Improving Rocchio with weakly supervised clustering

Romain Vinot and François Yvon

GET/ENST, 46 rue Barrault,
75634 Paris Cedex, France
{romain.vinot, francois.yvon}@enst.fr

Abstract. This paper presents a novel approach for adapting the complexity of a text categorization system to the difficulty of the task. In this study, we adapt a simple text classifier (Rocchio), using weakly supervised clustering techniques. The idea is to identify sub-topics of the original classes which can help improve the categorization process. To this end, we propose several clustering algorithms, and report results of various evaluations on standard benchmark corpora such as the Newsgroups corpus.

1 Introduction

The automated categorization of documents [16] into predefined categories has progressively emerged as one of the most popular task in the area of Text Mining technologies. The categorization paradigm provides a very general framework for many practical applications such as filtering, routing, indexation, tracking... Historically, research in automated text categorization has been promoted by the Information Retrieval community, in the context of automated text indexation. In the past five to ten years, this task has been rediscovered by the Machine Learning community and has since been the subject of many empirical studies [20], demonstrating the applicability and efficiency of Machine Learning algorithms such as Support Vector Machines [6] and Boosting techniques [14].

The Rocchio algorithm [12], originally proposed as a means to improve information retrieval (IR) systems, is conceptually one of the simplest text categorization algorithm. As such, it has been shown to be, in many experimental conditions, less successful than other approaches, such as SVMs or k-NN. Various improvements of this algorithm have been proposed, eg. in [17] [1] [15] or [9], which have been successfull in effectively increasing the performance of this methodology.

As discussed for instance in [19], Rocchio is especially well suited for applications where (i) the number of classes is high; (ii) the n-best answers (rather than just the first one) are taken into account and (iii) class labels are noisy. This seems to happen in practical applications, especially when categories cannot be directly linked with the thematic content of documents. Conversely, we have experimentally demonstrated that classes with heterogeneous content can

badly impair its performance. This behaviour is consistent with the simplicity of the underlying statistical model and of the learning procedure.

In this paper, we investigate several procedures allowing Rocchio to automatically adapt its complexity to the data by the use of a weakly supervised clustering. The idea is to identify *useful* subclasses in the training data, and to use these to refine the decision surface; each new subclass increasing the overall model complexity. The main novelty of this work lies in our attempt to discover these subclasses in such a way that the resulting partition actually improves the resulting decision rule.

The idea of using unsupervised clustering to improve supervised classification is not entirely new, and has already been suggested in several contexts. In the context of the k-nearest neighbors algorithm, many authors have advocated unsupervised clustering as a means to organise very large instance sets, thus speeding up the k-nn computation.

In the context of the (TREC) batch filtering task, [4] and [11] incrementally cluster the incoming data into subclasses of the original classes. These clusters are then used as (new) regular class label, and serve to classify new data: any document falling into cluster S of class C is eventually labelled as C. While both papers use rather different classification algorithms (Rocchio for the former paper, and SVM for the latter), both report quite unconclusive results. In any case, this optimisation does not seem to increase overall performance. One should note that in this approach, clustering and classification are viewed as two separate and unrelated processes.

[3]'s motivations for using unsupervised clustering are clearer: these authors explicitely aim at structuring an heterogeneous corpus of training texts, in the context of a filtering task. Relevant texts are first partitionned into N clusters, using *autoclass* [2]. Test documents are then classified as follows: for each subclass a relevance judgement is separately computed; these judgements are then linearly combined, using a perceptron. This procedure seems to provide the authors an effective means to isolate those clusters which provide the most relevant judgements.

[7]'s idea is to use clustering techniques in order to make the k-nn decision rule less sensible to noisy data, and more like a linear classifier. Proceeding bottom-up, their algorithm recursively aggregates training instances subject to the condition that (i) they belong to the same class and (ii) they are sufficiently close in the representational space. Test documents are then classified according to the following two-steps procedure: first compute the similarity with each cluster using Rocchio or the Widrow-Hoff rule; then linearly combine these similarities to compute the final label. This procedure provides a significant improvement both over a "pure" k-nn approach and a "pure" linear decision.

