Kernel-based text categorization

Radwan Jalam*, Olivier Teytaud**
*ERIC, Université Lumière Lyon 2; 5, avenue Pierre Mendès-France, F-69676 Bron cedex
**ISC, UMR CNRS 5015; 67 Bd Pinel, F-69675 Bron Cedex
{*jalam, oteytaud*}*@eric.univ-lyon2.fr*

Abstract

This paper presents some techniques in text categorization. New algorithms, in particular a new SVM kernel for text categorization, are developed and compared to usual techniques. This kernel leads to a more natural space for elaborating separations than the euclidian space of frequencies or even in verse frequencies, as the distance in this space is the most usual pseudo-distance between distributions. We give an application to the recognition of the author of a text, and put into relief that our kernel could be used for any classification of distributions. We experimentally discuss the efficiency of our algorithms, depending on the precision of the estimation of frequencies, and the possibility of building statistical bounds on the error. All our experiments are made on underconstrained problems.

1 Introduction

Being given Q classes of texts, we call **text categorization** the task of determining the class of T, T being a text, after learning on a labeled training set. This can include language recognition, or topic recognition. We have restricted our study to algorithms using N-grams, because of their generality (they could be used for any kind of sequences on a discrete alphabet, see for example applications in biology, or generalization to dimension 2) and their robustness to noise ([12]); we do not work on approaches based on dictionaries. The most usual methods are 1-NN with dissimilarity measures, and [11] or [18] conclude roughly that the most efficient method is SVMs. We confirm these comparisons and introduce new techniques, based upon a new kernel.

Definitions: A being an alphabet, a N-**gram** is a sequence of N elements of A. For $N = 1$, a N-gram is a **letter**; 2-grams are called **bigrams**, 3-grams are called **trigrams**. The set of **words** is the set of all maximal (for inclusion) N-grams for any N with no punctuation or space. One calls *N-profile* of a texts the sequence of the N-grams of this text, in decreasing order of frequency, with their frequencies ([3],[12]).

2 Textcategorization

The following parts present two areas of text categorization: the first one uses dissimilarity measures, the second one is based on encoding in $\mathbb{R}^n$ and classical learning algorithms.

2.1 Text categorization with (dis)similarity measures

Many algorithms used for text categorization are based on distances or more generally on similarities and dissimilarities. All these methods rely on k nearest neighbor algorithms. The difficulty in this k-nearest neighbors approach is the definition of a distance or pseudo-distance. The simplest and oldest one consists in building the profiles of each class and of the text, and then using the dissimilarity measure CT used by Cavnar and Trenkle in [3]. Being given two profiles P_1 and P_2, the CT-distance is defined as $CT(P_1, P_2) =$

$$\sum_{w \in P_1, R_{P_1}(w) < NMAX} min(|R_{P_2}(w) - R_{P_1}(w)|, DMAX)$$

where $|x|$ is the absolute value of x, $R_P(w)$, with w a N-gram and P a N-profile, is the rank of w in the profile P, (if w belongs to P, and $DMAX$ otherwise, e.g., $NMAX = 500$ and $DMAX = 1000$). Another possible "distance" is the Kullbach-Leibler (KL) ([14]) dissimilarity measure:

$$KL(T_1, T_2) = \sum_{N_g} f_2(N_g) log(\frac{f_2(N_g)}{f_1(N_g)})$$

where the sum is taken over all N-grams, with T_1 and T_2 some texts, and $f_i(N_g)$ the frequency of the N-gram N_g in the text T_i. To avoid too much strong penalization of unseen N-grams, half of the frequency of a N-gram which would occur once is added to $f_i(N_g)$ if N_g has frequency 0 in T_i. This is done in [14] and is close

to Laplace Smoothing, and another solution suggested by an anonymous referee is the Jensen Shannon divergence (Kullbach-Leibler dissimilarity with the mean). More information can be found in [4, 17].

