Does a New Simple Gaussian Weighting Approach Perform Well in Text Categorization?

Giorgio Maria Di Nunzio
Dip. di Ingegneria dell'Informazione
Universita degli Studi di Padova
Via Gradenigo 6/b, 35131 Padova - Italia
dinunzio@dei.unipd.it

Alessandro Micarelii
Dip. di Informatica e Automazione
Universita degli Studi "Roma Tre"
Via della Vasca Navale 79,00146 Roma - Italia
micarcl@dia.uniroma3.it

Abstract

A new approach to the Text Categorization problem is here presented. It is called Gaussian Weighting and it is a supervised learning algorithm that, during the training phase, estimates two very simple and easily computable statistics which are: the Presence *P,* how much a term / is present in a category *c*\ the Expressiveness E, how much / is present outside *c* in the rest of the domain. Once the system has learned this information, a Gaussian function is shaped for each term of a category, in order to assign the term a weight that estimates the level of its importance for that particular category. We tested our learning method on the task of single-label classification using the Reuters-21578 benchmark. The outcome of the result was quite impressive: in different experimental setups, we reached a micro-averaged Fl-measure of 0.89, with a peak of 0.899. Moreover, a macro-averaged Recall and Precision was calculated: the former reported a 0.72, the latter a 0.79. These results reach most of the state-of-the-art techniques of machine learning applied to Text Categorization, demonstrating that this new weighting scheme does perform well on this particular task.

1 Introduction

Consider the problem of automated classification of text documents. This problem is of great importance as the accessible textual information increases and the volume of online texts available through the Internet expands rapidly. One solution is to categorize documents according to their topics beforehand or in real time.

A number of different Machine Learning methods have been applied and to Text Categorization (TC), including probabilistic classifiers, decision trees, regression methods, neural networks and support vector machines (see [Sebastiani, 2002]). Most of these methodologies share a common factor of high level complexity, that, often, makes a classifier design difficult to understand.

The question addressed in this paper is if it is possible to find a simple learning algorithm that uses naive, human comprehensible parameters, in such a way as to act like a non-expert human being in front of the problem of classifying new unknown documents.

Starting from this idea of simplicity, we decided to design a training algorithm that corresponds to the action of finding two information parameters: given a set of categories *C* of documents, for each term *t* of a category $c \in C$ we calculate its *Presence P,* that is, the percent of documents of the category *c* in which the term / appears at least once; and its *Expressiveness E*", that is, how much the term / is absent in the other categories of the domain. Then, we model a Gaussian Function with the two parameters above, whose value in the abscissae equal to 1 is used as the weight of the term /. We called this learning approach Gaussian Weighting. It is important to stress the fact that we don't use any stemming algorithm or Feature Selection function (see [Yang and Pedersen, 1997]). An eventual reduction of the number of features per category is done using only two thresholds named *ThresP* and *ThresE,* relative to the parameters *P* and *E.*

To make the evaluation of Gaussian Weighting comparable to most of the published results on TC, we chose the Reuters-21578[1] corpus as benchmark for the single-label classification task. Three different tests have been performed with a complete automated learning approach: the first run has been done without the use of a Gaussian Function, but simply giving a weight proportional to *P* and *E:* the second and the third test runs have been done with the Gaussian Weighting approach proposed here.

The outcome of the results was quite surprising: throughout the different experimental setups, we repeatedly reached a micro-averaged Fl -measure around the 0.89, with a peak of 0.899. Then, since the micro-average is known to be highly influenced by frequent categories (see [Yang and Liu, 1999]), we decided to compute the macro-averaged Recall and Precision. Again, the results reached a satisfactory macro-Recall around 0.72 and a macro-Precision of 0.79. These results reach most of the state-of-the-art techniques of machine learning applied to Text Categorization (see [Dumais *et al.,* 1998]), demonstrating that this new weighting scheme does perform well on this particular task. Moreover, the low computational cost of the algorithm we propose, makes this approach particularly preferable when the computing power is low.

[1] www.daviddlewis.com/resources/testcollcction/reuters21578

2 The Algorithm

In this section we present our supervised algorithm. First, we will analytically define the two parameters: Presence *P* and Expressiveness E. Then, we will explain how the training algorithm works and how the system classifies new documents at run-time.