This paper is organised as follows: in Section 2 we present the Rocchio algorithm and discuss its main advantages and drawbacks. Section 3 explains how this baseline is improved with a weakly supervised clustering procedure. We describe two algorithms which aim at creating those clusters that might prove beneficial for the classification procedure. Section 4 reports and discusses exper-

imental results obtained on 3 different textual databases and Section 5 presents some conclusions and directions for future work.

2 Rocchio

Rocchio is a text classifier originally introduced in [12] to improve information retrieval systems with relevance feedback. It uses the vector space model [13]: every text d is represented by a vector $[d]$ in R^n (with n the number of distinct word in the corpus), each coordinate d_w can be computed from the frequency $\mathrm{fr}(w,d)$ of word w in d in several manners. One of the most used formulation is :

$$d_w = TFIDF(w,d) = \log(1+\mathrm{fr}(w,d)) * \log(\frac{N}{N(w)}) \quad (1)$$

where N is the number of documents in the corpus and $N(w)$ the number of documents in which w occurs at least once. This measure emphasizes those words which simultaneously occur frequently in the document while occuring only in a few documents. Each vector $[d]$ is then normalized according to: $\underline{d_w} = \frac{d_w}{\sqrt{\sum_w d_w^2}}$ in $[\underline{d}]$ so as to lessen the weight of longer documents. A prototypical profile $[c]$ is computed for each class c according to :

$$c_w = \frac{t}{N_c}\sum_{d\in c}\underline{d_w} - \frac{1-t}{N_{\overline{c}}}\sum_{d\notin c}\underline{d_w} \quad (2)$$

with N_c the number of documents in c, $N_{\overline{c}}$ the number of documents not in c and t a free parameter between 0 and 1 . These profiles are defined as the centroid of the examples (with a positive coefficient for examples in the class and a negative one for the others). These vectors are also normalized. In the context of a routing task, one usually chooses $t = 1$ (in this case, negative examples do not contribute to the centroids).

Classification of new documents are performed by computing euclidian distance between the document vector and the prototype vector of each class; the document is then assigned to the nearest class. As all vectors (prototypes and documents) are normalized, euclidian distance is equivalent to cosine similarity or dot product which is then used for implementation reasons.

In the context of this paper, Rocchio exhibits two important characteristics :

- The decision rule computed by Rocchio is a linear separator (hyperplane) in the vector space. It gives Rocchio the same expressiveness as a Perceptron classifier. As [8] shows on filtering tasks, accuracy of Rocchio with dynamic feedback is comparable to that of a neural network trained with a gradient descent algorithm.
- The learning model of Rocchio is a generative one. Parameters are optimized to match the data, not to discriminate the different classes. Generative models usually have lower asymptotic performance value than discriminative ones, but they converge to their optimal performance faster, especially when

there is lots of parameters to estimate (which is the case in the textual classification domain) [10].

We have shown in [19] that Rocchio is robust in the presence of noise and very effective for routing task with a high number of classes. On the other hand, its inability to take into account the substructure of classes (such as different subtopics) is a significant drawback compared to others algorithms. Rocchio vastly underperforms other algorithms when classes contain intermixed subtopics: when some subtopics of different classes are nearer than subtopics of the same class (for example in the case of spam filtering, the subtopic on software advertisement can be closer to legitimate emails than to other subtopics of spams such as ones with pornographic content). In the remainder of this paper, we will call classes with such intermixed subtopics "heterogeneous classes". Figure 1 shows an example of homogeneous and heterogeneous classes in terms of relative positioning of subtopics in the vector space.

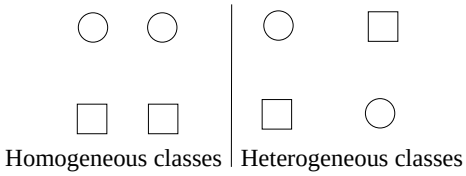

Fig. 1. Homogeneous and heterogeneous corpus (round and square classes have two subtopics).

These difficulties in dealing with classes with intermixed subtopics come from the generative model used by Rocchio. It assumes that each class has a spherical shape and that the information of the centroid is enough to describe correctly a class. Training is accordingly straightforward, because one only has to compute the centroid of all examples for each class. But in cases where this model doesn't suit the corpus, Rocchio will perform poorly.

