective*, (1995), 261-276
9. Lewis, D.D., Schapire, R.E., Callan, J.P., and Papka, R. Training algorithms for linear text classifiers, In *Int'l ACM Conf. on Research and Development in Information Retrieval* (SIGIR-96), (1996), 298-306.
10. Lewis D. D., The Reuters-21578 text categorization test collection. http://www.research.att.com/~lewis/reuters21578.html. Checked on 11 May 1998; Timestamp Tue Jan 20 21:07:21 EST (1998).
11. Lewis D. D., Applying Support Vector Machines to the TREC-2001 Batch Filtering and Routing Tasks, *Tenth Text Retrieval Conference (TREC-2001)*, (2002), 286-294
12. Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., Kandola, J.S. The Perceptron Algorithm with Uneven Margins. *ICML 2002*: 379–386
13. Mayfield J., McNamee P., Costello C., Piatko C., Banerjee A., JHU/APL at TREC 2001: Experiments in Filtering and in Arabic, Video, and Web Retrieval, at TREC-10, *Tenth Text Retrieval Conference (TREC-2001)*, (2002), 322–332
14. Morik, K., Brockhausen, P., Joachims, T. Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring. *Proc. 16th Int'l Conf. on Machine Learning (ICML-99)*, (1999)
15. Osuna, E., Freund, R., and Girosi, F. Training support vector machines: An application to face detection. In *Proceedings of Computer Vision and Pattern Recognition '97*, (1997), 130-136
16. Platt, J. Fast training of SVMs using sequential minimal optimization. In: B. Scholkopf, C. Burges, and A. Smola (Eds.) *Advances in Kernel Methods – Support Vector Learning*, MIT Press, (1998).
17. Robertson S., Soboroff I., The TREC 2001 Filtering Track Report, *Tenth Text Retrieval Conference (TREC-2001)*, (2002), 26–37.
18. Salton G., Introduction to Modern Information Retrieval, Verlag: Mc Graw Hill , New York, (1983).
19. Vapnik, V., The Nature of Statistical Learning Theory, Springer-Verlag, (1995)
20. Vapnik, V., Statistical Learning Theory, Wiley, (1998)
21. Voorhees E.M., Overview of TREC 2002, Eleventh Text Retrieval Conference (TREC-2002), (2002), 1–16
22. Yang Y., A study on thresholding strategies for text categorization, Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'01), (2001), 137-145
23. Zhai, C., Jansen, P., Stoica, E., Grot, N., Evans, D.A. Threshold Calibration in CLARIT Adaptive Filtering. *Seventh Text Retrieval Conference (TREC-7)*, (1999), 149–156
24. Zhang, Y. and Callan, J. "YFilter at TREC-9". In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, (pp. 135-140). National Institute of Standards and Technology, (2001), 500-249