On the Importance of Parameter Tuning in Text Categorization

Cornelis H.A. Koster
Dept. Comp. Sci., University of Nijmegen, The Netherlands
kees@cs.kun.nl
Jean G. Beney
Dept. Informatique, INSA de Lyon, France
jean.beney@insa-lyon.fr

Abstract. Text Categorization algorithms have a large number of parameters that determine their behaviour, whose effect is not easily predicted objectively or intuitively and may very well depend on the corpus or on the document representation. Their values are usually taken over from previously published results, which may lead to less than optimal accuracy in experimenting on particular corpora.
In this paper we investigate the effect of parameter tuning on the accuracy of two Text Categorization algorithms: the well-known Rocchio algorithm and the lesser-known Winnow. We show that the optimal parameter values for a specific corpus are sometimes very different from those found in literature. We show that the effect of individual parameters is corpus-dependent, and that parameter tuning can greatly improve the accuracy of both Winnow and Rocchio.
We argue that the dependence of the categorization algorithms on experimentally established parameter values makes it hard to compare the outcomes of different experiments and propose the automatic determination of optimal parameters on the train set as a solution.
Keywords: Text Categorization, automatic classification, Winnow, Rocchio, parameter tuning.

1 Introduction

Information Retrieval is an interdisciplinary subject with a long history. Recently, it has been much influenced by theory and methods from Machine Learning (automatic Text Categorization) and Language and Speech technology (Language Modelling, Linguistic Techniques). It has a proud history of strict experimental methodology, exemplified by a sequence of TREC and SIGIR conferences, and a steady progress in its theory.

Text Categorization (for a recent overview see [**?**]) is a perfect area for experimentation in Information Retrieval, because the abundance of documents labeled with categories provides a solution for the otherwise vexing problem of computing Recall.

An automatic Text Categorization algorithm learns from examples, train documents tagged with the categories to which they belong. It determines from

the examples which are characteristic of the categories learned. But these algorithms are to some extent based on formulae that include empirical constants whose value depends on the document set but can not easily be predicted objectively or intuitively, like

- the Term Selection criterion, and the number of terms selected
- the Term Weighting technique (e.g. Boolean, linear, square root, logarithmic, ltc)
- the document length normalization technique
- some parameters typical for the algorithm used (the number of neighbours in Knn, the choice of Kernel Function in SVM, ...), and
- parameters that determine the kind of classification desired (mono- or multi-, the Utility Function to use, the thresholding technique) and others.

These parameters influence to a large extent either the accuracy of the classification algorithm or its speed, or both. When comparing various classification algorithms on the same task, the differences in performance found may well be attributable to a large extent to differences in tuning, rather than to inherent qualities of the algorithms, falsifying the experiment (see also [**?**].

In principle, given enough train documents the optimal choice of a technique or parameter value can be computed from the train set by brute force, but this is time-consuming and difficult to repeat for every new train set. Therefore researchers tend to reuse choices and parameter values which have been reported in literature.

The ideal Text Categorization system should have only parameters whose effect is intuitively clear and predictable, and as few of them as possible. Otherwise the user of the system might well feel lost in a high-dimensional parameter space (like the Nuclear Physicist of the late sixties, trying to manually fit the Optical Model to his data by twiddling 16 parameters).

In this note we'll investigate the typical parameters of the Rocchio and Winnow Text Categorization algorithms and their interaction with Term Selection and propose a technique to make these algorithms self-tuning.

2 Experimental setup

In our investigations we made use of the LCS classification engine, which implements the Winnow and Rocchio algorithms (see the next sections), automatically learns class thresholds from the train data and has a choice of Term Selection algorithms [**?**].

Our experimental approach is as follows: we shall first tune each algorithm on a variety of corpora, i.e. determine the optimal values of their parameters, without performing any Term Selection. Then we shall determine the effect of optimal Term Selection on the same corpora with and without tuning, and interpret the results.

Our experimental approach is as follows: we shall tune each algorithm on a variety of corpora, and determine optimal values for their parameters on each corpus, maximizing the accuracy (measured by the micro-averaged F1 value).