To avoid these shortcomings, one need to allow Rocchio to use more complex models while at the same time preserving its simple learning process. This can be achieved with the use of a clustering algorithm which will split the classes into coherent sub-classes. A prototype is then computed for each cluster, as if it was a class in its own right. New examples are labelled according to the class of the nearest prototype. We call this class of algorithms Multi-Prototypes Rocchio (MPR). Even if this procedure can be used with any classifier, it is especially tailored to avoid Rocchio's shortcomings.

Rocchio can be seen as a neural network with no hidden layer. Weights are not learned by a back-propagation algorithm but simply computed as the mean of weights of all examples. Clusters are analog to a hidden layer having as many neurons as clusters. Weights of connection between initial layer and hidden layer are still computed by the mean of examples, weights between the hidden layer and the final layer are binary values according to the class of the cluster. Propagation

of weights are different in MPR, because classification is based on the nearest prototype (the hidden neuron with the highest weight) instead of a weighted sum of all hidden layers. With this analogy, we see that the clustering allows to change the complexity of the underlying model.

Moreover, preliminary experiments [19] show that the use of subclasses can be very useful for heterogeneous classes but not for homogeneous ones. The model complexity (in other words number of clusters) must not be chosen according to the data but driven by the accuracy of Rocchio. There is no need to separate two subtopics of the same class if they are not mixed with any subtopic of another class. This suggests that clustering must be performed using a discriminative approach rather than a generative one.

Most of the clustering algorithms described in introduction don't meet these requirements, because the clustering process is independent of the categorization one. They don't try to directly optimize the accuracy of the induced classifier (with the exception of the GIS algorithm from Lam).

3 Clustering algorithms

We want to devise a clustering algorithm which detects clusters in heterogeneous classes but not those in homogeneous classes. This condition can be translated as a mathematical criterion in various ways. We have explored two different criteria: the first one is based on the relative positioning of clusters, the second one is more directly error-driven and based on example categorization. Both are integrated into a top-down hierarchical clustering (at each step, we split one cluster into two new smaller clusters). Because the criteria use the class labels of examples, this procedure is called weakly supervised.

3.1 Notations

Let $\mathcal{C} = \{c_1, ..., c_k\}$ be a set of k classes, N the number of examples and $\Pi = \{\Pi_1, ..., \Pi_p\}$ the unknown partition of examples. $c(x)$ is the class of example x and $c(\Pi_i)$ the class labels of the examples in Π_i. Clusters are only allowed to split the existing classes. The partition must then verify:

$$\forall (x, y)\ \Pi(x) = \Pi(y) \Rightarrow c(x) = c(y) \tag{3}$$

3.2 Top-down clustering

Our algorithm starts with one cluster per class. At each step, we want to split the cluster with the largest dispersion of its documents. As explained in [5], the norm is the square-root of the average pairwise similarity between the documents:

$$\begin{aligned} ||c||^2 &= \sum_w c_w^2 = \frac{1}{N_c^2} \sum_w \left(\sum_{d \in c} d_w \right)^2 \\ &= \frac{1}{N_c^2} \sum_w \left(\sum_{d1, d2 \in c} d1_w d2_w \right) \\ &= \frac{1}{N_c^2} \sum_{d1, d2 \in c} [d1][d2] \end{aligned} \tag{4}$$

So we choose the cluster with the smallest norm and apply any clustering algorithm to create two subclusters. We choose K-Means for its simplicity. The split is tested according to the wanted criterion. If the division is not accepted, we try with the cluster with the second smallest norm and so on until no more splitting is accepted.