2.1 Presence *P* and Expressiveness *E*

A central key of this naive approach is the computation of two easy human understandable parameters that we call: Presence *P,* and Expressiveness *E*. The former captures how much a term *t* appears at least once in the documents belonging to a category c, the latter estimates how much the same term *t* does not appear in the documents of the other categories. Given a set of categories $C = \{c_1, c_2, ..., c_i, ..., c_n\}$, each category having a number of documents *m,* (e.g. $c_1 = \{d_{1,1}, d_{1,2}, ..., d_{1,j}, ..., d_{1,m}\}$), we give the following definitions:

- The number of documents D_i of the i-th category is:

$$D_i = |c_i| = \sum_{j=1}^{m} d_{i,j} \quad (1)$$

- the number of documents of category / in which the term / is present:

$$D_{i|t} = |c_{i|t}| = \sum_{j=1}^{m} (d_{i,j}|t) \quad (2)$$

Note that we use the symbol *|" to denote the presence of a term inside a category or a document. This is not to be confused with the same symbol when used for the conditional probability (P(x|y)).

Then, the Presence *P* of a term *t* in the i-th category is, using (1) and (2):

$$P_{i|t} = \frac{D_{i|t}}{D_i} \quad (3)$$

while, the Expressiveness *E* of term *t* in the i-th category is calculated using 1, 2):

$$E_{i|t} = 1 - \frac{\sum_{p=1}^{n} P_{p|t}}{|C|-1} \quad p \neq i \quad (4)$$

It is important to stress that the same term in different categories has different values of expressiveness. This fact is better explained in Table 1. A term *t* is present in all the three categories with presence $P_{i|t}$ shown in the first row. The second row is the expressiveness $E_{i|t}$ of the same term for each category. Note how / of category c_1 has a greater expressiveness given the relatively smaller presence of the term in the rest of the domain.

	c_1	c_2	c_3
$P_{i\|t}$	0.9	0.4	0.2
$E_{i\|t}$	0.7	0.45	0.35

Table 1: A numerical example of Presence and Expressiveness.

Locality of Presence and Expressiveness

The approach uses a weighting method for terms per each class, which is in a sense a local type of weighting/modeling scheme of classes. This idea is similar to the Local LSI representation [Hull, 1994] where, in order to characterize the term space, the singular value decomposition is applied to a matrix consisting only of the known relevant documents (in our case the documents belonging to a class).

There are substantial differences with the ***Tf-Idf*** weighting scheme. ***Tf*** counts the number of occurrences of a term / in a document, while *P* counts the documents in which / appears at least once. ***Idf*** computes the number of documents in which *t* appears over the whole collection, meanwhile *E* calculates, according to which class we are computing the weights, the partial averaged Presence of/ in the domain.

2.2 Learning the Gaussian Weights (GW)

In this paragraph we explain how we shape a Gaussian function using the values *P* and *E*. Starting from the definition of a generic Gaussian function:

$$G(x) = exp\left(-\frac{(x-\mu)^2}{2 \cdot \sigma^2}\right) \quad (5)$$

we estimate the parameters $\hat{\mu}$ and $\hat{\sigma}^2$ as follows:

$$\hat{\mu} = P_{t|i} \quad (6)$$

$$\hat{\sigma}^2 = P_{t|i} \cdot E_{t|i} \quad (7)$$

Then, the Gaussian Weighting GW for a term *t* of a category i is defined, substituting (6) and (7) in (5), as:

$$GW(x)|_{x=1} = exp\left(-\frac{(x-\hat{\mu})^2}{2 \cdot \hat{\sigma}^2}\right) \quad (8)$$

The GW calculated for x = 1, returns a real value belonging to the (0,1] interval. In particular, the maximum weight can be reached only when the Presence of a term is equal to 1, that is to say, when it appears in each document of the category of interest. Figure 1 shows an example of a GW with *P* = 0.5 and *E* = 0.5 (continuous), and another one with *P* = 0.5 and *E* = 0.9 (dashed); when two terms have the same Presence, the one with a higher Expressiveness has a higher weight too.

