Error Correcting Codes with Optimized Kullback-Leibler Distances for Text Categorization

Jörg Kindermann, Gerhard Paass, and Edda Leopold

GMD – German National Research Center for Information Technology
D-52754 Sankt Augustin

Abstract. We extend a multi-class categorization scheme proposed by Dietterich and Bakiri 1995 for binary classifiers, using error correcting codes. The extension comprises the computation of the codes by a simulated annealing algorithm and optimization of Kullback-Leibler (KL) category distances within the code-words. For the first time, we apply the scheme to text categorization with support vector machines (SVMs) on several large text corpora with more than 100 categories. The results are compared to 1-of-N coding (i.e. one SVM for each text category). We also investigate codes with optimized KL distance between the text categories which are merged in the code-words. We find that error correcting codes perform better than 1-of-N coding with increasing code length. For very long codes, the performance is in some cases further improved by KL-distance optimization.

1 Introduction

Automatic text categorization has become a vital topic in many text-mining applications. Imagine for example the automatic classification of Internet pages for a search engine database. There exist promising approaches to this task, among them, support vector machines (SVM) [9] are one of the most successful solutions. One remaining problem is however that SVM can only separate two classes at a time. Thus the traditional 1-of-n output coding scheme is applied in this case: n classes will need n classifiers to be trained independently. Early alternative solutions were published by Dietterich and Bakiri [4], and Vapnik [14, p438]. More recently, research in multi-class SVMs increased, see for example Guermeur et al. [7], Platt et al. [13], Crammer et al. [2], and Allwein et.al. [1].

Hsu and Lin [8] report that for many categories, multi-class solutions which construct a system of binary SVMs have advantages in computational resources over integrative extensions of SVM. It is therefore interesting to apply the approach of [4] to text categorization. Dietterich and Bakiri use a distributed output code. A second argument is that if the output code has more bits than needed to represent each class as a unique pattern, the additional bits may be used to correct classification errors. In this paper we investigate the potential of error correcting codes for text categorization with many categories. We extend the work in [4] by a simulated annealing algorithm for code generation and optimization of Kullback-Leibler (KL) category distances within the code-words.

L. De Raedt and A. Siebes (Eds.): PKDD 2001, LNAI 2168, pp. 266–276, 2001.

For the first time, we apply the scheme to text categorization on several large text corpora with 42 to 109 categories and up to 11 million running words. The results are compared to 1-of-N coding (i.e. one SVM for each text category).

2 The Text Corpora

Table 1 gives an overview over the quantitative properties of the four different text corpora we used. The columns have the following meaning: '# categories' is the number of categories to distinguish. '# documents' is the total number of texts in the corpus. '# types' is the total number of different words, or, more precisely, different alphanumeric strings including punctuation marks to be found in a corpus. '# tokens' is the number of running words in a corpus. 'min length' is the minimal length of a document in running words (tokens). Shorter documents were discarded from the corpus. 'min # docs per cat' is the minimal number of documents of a specific category to be found in the corpus. All categories with fewer documents were discarded. So each corpus was divided into n disjoint sets by its n remaining categories.

Table 1. Quantitative properties of the text corpora

corpus	# categories	# documents	# types	# tokens	min length	min # docs per cat.
bz	64	4366	103665	992483	200	10
reuters	42	10216	54832	887357	15	10
sjm	109	36431	254440	11163970	300	80
wsj	101	41838	203638	11989608	100	50

We did not apply any preprocessing steps such as stemming, elimination of stop-words, etc. to the corpora. Our research on text coding for SVMs in [11] indicated, that exhaustive inclusion of full word-forms, numbers and punctuation improves recognition rates. We now describe the contents of the four corpora:

- *bz:* Texts from the German newspaper "Berliner Zeitung". These texts were drawn from the online archive of the newspaper[1]. The categorization task here was to recognize the **author** of the document. There are 64 different authors in total. We choose this corpus, because we already had promising results for authorship attribution (see [3]).
- *reuters:* We used the Reuters-21578 dataset[2] compiled by David Lewis and originally collected by the Carnegie group from the Reuters newswire in 1987. The task was to recognize the correct topic out of 42 selected topics. We already used a smaller subset of the *reuters* corpus in a pilot study [10].