Another possibility is the cosine dissimilarity measure; [10] uses a centered space on the mean of the frequency vectors; we here do not use this translation. This is the following:

$$COS(T_1, T_2) = 1 - \frac{\sum_{N_g} f_1(N_g) f_2(N_g)}{\sqrt{(\sum_{N_g} f_1(N_g)^2) \times (\sum_{N_g} f_2(N_g)^2)}}$$

We chose another dissimilarity measure, the χ^2 dissimilarity. This is the following:

$$\chi^2(T_1, T_2) = \sum_{N_g} \frac{(f_1(N_g) - f_2(N_g))^2}{f_2(N_g)}$$

One can symmetrize this "distance" by using $\chi^2(T_1, T_2) = 2\frac{(f_1(N_g) - f_2(N_g))^2}{f_1(N_g) + f_2(N_g)}$. We do this in our practical experiments. When $f_1(N_g)$ and $f_2(N_g)$ are 0, then we replace $\frac{(f_1(N_g) - f_2(N_g))^2}{f_1(N_g) + f_2(N_g)}$ by 0 which is its continuous extension.

2.2 Classification methods based on $\mathbb{R}^n$-encoding

Another approach consists in encoding documents by vectors, in order to classify points in $\mathbb{R}^n$. This allows the use of all classical methods: backpropagation neural networks, thanks to sparseness of vectors (see [18] for experiments with a backpropagation based upon lists), support vector machines (SVMs), k nearest neighbors in $\mathbb{R}^n$, which can used directly with the previous dissimilarity measures, too, decision trees, etc. E.g., let $w_1, ..., w_q$ be a finite set of *words* (or subwords - the essential parts of words for example), and let's define x_i as the number of occurings of w_i in T (or its frequency). x will be the vector associated with T. The finite set of *words* can indeed be the set of all the words included in the considered texts, or the set of all N-grams for a given N. This number α of occurings can be replaced by different functions of α; [13] lists different possibilities. It's possible to consider only significant variables among all these ones. Different solutions are possible, among which, for this kind of data, the most famous is likely the information gain criterion (see [19]). Experimental results from [11] show that as much as possible, we must keep all the variables - what will be done in the sequel.

3 A new positive definite kernel for SVM ?

Encoding in $\mathbb{R}^n$ allows the use of lots of training algorithms, and in particular SVMs (see [16]). But one can use SVMs in another way: we define $K(T_1, T_2) = exp(-d(T_1, T_2))$, with d one of the dissimilarity measures suggested above. We experimented $K(T_1, T_2) = exp(-\frac{\chi^2(T_1, T_2)}{\sigma^2})$. We conjecture that the function $k(T_1, T_2) = exp(-\frac{\chi^2(T_1, T_2)}{\sigma^2})$ is a positiv edefinite kernel. In an attempt of proving this conjecture, we used [2, corollary 2.11, p78] to prove that $\phi : (x, y) \mapsto \frac{1}{x+y}$ for $x, y > 0$ is positive definite; and then, by [2, p66-67], we could deduce that $\psi : (x, y) \mapsto (x - y)^2$ is negative definite. Theorem 1.12 of the same reference was then enough, in conjunction with theorem 22, to prove that the χ-squared kernel is positive definite. Unfortunately, as the interested reader can verify it, there's a mistake in this argument. Intuitively, it sounds reasonable that Mercer's condition is verified, but we could not prove it completely.

We so have a new kernel at our disposal, which has the following advantages:

- This pseudo-distance is "natural"; whereas with linear SVMs the distance is the euclidian distance in the space of frequencies (or inverse frequencies), we look for RBF (radial-basis-function) separations in a space with a classical distance among distributions.
- We can learn on a compact representation of data - a kernel matrix $m \times m$, with m the number of texts in the training set, with SVM or RBF.
- The hyperparameter σ can be chosen thanks to results of [1] showing that the fat-shattering dimension is bounded by a function of Lipschitz coefficients (these Lipschitz-coefficients depending upon σ) and of weighs. Moreover, for the experiments on the first benchmark, results were the same for lots of different values of σ.

4 How to use RBF networks for text categorization

As in the case of SVM, one can use an RBF network with the encoding of texts in $\mathbb{R}^n$; but one can use the χ^2 dissimilarity for example. As explained above, this corresponds to a linear separation in a feature space. This method is successfully tested below. The algorithm is summarized below, with (T_i) the family of labeled texts (used for training), (T'_i) the family of texts to be classified:

1. Let O be a matrix such that $O_{i,j} = 1$ if T_i belongs to class j, -1 otherwise else.
2. Let K be the matrix such that $K_{i,j} =$

$exp(-\frac{\chi^2(T_i,T_j)}{\sigma^2})$ and K_1 the matrix resulting of K by adjunction of a column of 1's at its right.