2.1 The corpora

We have experimented with many different corpora, exemplifying different document representations and classification tasks, in order to make the results more general. In this publication we shall use three:

- ModApte – The well-known Apté subset of the Reuters 21578 corpus, consisting of 12902 newspaper stories in 135 TOPICS categories, with an average length of 116 words [**?**]
- The EPO1A corpus consists of 16000 abstracts of patent applications in English from the European Patent Office, with an average length of 143 words (see [**?**,**?**]). For this corpus, we shall also show results using a bag-of-phrases representation (EPO1Afr, see [**?**]) besides the customary bag-of-words representation (EPO1A kw).
- EPO1F – These are the full-text patent applications corresponding to the EPO1A abstracts, of about 2000 words each; The total collection has a size of 4611 M-bytes.

All these corpora are in English, but they differ in other properties, as shown in table **??**, representing different classification tasks.

Corpus	doc size	classes	nmb docs/class
ModApte corpus	short	135 multi	widely varying
EPO1A abstracts	short	16 mono	1000 documents
EPO1F full-text	long	16 mono	1000 documents

Table 1. Some quantitative differences between the corpora

The ModApte corpus has been used (in slightly different subsets) in many experiments reported in literature, allowing comparison with other work.

2.2 The experiments

In each experiment, the corpus used was split into 4 subsets of equal size, chosen at random, in a four-fold cross-validation (training on one subset while using the union of the other three as test set). In each run, 25% of the documents were used as train documents and 75% as test documents. The relatively large number of test documents was chosen in order to reduce variance (and because testing is much faster than training). As a Measure of Accuracy we used the micro-averaged F1 value. The train sets were kept small, since the goal was to make many comparisons, rather than to achieve the highest possible accuracy. In testing we made no use of the information that the EPO1A/F corpora are mono-classified, allowing 0-3 classifications per document, and 0-16 for the ModApte corpus, so that the results are also applicable to multi-classification. We used the

same term weighting (ltc) and the same document length normalisation (cosine) in all experiments.

For the keyword representation, no pre-processing was applied, apart from de-capitalization and the removal of some special characters. For the phrase representation, the EPO1A corpus was parsed and translated to Head/Modifier pairs and unnested as in [**?**].

In the graphs we shall usually not indicate the variance, because this would make the graphs illegible, but the amount of variance in the measurement results is mostly obvious from the irregularity in the graphs.

3 Tuning Rocchio

In origin [**?**], the Rocchio algorithm was conceived for retrieval with relevance feedback [**?**]

$$Q_{new} = \alpha \times Q_{orig} + \beta \times \frac{1}{R} \sum_{D \in Rel} D - \gamma \times \frac{1}{N-R} \sum_{D \notin Rel} D$$

in which each document is represented by a vector of term frequencies. In the Rocchio classification algorithm a class profile is computed as the centroid of the documents relevant to the class, subtracting the irrelevant documents (there is no original document).

A more sophisticated form of the Rocchio algorithm [**?**] assigns to each individual term t a weight for the class c, according to the formula

$$w(t,c) = max(0, \frac{\beta}{|D_c|} \sum_{d \in D_c} s(t,d) - \frac{\gamma}{|\overline{D}_c|} \sum_{d \in \overline{D}_c} s(t,d))$$

where

- $s(t,d)$ is the normalized strength of the term t in the document d, using some sub-linear function of the frequency of the term and compensating in some way for variations in document length
- D_c is the set of documents that are labeled with class c and $\overline{D}_c$ the set of non-c documents.

The score of a document for a class is the inproduct of the weights of its terms times their strength, and a document is assigned to class c if its score for c exceeds a class threshold which is computed from the train set.

Notice also that terms that would yield a negative contribution are eliminated in the above formula. This contradicts the original intuition of the centroid computation. We shall therefore investigate both variants.

3.1 Choice of Beta and Gamma

The Rocchio formula contains two parameters β and γ, whose values are well-known to be 16 and 4, respectively (according to many publications, including

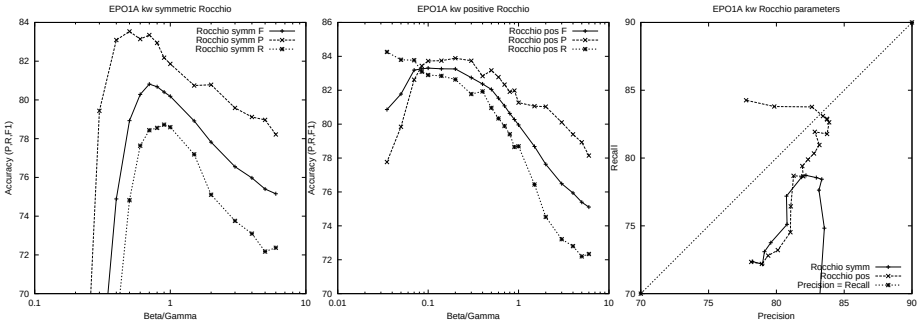

Fig. 1. Varying Rocchio's beta parameter

very recent ones like [**?**,**?**]). Why 16 and 4? The reason for these curious values is that there was originally also a factor α, which is zero in the present formula.