Computing the parameter *P*

The algorithm to compute *P* is the following:

1. For each category $c_i \in C$
2. $D_i = 0$
3. For each document $d_{i,j} \in c_i$
4. $D_i = D_i + 1$
5. For each unique term t in $d_{i,j}$
6. if t is in $D_{i|t}$
7. then $D_{i|t} = D_{i|t} + 1$
8. else Add $D_{i|t} = 1$
9. end For
10. end For
11. For each term in $D_{i|t}$

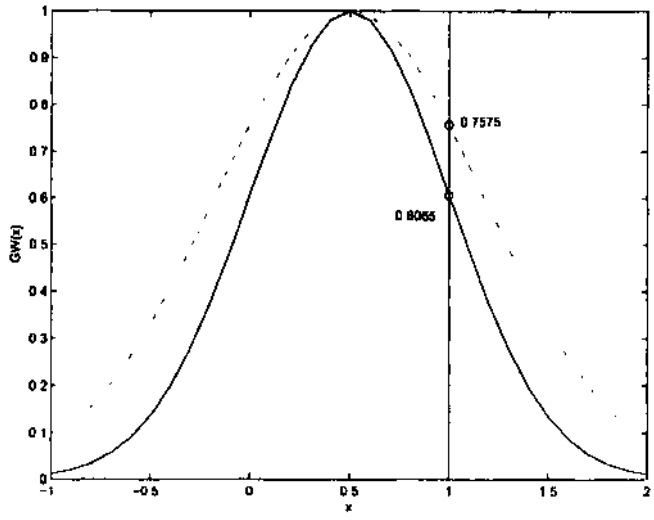

Figure 1: Example of a GW. The continuous line is a GW with *P* = 0.5 and £ = 0.5. The dashed line is a GW with *P* = 0.5 and *E* = 0.9. For the same *P,* the higher expressiveness a term possesses, the bigger the output GW gives.

12. $P_{i|t} = \frac{D_{i|t}}{D_i}$
13. end For
14. end For

If we assume we can use a HashMap HM to store the results of $D_{i|t}$, such that any update or search in the HM has a constant cost K, in the worst case the algorithm to compute the parameter *P* has a computational cost of $O(C \times D \times T \times k)$

Computing the parameter *E*

The algorithm to compute *E* is the following:

1. For each category $c_i \in C$
2. sum- 0
3. For each term t in $P_{i|t}$
4. For each category $c_j \in C$ where *i* =£ *j*
5. if t in $P_{j|t}$
6. then sum =• sum + $P_{j|t}$
7. end For
8. $E_{i|t} = 1 - \frac{sum}{|C|-1}$
9. end For
10. end For

Again, if we assume that $P_{j|t}$ is a HashMap with a constant cost *k* of access and search, the computational cost of this second algorithm is $O(C^2 \times T \times k)$,, which is quadratic cost with respect to the number of categories *C.*

2.3 Categorizing a new document

Once the system has been trained and the parameters P and E have been found for each term of the domain, it is possible to feed the system with an unknown document and, according to the value of a couple of optional thresholds *ThresP* and *ThresE,* classify it with the following algorithm:

1. For each $c_i \in C$
2. *output* $c_i = 0$, $Nterm = 0$
3. For each term in $P_{i|t}$
4. if t is in test
   And $P_{i|t} > ThresP$ And $E_{i|t} > ThresE$
5. then $outputc_i = outputc_i + GW(1,t,c)$
   Nterm = Nterm + 1
6. end For
7. $outputc_i = \frac{outputc_i}{n}$
8. end For
9. Assign the document to the c_i with the highest $outputc_i$

For each category, the algorithm calculates, the mean of the activated Gaussian functions of the terms whose $P_{i|t}$ and $E_{i|t}$ are greater than two fixed thresholds, respectively *ThresP* and *ThresE.* The computational cost of this algorithm is $O(C \times T \times k)$.

3 Experimental Setup

3.1 Test Collection

To make the evaluation of GW comparable to most of the published results on TC, we chose the Reuters-21578 corpus as benchmark. During the last few years this corpus has been used as a standard benchmark on which many TC methods have been evaluated, although the results are sometimes difficult to compare as slightly different version have been used. For this paper we used the ModApte split of Reuters-21578 in which 75% of the stories (9603) are used as training documents to build the classifier and the remaining 25% (3299) to test the accuracy of our single-label classifier. Now, the Reuters-21578 is known to be quite unbalanced on the distribution of stories per category. Therefore, of the 135 potential topics categories only the 10 most frequent are here used; these 10 categories account for almost the 75% of the training instances, while the remainder is distributed among the other 115. Table 2 shows the number of training and test samples for each category.