Algorithm 1 Top-down clustering with criterion

Parameter: A criterion Cri
Π initial partition with one cluster per class.
$S = \emptyset$
while $S \neq \Pi$ **do**
  $p = \text{argmin}_{p \in \Pi - S}(||p||)$
  Apply *2-means* on p. Let p_1 and p_2 be the two clusters.
  if Cri is verified **then** {Split is accepted}
    $\Pi = \Pi - p + p_1 + p_2$
    $S = \emptyset$
  else
    $S = S + p$
  end if
end while

3.3 Criterion *RP*: with relative positioning of clusters

$$\forall i,j \; c(\Pi_i) = c(\Pi_j) \Rightarrow \exists k, \; c(\Pi_k) \neq c(\Pi_i) \text{ and } \begin{cases} d(\Pi_i, \Pi_k) \leq d(\Pi_i, \Pi_j) \\ d(\Pi_j, \Pi_k) \leq d(\Pi_i, \Pi_j) \end{cases} \quad (5)$$

This constraint means that for each pair of clusters of the same class, there must exist a third cluster of another class *between* the two (see figure 2 for a small illustration). This constraint is obviously satisfied with one cluster per class. The criterion is then to maximize the number of clusters while keeping the constraint satisfied.

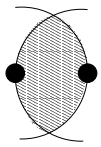

Fig. 2. Illustration of the "zone between two clusters".

At each step of the full clustering process, we only test if the two newly constructed clusters verify our criterion. The global constraint is not garanteed because we do not verify it for all the other pairs of clusters.

An additional threshold controls the clusters size preventing the building of too small solutions, which are often noisy and unreliable : clusters are never

accepted if they contain less than N_0 examples. If we don't use this filtering, the algorithm will create noisy clusters with very few examples. Then, these clusters will "help" other splits to be accepted. In our experiments, we see that without filtering, the algorithm continues very long and finishes with lots of very little clusters which is not what we want. We have chosen $N_0 = 20$ for all our experiments.

3.4 Criterion *DC*: with training documents categorization

$$\text{Min} \left| \left\{ x | \exists \Pi_1, \Pi_2 \text{ s.t. } \begin{array}{l} c(\Pi_1) = c(\Pi_2) = c(x) \\ \Pi_1,\ \Pi_2 \text{ nearest clusters of } x \end{array} \right\} \right| \tag{6}$$

This formula expresses the fact that we want to minimize the number of examples for which the two nearest prototypes are in the same class. The idea is that if this is true for a lots of examples, then we don't need to distinguish those two prototypes. As before, this formula is obviously minimal with one cluster per class. Whenever a second cluster is created for any class, there is almost always at least one example which doesn't satisfy the criterion anymore. We need to relax our formula to allow the creation of some clusters even if few examples don't satisfy the criterion. So clusters are accepted even though the formula is not minimized according to the relative number of non satisfied examples and total number of examples in clusters. New clusters are selected if less than $m * min(|p_1|, |p_2|)$ examples of the two clusters don't satisfy the constraint anymore, with m a free parameter. Experiments show that results are not very sensitive to the value of m (we tried from $m = 0.5$ to $m = 5$ with no significant differences).

Finally, we also need to point out that only correctly classified examples are taken into account in this criterion, which implicitely lowers the influence of noisy examples. With this caracteristic, there is no need to filter small clusters, filtering being already implicitely performed by the criterion.

4 Experiments and results

4.1 Corpora

We have used three different corpora: Newsgroups, Spam and Mail Center.

The *Newsgroups* corpus, collected by Ken Lang, contains 20000 messages, evenly distributed in 20 classes, each class corresponding to a different Usenet newsgroup[1]. A list of Newsgroups is given in table 1. Two new corpora were then derived: the first one is obtained by merging newsgroups with the same Usenet prefix[2], leading to a partition of messages into 4 homogeneous classes;

[1] The corpus can be downloaded at
`http://www-2.cs.cmu.edu/afs/cs/project/theo-20/www/data/news20.html`

[2] We have placed *soc.religion.christian*, *alt.atheism* and *misc.forsale* in the most similar classes so as to have super-classes with homogeneous sizes.