Obviously, dividing both parameters by the same factor makes no difference, apart from a shift of the threshold, so we can fix γ at 1, leaving only one parameter, which can be interpreted as the relative weight attributed to positive examples. The standard value of β is 4; but is this optimal?

3.2 Symmetric Rocchio and Positive Rocchio

We shall now try to find optimal values of β for various corpora, without Term Selection, i.e. without discarding any of the terms, keeping $\gamma = 1$ and distinguishing between a variant in which negative term weights are allowed ("symmetric Rocchio") and one in which negative term weights are omitted ("positive Rocchio"). Figure **??** shows the the result of varying β in classifying the EPO1A corpus (without Term Selection).

The left graph shows that the optimum β-value for symmetric Rocchio is not 4 but 0.7; the middle one shows that positive Rocchio reaches an even higher Accuracy at $\beta = 0.1$. The elimination of negative terms pays off. The rightmost graph (which is hard to interpret) shows the trajectory followed by Precision and Recall when reducing the β-value, for both variants. When β becomes too small, positive Rocchio fails catastrophically through loss of Recall and symmetric Rocchio through loss of Precision.

Figure **??** compares the accuracy achieved by the two Rocchio variants at various β-values for the EPO1A corpus and the ModApte and EPO1F corpora.

They show roughly the same behaviour, and practically the same optimal parameter values. In all cases, positive Rocchio performs better than symmetric Rocchio, with an optimum near $\beta = 0.1$.

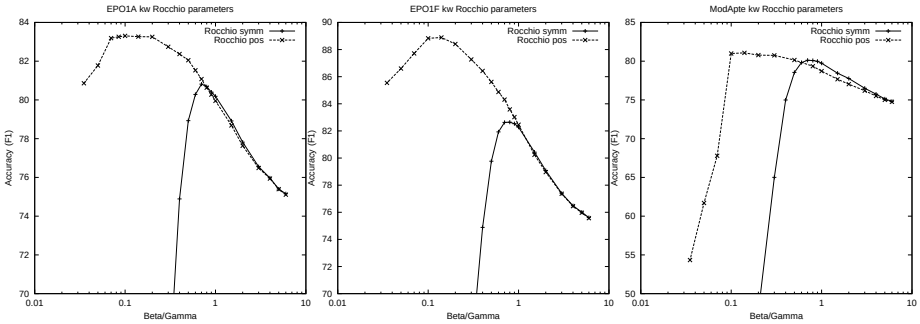

Fig. 2. Comparing two Rocchio variants on 3 corpora

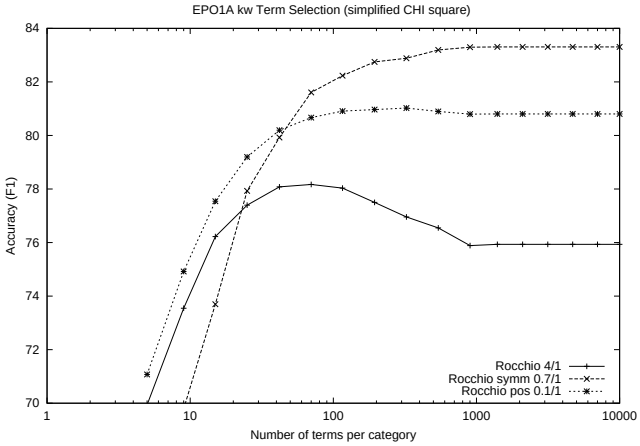

Fig. 3. Tuning versus Term Selection for Rocchio

3.3 Interaction with Term Selection

Next, we investigate the interaction between parameter tuning and term selection, by showing the Accuracy on EPO1A as a function of the number of terms per class, selected by the Simplified χ^2 criterion. The results are shown in fig. **??** for both versions of Rocchio at optimal tuning and only one version at standard tuning (where they are indistinguishable).