[1] see `http://www.BerlinOnline.de`

[2] (obtainable at `http://www.research.att.com/~lewis/reuters21578.html`)

- *sjm:* News articles from 1991 "San Jose Mercury News" from the TIPSTER database vol. 3. The TIPSTER catalogs are available from the Linguistic Data Consortium (`http://www.ldc.upenn.edu`) Each news document contains a list of manually-assigned codes for categories. To exclude problems of overlapping categories, we chose only the first item from the list as a label of the document.
- *wsj:* Newspaper texts from the "Wall Street Journal", 1990-92, from the TIPSTER database vol. 2. A for the preceding corpus we chose the first item from the list of categories for each text.

Figure 1 shows the type-frequency spectra of the corpora. It displays frequency on the x-axis and number of words with a given frequency on the y-axis. A small slope of the spectrum is an indicator of standardized language use in a text corpus (see [11] for details). With respect to this criterion, language use in the *bz* corpus appears to by more creative than in the others.

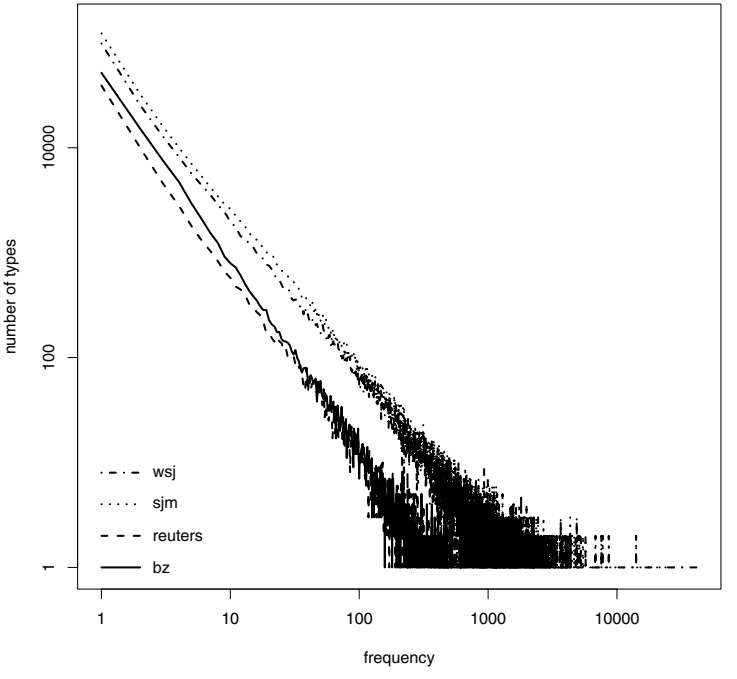

Fig. 1. Type-frequency spectra of the text corpora

3 Methods

3.1 Error Correcting Codes

Classification with error correcting codes can be seen "as a kind of communications problem in which the identity of the correct output class for a new example is being transmitted over a channel which consists of the input features, the training examples, and the learning algorithm" ([4, p266]). The classification of a new set of input features can no longer be determined from the output of one classifier. It is coded in a distributed representation of l outputs from *all* classifiers. Table 2 shows an example of an error correcting code for $n = 8$ classes with $l = 5$ code-words. The code-words are the columns of the table. Each classifier has to learn one of the code-words. This means, that the classifier should output a 1 for all input data belonging to one of the classes which are assigned 1 in the code-word of the classifier, and 0 in all other cases. The code for a specific class is to be found in the row of the table which is assigned to the class. The code length in bits is the number of code-words, 5 in our example. Note that at least $ceiling(log_2(n))$ code-words are required to distinguish n classes.

Table 2. Error correcting code for 8 classes with 5 code-words

	code-word				
class	1	2	3	4	5
1	0	0	0	1	1
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	0	0
5	1	0	0	1	0
6	1	1	0	1	1
7	1	1	1	0	1
8	1	1	1	1	0

Noise is introduced by the learning set, choice of features, and flaws of the learning algorithm. Noise may induce classification errors. But if there are more code-words than needed to distinguish n classes, i.e. $l >> log_2(n)$, we can use the additional bits to correct errors: If the output code does not match exactly one of the classes, take the class with minimal Hamming distance. If the minimum Hamming distance between class codes is d, we can in this way correct at least $\frac{d-1}{2}$ single bit errors (see [4, p.266]). It is therefore important to use codes with maximized Hamming distance for the classes.