3. Let K' be the matrices such that $K'_{i,j} = exp(-\frac{\chi^2(T'_i,T_j)}{\sigma^2})$ and K'_1 the matrix resulting of K' by adjunction of a column of 1's at its right.

4. Let W be the weight matrix such that $K_1 \times W = O$, let $O' = K'_1 \times W$. W might be non-unique; we then choose W by multiplication of O by the pseudo-inverse of K_1, chosen with minimal norm. This problem is solved by the singular-value-decomposition algorithm, which has a good numerical stability.

5. We classify T'_i in class $argmax_k\ O'(i,k)$.

[11] explains (partly) the good behavior of SVMs on text categorization by its capacity to treat so many dimensions without having to select relevant variables. One can notice that RBF, with this particular kernel, verifies the same property: the training set is translated into a kernel matrix of size $n \times m$, taking into account *all* the information, with m the number of texts in the learning set.

The differences and similarities with the previous SVM algorithm are :

- With RBF there's no reason for W to be sparse. This is in favor of SVM.
- RBF does not minimize a geometrical margin as SVM, but as the pseudo-inverse algorithm looks for a minimum norm solution, the coefficients are supposed to be small, and (as in backpropagation, but here without problems of local minima) the resulting classifier is expected to have small γ-empirical error with $\gamma = 1$. So we can expect low fat-shattering dimension, low γ-empirical error, and bounds as practical as the ones of SVM (see [1] for definitions and detailed bounds).

5 Writer recognition: working on large samples

The success rate is evaluated by leave-one-out in the case of author recognition, as the training set is small (28 classes (= authors), 130 texts).

We use a set of French books (130), written by well known writers, like Balzac, Bloy, Corneille, Diderot, Engels, Flaubert, Fourier, France, Gaberel, Gautier, Gobineau, Hugo, Huysmans, Lamartine, Leibnitz, Maistre, Maupassant, Moliere, Pascal, Racine, Renard, Rostand, Rousseau, Sand, Stendhal, Verne, Voltaire, Zola. Some of this writers are translated from other languages. The complete list of titles is too much long for being listed here, but the used files can be asked by email to the authors. The fact that texts are not all formatted the same way hasn't been corrected, and is considered as a supplementary difficulty for the algorithm (notice that the formatting is not correlated with the author). Most of these texts come from the ABU site, `cedric.cnam.fr/ABU/`, the others from the Biblioth`eque Nationale de France, `www.bnf.fr/`. The experimental results (with 3-grams) are given in table 5.

Table 1: The result on the first line is the same for all these values of p.

Algorithm	Success Rate
RBF with $(\chi^2)^p$ kernel for $p = \frac{1}{2}, \frac{1}{4}, \ldots \frac{1}{32}$	87.69 %
RBF with χ^2 kernel	86.15 %
Multiclass svm with χ^2 kernel	86.15 %
Multiclass linear svm	78.46 %
SVM with χ^2 kernel	72.31 %
1-NN with χ^2 dissimilarity	70.77 %
linear SVM	67.69 %
1-NN with KL dissimilarity	52.31 %

All our tests are made with implementations in Octave (see `www.che.wisc.edu/octave` for a description of this very interesting free clone of Matlab). All the source codes can be asked by email to the authors. We call "multiclass SVM" a SVM designed for multiclass categorization, defined in [9]. It is worth putting into relief that in this case (high dimensionality, 28 classes) this SVM is significantly better than the usual method consisting in combining SVMs one-against-all as suggested in [16]. We have both SVM multiclass with χ^2 significantly better than SVM with χ^2 and linear SVM multiclass significantly better than linear SVM.
Our experiments gives the following results, with >> denoting a difference with confidence 5 %, $\geq$ a difference with confidence 15 %:

$$\{ \text{RBF - SVM Multiclass } (\chi^2) \} >> \text{SVM Multiclass} \geq \text{SVM } \chi^2 \text{ - SVM - 1-NN}$$

One can notice that our experiments, as the ones of [18], concern texts large enough for a nice approximation of frequencies. The following experiments will be done in another case.