As was found in our earlier experiments, (cf. [**?**]), at standard parameter values Rocchio reaches its optimal Accuracy when 100 terms per class are selected. After tuning, the Accuracy (which is much higher) is no longer improved by Term Selection, although selecting at most 1000 terms/category may still be useful for performance reasons. An optimal choice of β/γ has much more effect than optimal Term Selection. At optimal parameter values, Rocchio itself eliminates the noisy terms.

3.4 Discussion

The positive Rocchio variant in each case reaches a higher accuracy than the symmetric one, at a much lower β value. It is surprising to see that the β parameter must be smaller (or even much smaller) than the γ parameter, so that a greater weight is given to negative examples, even though there are many more of them than positive examples. A plausible explanation of the behaviour of positive Rocchio is that, as β gets smaller, more noisy terms are eliminated (because their weight becomes negative) and the Accuracy is improved, until eventually too many terms are eliminated to achieve Recall.

Similarly, for symmetric Rocchio the effect of giving more weight to negative examples is initially beneficial, but when β gets too small, documents tend to get classified on the non-occurrence of negative terms, rather than on the occurrence of positive ones (keep in mind that the threshold will shift automatically with the sinking document scores).

At the traditional value $\beta = 4$ the contribution of negative terms is very small and the two Rocchio variants behave indistinguishably. This is probably the reason that we found only one article in literature [**?**] which describes and explains the superiority of the positive Rocchio variant.

4 Tuning Winnow

The Balanced Winnow algorithm is a child of the Perceptron, as is clear from the formulation given in [**?**]:

BalancedWinnow$(\overrightarrow{w}, \overrightarrow{z}, (\overrightarrow{x}, y))$:
if sign $(\overrightarrow{w} \cdot \overrightarrow{x}) \neq y$ then
begin $\overrightarrow{z} := \overrightarrow{z} + \alpha y \overrightarrow{x}$
$\overrightarrow{w} := 2\overrightarrow{sinh}(z)$
end

where $y = 1$ for a relevant document and $y = -1$ for an irrelevant one, and the weights are exponentiated by the function $sinh$.

In the description of Balanced Winnow given in [**?**], for every class c and for every term t two weights W_t^+ and W_t^- are kept. The single parameter α of Winnow has been split into a promotion parameter α and a demotion parameter β.

The score of a document d for a class c is computed as

$$SCORE(d, c) = \sum_{t \in d} (W_{t,c}^+ - W_{t,c}^-) \times s(t, d)$$

where $s(t, d)$ is the normalized strength of the term t in d. A document d belongs to a class c if $SCORE(d, c) > \theta$, where the threshold θ is usually taken to be 1.

Winnow learns multiplicatively, driven by mistakes, one document at a time: When a train document belonging to some class c scores below θ, the weights of its terms t in W_t^+ are multiplied by a constant $\alpha > 1$ and those in W_t^- multiplied by $\beta < 1$; and conversely for documents *not* belonging to c which score above θ.

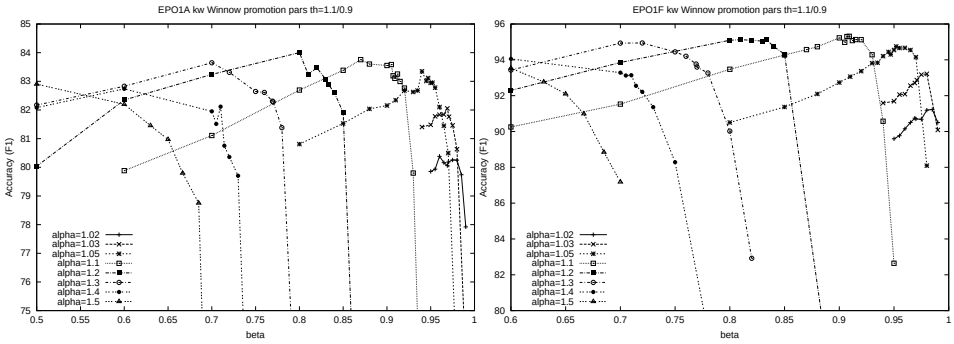

Fig. 4. Varying Winnow's promotion parameters

Winnow is by origin an *on-line* algorithm, meaning that the weights are adjusted for each incoming train document, but it is also possible to iterate over a set of train documents. In the following experiments, we shall at first keep the number of iterations at 5, and later study the effect of different numbers of iterations.