1. comp.graphics	11. alt.atheism	**Base 1**: non mixed subclusters
2. comp.windows.x	12. sci.electronics	C1 : 1,2,3,4,5
3. comp-os.ms-windows.misc	13. sci.crypt	C2 : 6,7,8,9,10,11
4. comp.sys.mac.hardware	14. sci.space	C3 : 12,13,14,15
5. comp.sys.ibm.pc.hardware	15. sci.med	C4 : 16,17,18,19,20
6. talk.politics.guns	16. misc.forsale	
7. talk.politics.mideast	17. rec.sport.baseball	**Base 2**: Intermixed subclusters
8. talk.politics.misc	18. rec.sport.hockey	C1 : 1,2,6,12,16
9. talk.religion.misc	19. rec.autos	C2 : 3,7,8,13,17
10. soc.religion.christian	20. rec.motorcycles	C3 : 4,9,10,14,18
		C4 : 5,11,15,19,20

Table 1. List of groups for the newsgroups corpus.

the second one results from the merging of unrelated newsgroups into four superclasses, so as to have intermixed subclusters. This corpus is later referred as the heterogeneous corpus.

The *Spam* corpus contains 2193 emails. The task is here to discriminate junk or unsollicited emails from the legitimate ones. The corpus contains 1460 spams for 733 legitimate emails. Messages in English and in French are mixed in the two classes.

The *Mail Center* corpus contains 2393 emails received by a customer service classified in 40 categories (see [18] for more information regarding this corpus). Messages are mostly written in French.

In all our experiments, two third of the corpus is used for training and the remaining part is used for testing.

4.2 Accuracy measures

Experiments have been performed for all corpora with the following algorithms: K nearest neighbors, Support Vector Machine, simple Rocchio, Clustered-SVM, MPR with K-Means, criterion-based clustering and a greedy division clustering. The greedy clustering always splits the cluster with the smallest norm. Clustered-SVM is similar to the work of [11]: after clustering, a SVM is learned for each cluster and new documents are assigned to the class of the best cluster. For algorithms where the number of clusters must be provided (such as K-Means), we use the number found by criterion-based clustering. To compare the algorithms, we use a more general performance measure than usual accuracy. Acc-n is defined as the pourcentage of testing examples for which the correct class is found in one of the n-best answers of the algorithm. This measure seems to provide a reasonable estimate of actual performance in applicative contexts involving a human validation [18]. We have shown in [19] that Rocchio performs comparatively much better with Acc-n ($n > 1$) than with Acc-1.

Results presented in table 2 suggest the following comments:

- SVM constantly outperforms all other algorithms for the Acc-1 measure.

	Newsgroups			Spam	Mail Center
	normal	heterogeneous	homogeneous		
Rocchio	0.810/0.921	0.754/0.921	0.892/0.973	0.771/.	0.508/0.724
K-NN	0.844/**0.944**	0.864/**0.970**	0.932/0.983	0.930/.	0.535/0.649
SVM	**0.865**/0.932	**0.890**/0.968	**0.959/0.990**	**0.974**/.	**0.578**/0.713
clustered-SVM	..	0.878/0.960	0.940/0.984	0.978/.	0.532/0.698
K-Means	0.810/0.921	0.815/0.944	0.904/0.973	0.959/.	0.527/0.726
Greedy clustering	0.809/0.921	0.804/0.943	0.908/0.975	0.948/.	0.560/0.729
Criterion RP	0.813/0.924	0.818/0.942	0.905/0.972	0.964/.	0.522/0.723
Criterion DC	0.818/0.922	0.813/0.932	0.907/0.973	0.962/.	0.562/**0.731**

Table 2. Performance measure: Acc-1/Acc-2

Bold values show the best algorithm for each corpus. There is no Acc-2 measure for the Spam corpus since it contains only two classes.

- Clustered-SVM is always lower than simple SVM, as in [11]. This confirms that clustering is useful for Rocchio but not necessarily for other classifiers.
- For the three newsgroups corpora, MPR is only slightly better than Rocchio and always worse than k-NN or SVM.
- For Spam and Mail Center databases, the improvement over Rocchio is very sensible and MPR surpasses k-NN and even SVM for Mail Center with Acc-2.
- MPR improves Rocchio more for Acc-1 than for Acc-2.
- The different clustering algorithms have a very similar behaviour. The main advantage of our criterion-based clustering is its ability to automatically determine the "right" number of clusters. Overall, the DC criterion is more stable and accurate and is used for the rest of our experiments.