6 Language recognition: working on small samples

In this case the success rate is evaluated by validation on a disjoint part of the data set. After the previous benchmark, one could conclude (too quickly) that RBF with χ^2 kernel seems to be the ultimate algorithm for text categorization. The multiclass version of SVMs looks as powerful as it, but RBF are much faster and more simple to implement. In our following experiments, we will focus on two algorithms: RBF, because of their efficiency shown in the previous benchmark, and 1-NN, because of its simplicity, efficiency in the following case as we will see in the experiments below, and because it's widely used in practical applications. The following experiments are made with Java implementations, based on the Jama matrix package. All java source codes can be asked by email to the authors. The task consists in recognizing in which language is written a given text. We work on five languages: French, Arabic, English, Spanish and German. As this is known a very easy task, we complicate it by using very small parts of texts. We detail a comparison on a particular set of 250 samples of 100 bytes, then 500 samples of 50 bytes, then 1250 samples of 20 bytes (20 bytes on average). We have 5 big texts of 5 Kb used to define profiles (come from G. van Noord's page `odur.let.rug.nl/~vannoord/TextCat/list.html`), and short samples from 5 languages (Arabic ones built with html pages, German ones from "Stochastic Language Identifier" (`www.dougb.com`), french ones from a book at `www.alyon.org`, English and Spanish ones from the corpus of [7]). All the used datasets can be asked by email to the authors. With a testing set made of samples of 100, 50 or 20 bytes, SR meaning "success rate", we get results of table 6.

Table 2: The result between parenthesis is got with 50 profiles (per class) computed on 50 subparts of the training set instead of one profile computed on the whole class of the training set. This leads to better results for some RBF learnings. This trick doesn't work as well for the experiments below with shorter samples.

Algo.	SR (100 b.)	SR (50 b.)	SR (20 b.)
1-NN (KL)	100 %	99.4 %	92.8 %
1-NN (χ^2)	98.8 %	96.6 %	87.92 %
RBF($\sigma^2 = 10$)	37.6 % (100 %)		
RBF($\sigma^2 = 100$)	98.8 %	93 %	71.04 %

We now work with 250 samples of 100 bytes as learning set, to study more precisely the influence of "gathering" learning texts for RBF or k-NN. Results are reported in table 6.

Table3: "m-g" means that the training texts have been gathered in sets of m texts; "gathered", that all texts of a class in the training set have been gathered (i.e. m-gathered, with m maximal). Keeping m small preserves the variability of the training set, m larger leads to more precise profiles.

Algo.	Hyperp.	SR (100)	SR (50)	SR (20)
RBF	$\sigma^2 = 10$	99.2 %	84.8 %	31.52 %
	$\sigma^2 = 100$	98 %	93.2 %	71 %
RBF (2-g)	$\sigma^2 = 100$	97.2 %	88 %	68.56 %
RBF (5-g)	$\sigma^2 = 1000$	98.8 %	94 %	80.88 %
RBF (10-g)	$\sigma^2 = 1000$	99.2 %	95.2 %	76.72 %
RBF (25-g)	$\sigma^2 = 1000$	98.8 %	94.8 %	82.4 %
RBF (g)	$\sigma^2 = 100$	88.4 %	80.6 %	
	$\sigma^2 = 10^5$	87.6%	77.4 %	61.36 %
1-NN	χ^2	99.2 %	96.6 %	88.4 %
1-NN	KL			47.2 %
1-NN (2-g)	χ^2	99.6 %	96.8 %	88.8 %
1-NN (5-g)	χ^2	100 %	97.6 %	90 %
1-NN (10-g)	χ^2	99.2 %	97.2 %	88.56 %
1-NN (10-g)	KL			89.84 %
1-NN (25-g)	χ^2	100 %	96.8 %	87.2 %
1-NN (g)	χ^2	100 %	93 %	84.56 %
1-NN (g)	KL	99.7 %	97.4 %	89.4 %

In the case of small testing samples, KL remains better than χ^2, but KL seems to be unable to work with short learning samples, as illustrated by the case of non-gathered learning samples. The hyperparameter σ^2 for RBF-learning was very easily chosen in the previous benchmark (classification by authors), as the success rate was constant for a wide range of σ and as empirical success was closely related to generalization success; *but in the case of* 20 *bytes strings*, the efficiency was very depending on σ and on the gathering; this leads to **two** difficult hyperparameters.