4.1 The Winnow Promotion parameters

According to [**?**] β should be equal to $1/\alpha$. The values suggested in [**?**] are 1.1 and 0.9, whose product is in fact not quite equal to one.

There are good reasons to choose β smaller than $1/\alpha$: If β is smaller than $1/\alpha$, Winnow learns faster from positive examples than from negative examples, which may be justified since there are fewer positive examples. Furthermore, consider a term that is promoted and demoted a large but equal number of times. When $\alpha \times \beta < 1$, both W_t^+ and W_t^- will tend to zero: this noisy term is eliminated. When $\alpha \times \beta = 1$, they keep their initial values. One would expect a smaller value of α to lead to slower but more precise convergence.

Measuring the Accuracy as a function of α and β for EPO1A kw and EPO1F gives the results shown in fig. **??**.

In spite of the four-fold cross-evaluation, there is a lot of variance (especially for EPO1A kw), which makes it hard to choose an optimal value. From these graphs (and many others not shown here), it appears that $\beta = 2 - \alpha$ is a better choice than $1/\alpha$. The optimal choice of α depends on the number of training documents. For larger numbers of docs it is better to choose a smaller alpha: On the EPO2F corpus (which is like the EPO1F corpus but with 68418 instead of 16000 train documents) we found the highest accuracy at $\alpha = 1.04$ when training on all train documents and at $\alpha = 1.1$ when training on a tenth of the documents [**?**].

In the following experiments we will stick to Dagan's choice [**?**] (1.1/0.9) for the promotion parameters, which is quite good for all three corpora, and tune the other parameters accordingly.

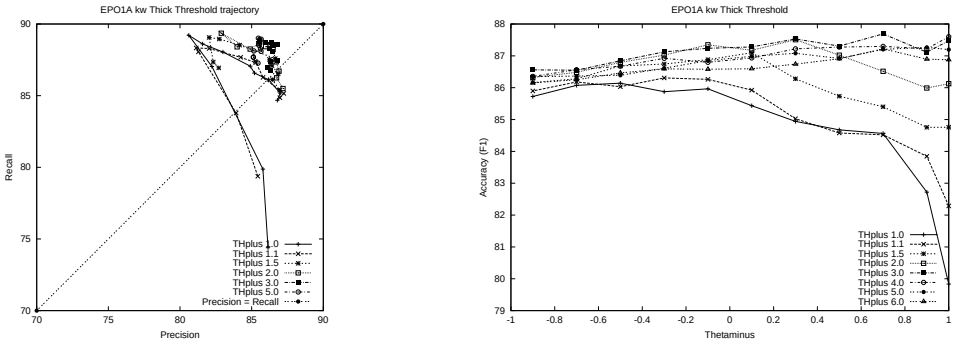

Fig. 5. Effect of Thick Threshold Heuristic (1)

4.2 The Thick Threshold heuristic

Again following [**?**], the Accuracy of Winnow can be improved by means of the *thick threshold* heuristic: In training, we try to force the score of relevant documents up above $\theta^+ > 1.0$ (rather than just 1) and irrelevant documents below $\theta^- < 1.0$. This resembles the "query zoning" [**?**] heuristic for Rocchio, in the sense that documents on the borderline of a class (scoring between θ^- and θ^+) receive extra attention.

According to [**?**] the optimal values for these Thick Threshold parameters are 1.1 and 0.9, respectively (just like the α and β). A heuristical argument suggests that the best value for θ^- might be $1/\theta^+$, since they each represent a number of steps (multiplications by α or β) away from the threshold 1. Figure **??** shows the effect of varying the thickness of the threshold for the EPO1A corpus.

The left graph in figure **??** plots Precision against Recall. Each line is a trajectory (going approximately first upwards and then to the right) which represents one value of θ^+ together with values of θ^- going down from 0.9 to 0.1. The dotted line represents Precision = Recall.

The right graph is easier to read. The F1-value fluctuates wildly, but by and by an increase of θ^+ improves the Accuracy, and 3.0/0.7 raises the F1-value over 1.1/0.9 by more than 2 points.

The Thick Threshold graphs in fig. **??** show roughly similar behaviour on the different corpora, but it is clear that the optimal values for the Thick Threshold parameters depend strongly on the corpus. When increasing θ^+, the Accuracy first increases, then falls off again. The curves are rather erratic due to high variance. The intuition that the best value for θ^- is $1/\theta^+$ is not supported by the measurements.