Category	# Training	# Test
Earn	2877	1087
Acquisitions	1650	719
Money-fx	538	179
Grain	433	149
Crude	389	189
Trade	369	118
Interest	347	131
Ship	197	89
Wheat	212	71
Corn	182	56

Table 2: Number of Training and Test documents for the ten most frequent categories of Reuters-21578 ModApte split

3.2 Parameters Setting

We performed three different test runs on our system:

1. The 'what if?' test run: what if we didn't use a Gaussian function? we tried to run our system with the most simple function we could use: $GW = P \cdot E$
2. The GW test run: uses the GW function defined in equation (8)
3. The simple GW test run: we simplified the Gaussian function with a variance equal to $\sigma^2 = P_{t,l}$

All the tests maintained *ThresE* equal to 0.6, that is to say, a term in a category should have an Expressivity value greater than 0.6. This value permitted to achieve the highest performance during the test phase. *ThresP* has been varied in the range between 0.0 (all the terms of the categories were used) and 0.5 (only the terms present in the 50% of the training documents were used).

The number of terms per category according to the value *P* is reported in Table 3.

	0,0	0,1	0,2	0,3	0,4	0,5
acq	11115	65	15	7	6	3
corn	3257	82	20	12	7	3
crude	6126	99	24	12	4	1
earn	10455	46	19	12	9	6
grain	5448	64	17	10	5	3
interest	3863	92	20	9	6	4
money-fx	5545	116	30	13	4	2
ship	4084	80	17	3	0	0
trade	5795	149	37	17	6	3
wheat	3543	75	17	10	4	4
feat/cat	592.3	87.8	21.6	10.5	5.1	2.9

Table 3: The table shows the number of terms per category according to the threshold of Presence *P*. The last row reports the average of features per category.

A stoplist of 331 terms has been used to remove the most frequent words of the English Language, but no stemming or Feature Selection have been performed.

In order to evaluate the accuracy of the classifier we have computed the standard 1R measures, such as Recall and Precision, and the following averaging measures: micro-Recall (μRe), micro-Precision (μPr) and micro-averaged FI measure (micro-FI):

$$\mu Re = \frac{\sum_{category} \#correctly\ classified}{\sum_{category} \#belonging\ to\ category}$$

$$\mu Pr = \frac{\sum_{category} \#correctly\ classified}{\sum_{category} \#classified\ under\ category}$$

$$microF1 = \frac{2 \cdot \mu Re \cdot \mu Pr}{\mu Re + \mu Pr}$$

In particular we have reported the values of the FI-measure in order to directly compare our results with the ones in the literature. But, since the micro-averaged measure is known to be highly influenced by the most frequent category (see [Yang and Pedersen, 1997]), we have decided to calculate the Macro-Recall (MRe) and the Macro-Precision (MPr)), as well:

$$MRe = \frac{\sum_{category} Re}{|category|}$$

$$MPr = \frac{\sum_{category} Pr}{|category|}$$

4 Results

4.1 The 'What if?' test run

In this test, the weight of each term / of a category *i* is computed as:

$$GW = P_{t|i} \cdot E_{t|i} \qquad (9)$$

The 'What if?' test run results are shown in Figure 2.

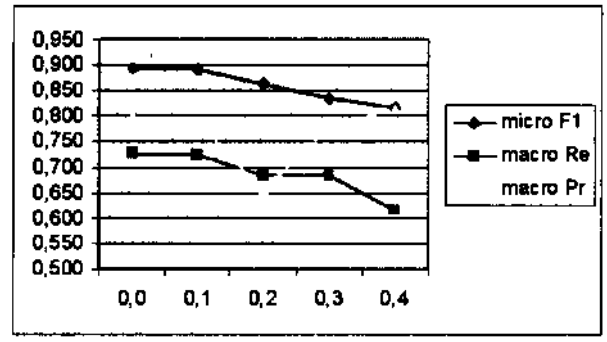

Figure 2: The 'What if?' test run. The x-axis represents the values of Presence P.

	0.0	0.10	0.20	0.30	0.40	0.50
micro F1	**0.894**	0.891	0.862	0.836	0.816	*
macro Re	**0.727**	0.725	0.682	0.684	0.615	*
macro Pr	**0.797**	0.789	0.656	0.778	0.814	*

Table 4: The 'What if?' test run.

In this particular test we did nOt expect so remarkable a performance. Considering the weight of a term proportional to the parameters *P* and *E* seems to be another possible solution to investigate. The performance of the system seems not to be influenced significantly when the threshold *ThresP* is set either to 0.0 or 0.1. It is worth stressing that this test performed better when all the terms (except those belonging to the stoplist) were used.