7 Conclusion

On datasets for which all frequencies are precise (what doesn't mean that they only depend upon the class - they depend upon the author, the language, the topic, the time of the writing...), one can finally sum up previous results ([18], [11]) and our results by:

RBF > SVM Mc (χ^2) > SVM Mc > SVM (χ^2) > SVM > 1-NN

> LLSF, C4.5, NNets > NB

With SVM Mc the multiclass SVM from [9], SVM being a classical one-against-all SVM, LLSF as described in [18], NNets being neural nets other than SVM, C4.5 being the most famous algorithm of induction trees (see [6] for a use in text categorization) and NB being the Naive Bayes algorithm (see [8]). Notice that RBF > SVM Mc is not significant in terms of performance; we keep this comparison as RBF have the advantage of being much faster for learning and much easier to implement. Our part of the result must be restricted to the case of relatively small learning samples. The good results resulting from linear separations in the Reproducing Kernel Hilbert Space associated to our symmetrized χ^2 distance (assuming that this kernel is positive definite!) suggests that this space is the natural place where one can study separations between classes of distributions.

Notice that some experimental results, in the case of overconstrained problems, have put into relief the fact that the Euclidean distance was often nearly as efficient as the χ^2 distance. Our experiments, as some others in histogram-based image-classification ([5],[15]), suggest that this is not true for underconstrained problems. Of course, this conclusion, as the benchmarking above, are based upon a generalization of experiments on a few benchmarks.

In the case of less-precise frequencies (with very small parts of text in the testing set), 1-NN becomes better than RBF, with KL > χ^2 provided that the learning set is large enough to compute precise frequencies. The results of [7] with Markov Models, with two languages instead of five here, compared with our results, suggest that Markov Models trained with $25Kb$ per language with 2 languages have nearly the same error rate than 1-NN with $5Kb$ per language with 5 languages, four of them being as difficult as the two ones of [7] - error rate for random classifier being 20% with 5 languages and 50% with 2 languages, 1-NN seem to be more adapted to this task than Markov Models. Our tested version of 1-NN uses 3-grams, as Markov models of order 2 (which are often the most efficient according to [7]); 1-NN do not require computations of bigger profiles than Markov Models. Moreover, k-NN can efficiently work only keeping one profile per class, what is not always true with RBF; k-NN have the advantage of robustness (any gathering of profiles, almost no hyperparameter); so we make the assumption that 1-NN and more generally k-NN are the most efficient solution to classify small samples of texts. The choice of the distance is an interesting question; because the dissimilarity CT isn't mathematically justified, and because the KL measure has difficulties for small learning samples (it implies particular cases for unseen N-grams and has an experimental bad behavior on small samples...) we prefer the χ^2 dissimilarity, which didn't give significantly worst results than other distances (KL, CT or cosine) with precise frequencies and sometimes much better ones; but we recall that for small testing sets KL gave the best results. The experiments of [14] confirm this point. Finally, we underline that a detailed study shows that for most of our algorithms errors come from unbalanced classifiers (ie one class is "invading" the others). This suggests that algorithms "helping" handicapped classes (typically boosting) could give good results. This might be the object of a further work.

Finally, we put into relief the fact that time complexity in the case of text mining, and especially in our experiments on underconstrained problems, does not have the same behavior than in many other problems. For example:

- The dimension is very large. [11] suggests that one should not remove inputs in order to get optimal results [1] . Decision trees, for example, become very slow. On the other hand, inputs being sparse, experiments such as the one of [18], show that implementations of backpropagation in neural networks can have a reasonnable computational time. SVM in recognition has a time complexity linear in the product of the number of support vectors and the dimension, RBF in recognition has a time complexity linear in the product of the number of support vectors and the dimension (except with polynomial kernels). So, SVM and RBF are probably not suitable when the learning sample is large.

- In our experiments, learning samples were small or medium (in the second part). The advantage of our algorithms was decreasing in the latter case, even if the time complexity was reasonnable. This suggests that simple algorithms remain better for large samples. For applications such that web mining, learning samples are usually very large.

- For very fast recognition time, using an important dimension reduction is a solution. Nevertheless, hash tables with linear kernels are almost as fast.