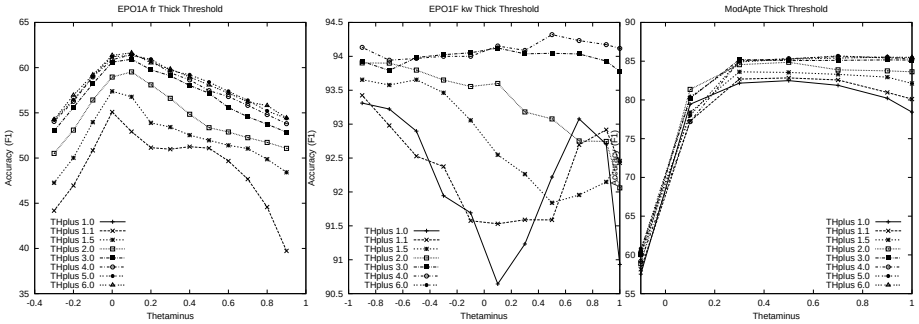

Fig. 6. Effect of Thick Threshold Heuristic (2)

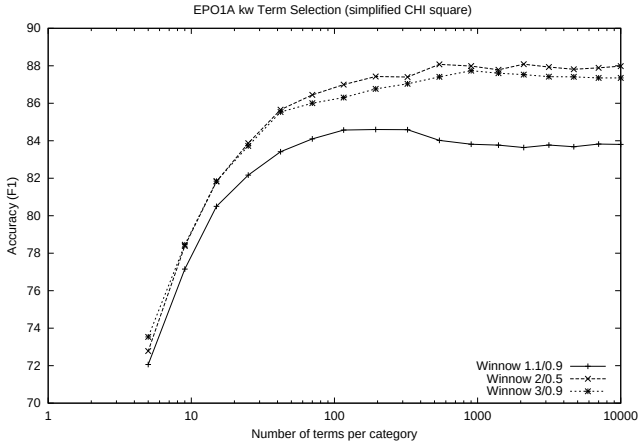

Fig. 7. Tuning versus Term Selection for Winnow

4.3 Interaction with Term Selection

The following Figure **??** shows the effect of Term Selection with and without tuning of Winnow. Again, it appears that optimal parameter tuning removes the noisy terms much more effectively than Term Selection.

The relation between Winnow training and Term Selection is quite obvious: given enough promotions and/or demotions of documents belonging to some class, all terms occurring in the profile for that class will reach one of the following states:

- **positive terms** – terms speaking for the category will obtain a relatively stable positive value for W^+ and W^- will be (practically) zero
- **negative terms** – terms speaking against the category will have a positive W^- while W^+ is zero
- **noisy terms** – terms that are not reliable will have both W^+ and W^- (practically) zero.

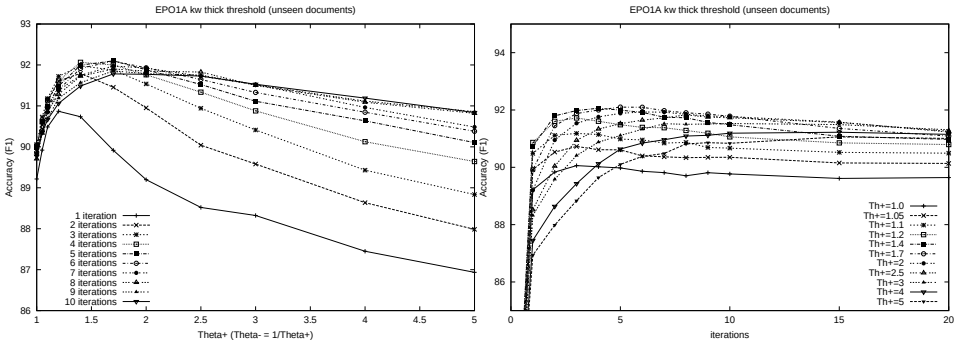

Fig. 8. Iterations and Thresholds

Experiments not reported here show that omitting the negative terms (positive Winnow) causes a significant reduction in the Accuracy. Symmetric Winnow is in all cases better than positive Winnow, in contrast with Rocchio, which appears to cope well enough with noisy terms, but can not use the information provided by negative terms. This may explain why, in our experiments, Winnow always outperforms Rocchio.