4.2 The GW test run

The results of this run are shown in Figure 3, while numerical values are reported in Table 5.

This test has been performed using equation (8) to calculate the weight of each term of the category. The best results have been obtained with $ThresP = 0.1$. Macro-Recall and macro-Precision are quite high, and they indicate that the system works well for every category. The performance of the system starts to decrease sensibly when $ThresP > 0.2$

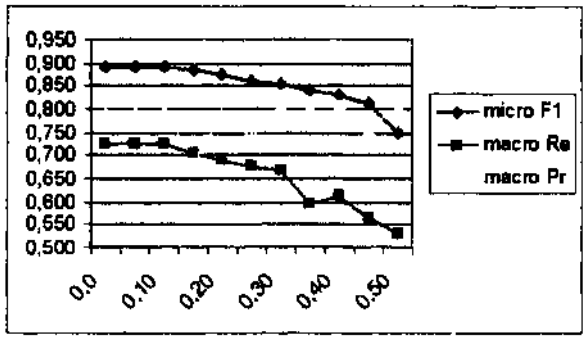

Figure 3: The GW test run. In this test the function 8 is used to weight each term.

	1 0.0	0.10	0.20	[0.30	0.40	0.50
1 micro F1	0.892	0.893	0.876	0.856	0.832	0.748
(macro Re	0.726	0.725	0.692	0.666	0.613	0.532
1 macro Pr	0.794	0.797	0.789	0.810	0.809	0.804

Table 5: The GW test run. Each column reports the values of micro F1, macro Recall and macro Precision for each *ThresP.*

4.3 The Simple GW test run

The results of this second run are shown in Figure 4. Numerical values are reported in Table 6.

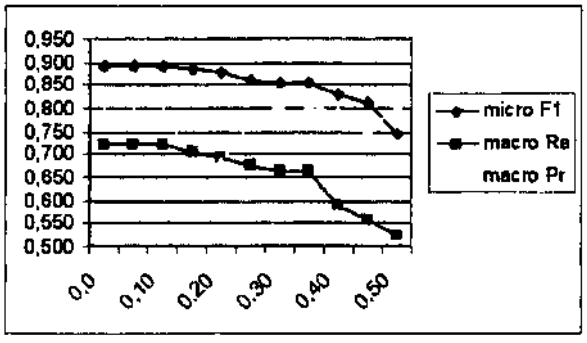

Figure 4: The simple GW test run. In this test the variance of the GW is equal to $\sigma^2 = P_{t,l}$.

This test has been performed using a simplified version of equation (8) to calculate the weight of each term of the category. The variance of the Gaussian function is set to $\sigma^2 = P_{t,l}$. The best results have been obtained with *Thresp*= **0.1,** there does not seem to be any particular difference with the GW test run, except in cases when ThresP equals 0.2, where the macro Precision is sensibly lower.

4.4 Discussion

The results reported above are satisfactory. In many situations the system reached an accuracy comparable to the ones in the state-of-the-art systems. Some aspects need to be discussed here: the performance of the system in all the test runs decades when *ThresP* exceeds the 0.2, that is to say, looking back to Table (3), when the number of features per

	0.0	0.10	0.20	0.30	0.40	0.50
micro F1	0.891	**0.893**	0.877	0.855	0.830	0.747
macro Re	0.722	**0.720**	0.693	0.662	0.588	0.524
macro Pr	0.791	**0.793**	0.685	0.806	0.791	0.802

Table 6: The simple GW test run.

category decreases from almost 90 to almost 20. This sharp decay influences the macro-averaged performance of the system. In fact, while the macro-Recall retains the same values, the macro-Precision is cut-off by almost 10%. This fact indicates that the system is incorrectly assigning the stories to all the categories of the domain rather that to the biggest ones only. As the *ThresP* gets bigger, the performance of the system shows the classic behavior in which the Recall tends to 0 and the Precision to 1.

For the 'what if?' test run we didn't expect such good results. The simple proportional weight assignment performed as well as the other approaches, nevertheless the performance decays slightly more rapidly. This test was very important for us to confirm the idea of using a simplified learning approach to solve the problem of TC.