As a conclusion of these computational considerations, for learning tasks with large learning samples and with short recognition time, RBF of SVM with linear kernels would probably outperform RBF or SVM with χ^2 or gaussian kernel. Moreover, for very large learning samples, both RBF and SVM could probably not be used anyway; some articles reported results with very large training samples, but they did not report results better than simple algorithms. For indexed documents, com-

[1] For the sake of intuitive understanding of the resulting classifier, such a reduction of dimension could be of some use. We here only consider the point of view of the error rate.

pletely different techniques should be used. This puts into relief the fact that our study is not more relevant when large learning samples are involved (a few thousands examples). When very small recognition time are necessary (and when indexation is not possible), RBF or SVM with linear kernels, or backpropagation with implementation by lists, could outperform decision trees, but for very fast results it is likely that decision trees remain the fastest solution.

Perhaps fast implementations of Support Vector Machines could place SVM beyond RBF from the point of view of time complexity. Moreover, results of Support Vector Machines are sensitive to the precision of the implementation of the quadratic problem. Results are not the same for interior points, sequential minimization; the cost function has small derivatives near the limit point, and all these algorithms rely on many heuristics, which make all implementations different. On the other hand, algorithms for pseudo-inverse are old and well known, so there is a kind of bias introduced in our comparison, due to the fact that RBF was efficiently implemented, whereas SVM was applied with an experimental implementation of a sequential optimization algorithm. So, perhaps better implementations of Support Vector Machines could modify the conclusions presented in these lines.

We thank André Elisseeff for the multiclass SVM and for fruitful discussions. We are grateful to B. Schölkopf for the reference [2].

References

[1] P.-L. Bartlett, *The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network, IEEE transactions on Information Theory, 44:525-536, 1998.*

[2] C. Berg, J.-P.-R. Christensen, P. Ressel, *Harmonic Analysis on Semigroups, Theory of Positive Definite and Related Functions, Springer, 1984*

[3] W.-B. Cavnar, J.-M. Trenkle, *N-gram Based text categorization. In 1994 Sympsium on Document Analysis and Information Retrievalin Las Vegas, 1994*

[4] M. Cover, J.A. Thomas, *Elements of Information Theory. Wiley Series in Telecommunictions. New York, 1991*

[5] Chapelle O., P. Haffner, and V.-N. Vapnik, *Support Vector Machines for Histogram Based Image Classification, IEEE transactions on Neural Networks, Vol 10,1999*

[6] S.-L. Crawford, R.-M. Fung, L.-A. Appelbaum,R.-M. Tong, *Classification trees for information retrieval, in Machine Learning: proceedings of the eighth International Workshop (1991), Morgan Kaufmann, pp 245-249*

[7] T. Dunning, *Statistical Identification of languages, Computing Research Laboratory Technical Memo MCCS 94-273, New Mexico State University, Las Cruces, New Mexico, 1994*

[8] I.-J. Good, *The estimation of probabilities: An Essay on Modern Bayesian Methods, MIT Press, 1965*

[9] Y. Guermeur, A. Elisseeff, H. Paugam-Moisy, *A new multiclass SVM based on a uniform convergence result. IJCNN'2000*

[10] S. Huffman, *Acquaintane: Language-Independent Document Categorization by N-Grams, in TREC 4 Proceedings, 1996*

[11] T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features, in Machine Learning: ECML-98, Tenth European Conference on Machine Learning, pp. 137–142, 1998*

[12] E. Miller, D. Shen, J. Liu, C.Nicholas, *Performance and Scalability of a Large-Scale N-gram Based Information Retrieval System, Journal of Digital Information 1, no 5, 1999*

[13] Mehran Sahami, *Thesis:Using Machine Learning to Improve Information Access, Ph.D. in Computer Science, Stanford University, 1999*

[14] P. Sibun, J.C. Reynar, *Language identification: Examining the issues. In Symposium on Document Analysis and Information Retrieval, pp. 125–135, Las Vegas, 1996*

[15] O. Teytaud, D. Sarrut, *Kernel-Based Image Classification, accepted in ICANN 2001*

[16] V.N. Vapnik, *The Nature of Statistical Learning, Springer, 1995*

[17] J.J. Verbeek, *An information theoretic approach to finding wordgroups for text classification, Institute for Logic, Language and Computation (ILLC-MoL-2000-03), 2000.*

[18] Y. Yang, X. Liu, *A Re-Examination of Text Categorizarion Methods, SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA. ACM, 1999*

[19] Y. Yang, J. Pedersen, *A comparative study on feature selection in text categorization, in International Conference on Machine Learning (ICML), 1997*