4.4 The number of iterations

In our previous experiments with Winnow we fixed the number of iterations at five: every document (in random order) was seen five times by the classification algorithm, and only in case the classification computed for it was not correct, the document was trained.

Training more than once may increase Accuracy, especially when there are few documents, but training too often may lead to overtraining. The optimal number of iterations is dependent on the corpus, and on other parameters. We shall now investigate experimentally the interaction between the number of iterations and the optimal Thick Threshold values.

Since it is hard to make an understandable graph depicting the relationship between the three parameters, we fix $\theta^- = 1/\theta^+$ ("geometrical symmetry"), even though according to figures 4 and 5 this is not optimal.

The two graphs in fig. **??** show the Accuracy achieved on EPO1A kw as a function of θ^+ and the number of iterations, respectively.

The left graph shows that the Accuracy increases with θ^+ at a suitable number of iterations, and then slowly goes down. The right graph shows that, for a given value of θ^+, the Accuracy increases with the number of iterations, until it is reduced again by overtraining. The overtraining point increases with the thickness of the threshold. It appears that choosing 3 to 5 iterations is best, at appropriate Theta values. The other corpora show similar behaviour.

Precision/Recall/F-value graphs for θ^+ against θ^- (like those shown in figure **??** for ModApte) show rich details for which no explanatory theory is available.

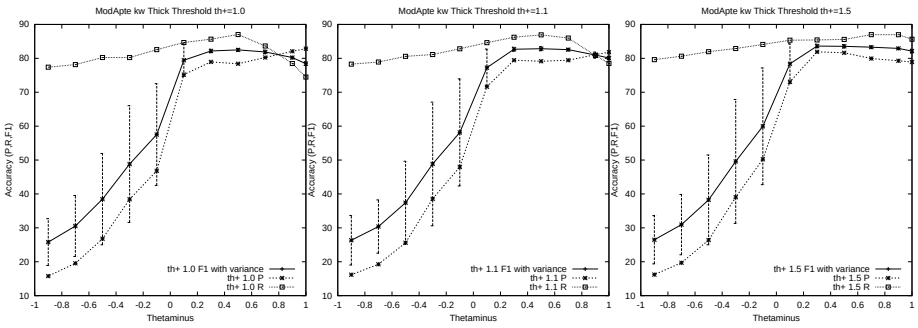

Fig. 9. Thick Threshold - details

Notice that the Precision = Recall point is not the optimum, and for $\theta^- < 0$ the severe drop in Precision and the large increase in variance.

5 Discussion

We have experimentally shown the importance of properly tuning the parameters of the Rocchio and Winnow classification algorithms. The optimal parameter values found in our experiments were mostly quite different from those found in literature, and gave a much higher accuracy. The good news is that in many existing applications the accuracy can be improved with just a little effort. The bad news is that it is hard to make meaningful comparisons between different classification algorithms. There is an unvoluntary bias against algorithms not familiar to the experimenter, caused by the difficulty of choosing parameters for an unfamiliar algorithm.

In comparing algorithms, the relevant parameter settings should also be published. And the comparison can not be fair unless all algorithms have been carefully tuned on the train set in order to capture the corpus-dependent aspects.

Since the manual tuning process is difficult and time consuming, every practical classification system should be made self-tuning. The best way to deal consistently with all those parameters is to tune them automatically, rather than using default values, for every new corpus or application.

Judging by the graphs in the paper (and the many thousand others we have seen in other experiments) the Accuracy (F1-value) of Winnow on a given corpus is a concave function of the parameters with a large noise term added (2 to 3 percent of the Accuracy). We conjecture that use of the Swarm Optimization technique (based on [**?**]) might lead to an efficient optimization scheme.

6 Conclusion

We have shed some light on the effect of the most important parameters of these algorithms, and on some of the interactions between them: Rocchio's weighting

parameters, Winnow's promotion parameters and Thick Threshold, and their interaction with (one form of) Term Selection. Unfortunately these effects are quite dependent on the corpus. Only when using optimal parameter values, tuned for the particular corpus, can we claim that the results are really representative for the algorithm, and can we stop worrying about parameters.

For Rocchio it was found that discarding all terms with a negative score contribution (positive Rocchio) leads to a better Accuracy than taking them into account (the symmetric Rocchio algorithm). We have argued that this is an effect of automatic Term Selection.

We found the optimal values given for the Rocchio and Winnow parameters in many publications to be wildly wrong, and conjecture that many published results comparing these with other algorithms are misleading.