4.3 Discussions about number of clusters found by MPR

Both implementations of MPR are also able to create useful clusters to improve overall accuracy. Unlike K-Means or any unsupervised clustering, they are able to choose the right number of clusters.

This fact is confirmed by an examination of the results on the Newsgroups homogeneous and heterogeneous corpora. For this experiments, the right number of clusters is known in advance: for the homogeneous corpus we expect clustering to be useless, whereas for the heterogeneous, we expect to get better results by recovering the 20 initial classes. Experimental results are reported in table 3. Obviously, MPR is able to see a difference between these two simulated corpora and to pick up an approximately correct number of clusters.

	homogeneous	**heterogeneous**
Optimal	4 or 20	20
Criterion RP	6.2 (0.905)	22 (0.818)
Criterion DC	5.4 (0.905)	14 (0.813)

Table 3. Number of clusters found by each algorithm.

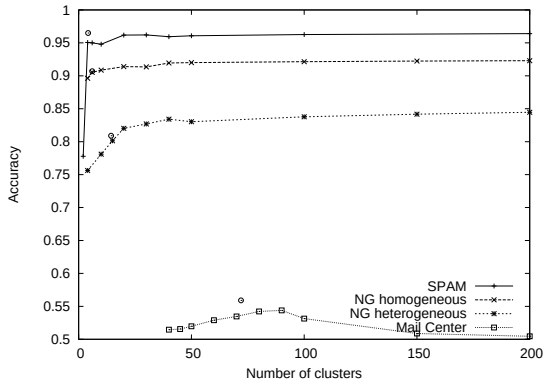

Fig. 3. Accuracy with increasing numbers of clusters.
The dots represent number of clusters and accuracy for MPR with criterion DC.

A second way to check that the number of clusters is correct is to do an exhaustive search. To this end, we have used the greedy clustering with an increasing number of clusters. Each of these is then integrated in the classifier. Results are presented on Figure 3. Our criterion seems to always find just a little less clusters than the optimum value, the resulting accuracy being slightly lower than the maximum found by greedy clustering. Overall, we think that the criterion-based clustering is able to find an appropriate number of clusters.

4.4 Influence of corpus size

Rocchio is very efficient and often outperforms other algorithms when the training corpus contains very few examples per class. The clustering algorithms presented here require an important number of examples to be able to create statistically coherent clusters. To see how the combination of clustering and Rocchio behaves on a small corpus, we have performed additional experiments with corpora of varying size.

Results are reported on Figure 4. As expected, Rocchio outperforms all others classifiers with fery few documents but its learning curve rapidly flattens. MPR is a compromise between Rocchio and k-NN / SVM with better accuracy than k-NN or SVM with few examples and better than Rocchio with lots of examples.

5 Conclusion and future work

We have presented here a weakly supervised clustering algorithm and demonstrate the usefulness of this learning procedure when used in conjonction with a Rocchio classifier for text categorization tasks. Unlike related work, this algorithm tries to use the supervision data (class labels) to guide clustering: this

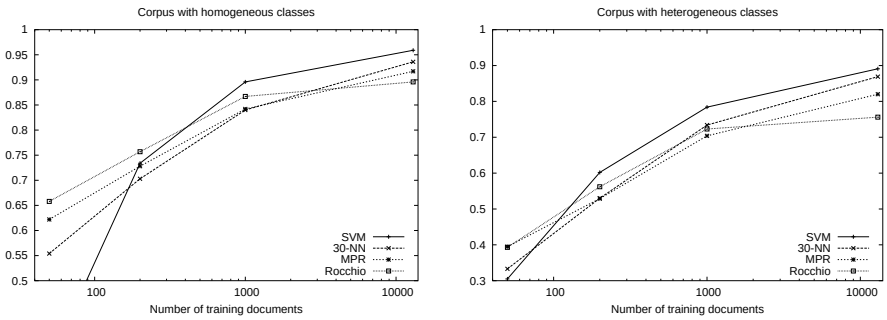

Fig. 4. Accuracy according to the number of learning examples.