The second and the third test runs show almost the same values. The one-per-thousand differences are not to be considered relevant. At the moment, we cannot draw any conclusion about which of the three learning approaches will perform better in general. We plan to investigate this matter testing on the complete Reuters-21578 and on other test collections.

Comparative Results

Dumais et al. tested a number of inductive learning algorithms on the same Reuters-21578 ModApte split we used ([Dumais *et ai,* 1998]). These results arc briefly summirized in Table 7 for the 10 most frequent categories. The

Method	Performance
FindSim	0.64
Naive Bayes	0.81
BayesNets	0.85
Tree	0.88
Linear SVM	0.92

Table 7: Microavcraged Fl-measure of the 10 most frequent categories of Reuters-21578, reported by Dumais

FindSim method is a variant of Rocchio's method for relevance feedback([Rocchio, 1971]). The Naive Bayes classifier is constructed by using the training data to estimate the probability of each category given the document feature values of a new instance. A discussion of the indipendence assumption of naive bayes classifier can be found in ([Lewis, 1998]). BayesNets is a bayesian network, that uses 2-dependence Bayesian (see [Sahami, 1996] for another example of Bayes nets for classification). Tree is a Decision Tree approach described in ([Chickering *et al,* 1997]). Meanwhile, Linear SVM is a linear hyperplane that separates a set of positive examples from a set of negative ones ([Joachims, 1998]).

Both SVMs and Decision Trees produce very high overall classification accuracy. As the results of Table 7 are directly comparable with ours, our Gaussian Weighting approach is placed just behind the SVMs, which are known to be the best machine learning method in the field of text classification, with a micro-averaged Fl-measure of 0.893 of the first two test runs. This result is important for two reasons: it demonstrates that this new learning method does perform well on this particular task, and encourage us to further investigate other fields of application, in which GW's simplicity could perform as well as other state-of-the-art machine learning methods.

5 Future Work

Two are the most compelling issues that we are going to complete in the future work are two:

- Incorporate a wrapper phase [Kohavi and John, 1997] to run systematically varying experiments with (i) using Expressiveness or not, (ii) varying values of ThresP and ThresE, (iii) varying values of GW variance, and (iv) using a Gaussian function or not;
- Test the full Reuters benchmark data rather than restricting ourselves to only the 10 most frequent categories, and on other collections like 20 Newsgroups[2] or MedLine[3].

6 Conclusions

In this paper we addressed the problem of finding a simple learning algorithm that uses naive, human comprehensible parameters. A very accurate text classifier can be learned automatically by using only two information parameters: presence *P* and expressivity *E.* The algorithm proposed here has, in the worst case, a quadratic computational cost, with respect to the number of categories of documents. We tested GW on the Reuters-21578 benchmark and calculated the micro-averaged Fl-measure to directly compare our results with the ones of the literature, and the macro-Recall and macro-Precision. The results achieved are satisfactory and can be compared to most state-of-the-art machine learning methods applied to TC.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful and helpful comments. The authors would also like to thank Prof. Maristella Agosti for her valuable comments and suggestions to improve the paper.

References

[Chickering *et al,* 1997] D. Chickering, D. Heckerman, and C. Meek. A bayesian approach for learning bayesian networks with local structure. In *Proceedings of Thirteenth Conference on Uncertainty in Artificial Intelligence,* pages 307-315. Springer Verlag, Heidelberg, DE, 1997.

[Dumais *etal,* 1998] S. Dumais, J. Piatt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management,* pages 148-155. ACM Press, 1998.

[Hull, 1994] D.A. Hull. *Information Retrieval Using Statistical Classification.* PhD thesis, University of Stanford, 1994.

[Joachims, 1998] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nedellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning,* number 1398, pages 137-142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[Kohavi and John, 1997] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence,* 97(l-2):273-324, 1997.

[Lewis, 1998] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nedellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning,* number 1398, pages 4-15, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[Rocchio, 1971] J.J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document PRocessing,* pages 313-323. G.Salton editor, 1971.

[Sahami, 1996] M. Sahami. Learning limited dependence Bayesian classifiers. In *Second International Conference on Knowledge Discovery in Databases,* 1996.

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

[Yang and Liu, 1999] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International S1GIR,* pages 42-49, Berkley, 1999.

[Yang and Pedersen, 1997] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, Nth International Conference on Machine Learning,* pages 412-420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[2] www.ai.mit.edu/jrennie/20Newsgroups/

[3] www.nlm.nih.gov