At optimal parameter values, both Rocchio and Winnow show no positive effect of Term Selection on the Accuracy; at best, performance may be somewhat improved. When optimally tuned, these algorithms are well capable of selecting their own terms. Tuning is much more important than term selection.

But many questions remain open: What happens for other forms of Term Selection? What happens when we use the macro- instead of micro-average (emphasizing the accuracy on small categories)? What is the influence of the Term Weighting and Document Size Normalization technique used? Are the optimal parameter values the same for larger and smaller subsets of the corpora? In fact, we are still groping around in high-dimensional space. More theoretical analysis (in particular of the fascinating Winnow algorithm) is needed.

References

[Apté and Damerau,1994] Apté, C. and Damerau, F. (1994) Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems* 12,3,233-251.

[Beney and Koster, 2003] J.G.Beney and C.H.A. Koster (2003), Classification supervisée de brevets: d'un jeu d'essai au cas réel. Proceedings of the XXIeme congre's Inforsid, pp.50-59. http://www.loria.fr/conferences/inforsid2003/ActesWorkshopRI.pdf (last visited October 2003).

[Caropreso et al, 2000] M.F. Caropreso, S. Matwin and F. Sebastiani (2000), A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization, In: A. G. Chin (Ed.), *Text Databases and Document Management: Theory and Practice*, Idea Group Publishing, Hershey, US, pp. 78–102.

[Cohen and Singer, 1999] W.W. Cohen and Y. Singer (1999), Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems* 13,1,100-111.

[Crammer and Singer, 2002] K. Crammer and Y. Singer, A New Family of Online Algorithms for Category Ranking, Proceedings SIGIR '02, pg. 154

[Daelemans et al, 2003] Walter Daelemans, Vronique Hoste, Fien De Meulder and Bart Naudts, Combined Optimization of Feature Selection and Algorithm Parameter Interaction in Machine Learning of Language. In: Proceedings of the 14th

European Conference on Machine Learning (ECML-2003), Cavtat-Dubrovnik, Croatia, pp. 84-95, 2003.

[Dagan et al, 1997] I. Dagan, Y. Karov, D. Roth (1997), Mistake-Driven Learning in Text Categorization. In: *Proceedings of the Second Conference on Empirical Methods in NLP*, pp. 55-63.

[Grove et al, 2001] A. Grove, N. Littlestone, and D. Schuurmans (2001), General convergence results for linear discriminant updates. Machine Learning 43(3), pp. 173-210.

[Hiemstra, 1998] D. Hiemstra (1998), A Linguistically Motivated Probabilistic Model of Information Retrieval, European Conference on Digital Libraries 1998, pp. 569-584.

[Koster et al, 2003] C.H.A. Koster, M. Seutter and J. Beney (2003), Multi-classification of Patent Applications with Winnow, Proceedings PSI 2003, Springer LNCS 2890, pp 545-554.

[Koster and Seutter, 2003] C.H.A. Koster and M. Seutter (2003), Taming Wild Phrases, Proceedings 25th European Conference on IR Research, Springer LNCS 2633, pp 161-176.

[Krier and Zaccà, 2001] M. Krier and F. Zaccà (2002), Automatic Categorisation Applications at the European Patent Office, World Patent Information 24, pp. 187-196.

[Moschitti, 2003] A. Moschitti (2003), A Study on Optimal Parameter Tuning for Rocchio Text Classifier, Proceedings 25th European Conference on IR Research, Springer LNCS 2633, pp 420-435.

[Peters and Koster, 2002] C. Peters and C.H.A. Koster, Uncertainty-based Noise Reduction and Term Selection in Text Categorization (2002), Proceedings 24th BCS-IRSG European Colloquium on IR Research, Springer LNCS 2291, pp 248-267.

[Rocchio, 1971] Rocchio, J.J. (1971). Relevance feedback information retrieval. In: Salton, G. (ed.) *The smart retrieval system—experiments in automatic document processing* p. 313-323. Prentice-Hall, Englewood Cliffs, NJ.

[Sebastiani, 2002] F. Sebastiani (2002), Machine learning in automated text categorization, ACM Computing Surveys, Vol 34 no 1, 2002, pp. 1-47.

[Singhal et al, 1997] A. Singhal, M. Mitra and C. Buckley (1997). Learning Routing Queries in a Query Zone. Proceedings SIGIR '97, pp. 25-32.