makes our approach quite similar to discriminative learning, in which parameter estimation is based on optimizing a criterion based on performance measure on the training set. This strategy provides the clustering algorithm with a well-behaved stopping criterion which allows to automatically discover the right number of clusters. The criterion based on performance measures seems to be the most accurate and most stable one. Using this weakly supervised clustering, we have successfully managed to improve Rocchio's performance, with errors rate dropping between 4 % and 84 % depending on the corpus. These differences in performance confirms that Rocchio can use some extra information regarding the internal organization of clusters, but this information is not useful for all corpora. In our experiments, we found that the number of useful subclusters is relatively small, thus preserving the efficiency of Rocchio during the classification phase. We have also identified an important characteristic of our clustering algorithm: it requires more documents than a simple Rocchio classifier; in fact using it with a too small corpus can even lower accuracy.

Our clustering algorithm allows to discover some hidden structure on any textual database in a way that is beneficial for the classifier Rocchio. We plan to investigate the use of similar unsupervised clustering techniques for other tasks including forgetting of past examples after a concept shift and management of a temporal stream of documents by monitoring these clusters.

References

1. Chris Buckley and Gerard Salton. Optimization of relevance weights. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference of Research and Development in Information Retrieval*, pages 351–357, 1995.
2. P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, Ann Arbor, June 1988. Morgan Kaufmann Publishers.
3. H. de Kroon, T. Mitchell, and E. Kerckhoffs. Improving learning accuracy in information filtering. In *International Conference on Machine Learning - Workshop on Machine Learning Meets HCI*, 1996.

4. D. Eichmann, M. Ruiz, P. Srinivasan, N. Street, C. Chris, and F. Menczer. A cluster based approach to tracking, detection and segmentation of broadcast news. In *Proceedings of the DARPA Broadcast News Workshop*, pages 69–76, 1999.
5. Eui-Hong Han and George Karypis. Centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431, 2000.
6. Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML-98, Tenth European Conference on Machine Learning*, pages 137–142, 1998.
7. Wai Lam. Using a generalized instance set for automatic text categorization. In *Proceedings of SIGIR-98, 21th ACM International Conference on Research and Development in Information Retrieval*, pages 81–89, 1998.
8. David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of SIGIR-96*, pages 298–306, Zürich, CH, 1996.
9. Alessandro Moschitti. A study on optimal parameter tuning for rocchio text classifier. In *proceedings of the 25th European Conference on Information Retrieval Research (ECIR)*, Pisa, Italy, 2003.
10. Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Neural Information Processing Systems*, 2001.
11. J-H Oh, K-S Lee, D-S Chang, C. Won Seo, and K-S Choi. Trec-10 experiments at kaist: Batch filtering and question answering. In *Proceedings of The Tenth Text REtrieval Conference (TREC-10)*, pages 347–354, 2001.
12. Joseph John Rocchio. *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, Relevance Feedback in Information Retrieval, pages 313–323. Gerard Salton (editor), Prentice-Hall Inc., New Jersey, 1971.
13. Gerard Salton, A. Wong, and C.S. Yang. A vector space model for information retrieval. *Communications of the ACM*, 18(11):613–620, November 1975.
14. Robert E. Schapire and Yoram Singer. BoosTexter: A boosting system for text classification. *Machine Learning*, 39(2/3):135–168, 2000.
15. Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. In W. Bruce Croft, Alistair Moffat, Cornelis J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 215–223, Melbourne, AU, 1998. ACM Press, New York, US.
16. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
17. Amit Singhal, Mandar Mitra, and Christopher Buckley. Learning routing queries in a query zone. In *Proceedings of SIGIR-97, 20th ACM International Conference on Research and Development in Information Retrieval*, pages 25–32, Philadelphia, US, 1997.
18. Romain Vinot and Franois Yvon. Semi-automatic response in a Mail Center. In *ASMDA 2001, 10th International Symposium on Applied Stochastic Models and Data Analysis*, pages 992–997. Universit de Technologie de Compigne, 2001.
19. Romain Vinot and Franois Yvon. Quand simplicité rime avec efficacité : Analyse d'un catgoriseur de textes. In *Colloque International sur la Fouille de Texte (CIFT'02)*, pages 17–26, Tunisie, 2002.
20. Yiming Yang. An evaluation of statistical approach to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999.