There are several potential advantages in this approach: The number of required classifiers increases with $\mathcal{O}(log_2(n))$ only, and additional bits can be used for error correction.

We did not use the optimization criteria given by Dietterich and Bakiri to find optimal codes. Instead, we used simulated annealing to optimize a mix of Hamming distances of class codes, and of code-words. For a corpus with n

Table 3. Minimal Hamming distances for the *sjm*-codes

# code words	min Hamming distance categories	code-words	KL-distance random	optimized
54	21	48	4.080	4.054
81	34	46	4.085	4.059
109	47	43	4.084	4.066
164	75	40	4.084	4.072

categories to distinguish, we generated error correcting codes of four different sizes: $n \cdot 0.5$, $n \cdot 0.75$, $n \cdot 1.0$, and $n \cdot 1.5$, bits.

See Table 3 for the Hamming distances of a set of codes we used to categorize the *sjm*-corpus. For larger numbers of code-words, i.e. increasing l, the hamming distances grow. The reason is that the number of class codes to be generated remains constant ($n = 109$) and therefore we have more degrees of freedom to place the bits in the class codes. The Hamming distances of code-words decrease with increasing l. Here, we have constant length of code-words, but increasing numbers of them. Therefore we have decreasing degrees of freedom to design the code-words.

3.2 Optimized Kullback-Leibler Distance

The mapping of corpus categories onto error correcting codes has been arbitrary so far. Remember that each SVM in the classifier system has to implement categorization according to one code-*word*, i.e. a special column of the code matrix like the one shown in Table 2. This means that it has to recognize documents from each category labeled 1 in that column, and to reject all categories labeled 0.

Our hypothesis was that a reordering of classes such that more similar classes were grouped together in each of the code-words should improve the performance. We used the Kullback-Leibler test [6, p.57] to compute the distance $KLD(p_1, p_2)$ between two categories p_1 and p_2 as shown in equation 1.

$$KLD(p_1, p_2) = \sum_{i=1}^{n} p_1(i) \log(\frac{p_1(i)}{q(i)}) + \sum_{i=1}^{n} p_2(i) \log(\frac{p_2(i)}{q(i)}) \quad (1)$$

$$q(i) = \frac{p_1(i) + p_2(i)}{2}$$

The two categories are represented by their word-frequency vectors p_1 and p_2 of length n. These vectors contain the frequencies of all types in the corpus, but the counts are restricted to documents which belong to the category in question. Therefore some values may be equal to 0.

The Kullback-Leibler Distance of a whole Matrix M of error-correcting codes is now computed by comparing the categories with label 1 against each other

and also the categories with label 0. Let M_1 be the set of text categories which are labeled 1 in M and M_0 the set of text categories which are labeled 0 in M. Then we can define the Kullback-Leibler Distance of M as

$$KLD(M) = \frac{\sum_{p_i,p_j \in M_0} KLD(p_i,p_j)}{l_0 * (l_0 - 1) * 0.5} + \frac{\sum_{p_i,p_j \in M_1} KLD(p_i,p_j)}{l_1 * (l_1 - 1) * 0.5} \quad (2)$$

l_0 is the number of elements in M_0, l_1 is the number of elements in M_1.

We tested optimization of $KLD(M)$ **after** generation of the code Matrix M, as well as simultaneous optimization of Hamming distance and Kullback-Leibler distances. In this paper we only report results on the simultaneous optimization, because the performance was better in general. See Table 3 for KLD values of the *sjm* corpus.

3.3 Support Vector Machines

Support Vector Machines (SVM) recently gained popularity in the learning community [14]. In its simplest linear form, an SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum interclass distance, the *margin*.

The SVM can be extended to nonlinear models by mapping the input space into a very high-dimensional feature space chosen a priori. In this space the optimal separating hyperplane is constructed [14, p.421].

The distinctive advantage of the SVM for text categorization is its ability to process many thousand different inputs. This opens the opportunity to use all *words* in a text directly as features. For each word w_i the number of times of occurrence is recorded. Joachims [9] and also we [11] used the SVM for the classification of text into different topic categories. Dumais et al. [5] use linear SVM for text categorization because they are both accurate and fast. They are 35 times faster to train than the next most accurate (a decision tree) of the tested classifiers. They apply SVM to the *reuters* collection, e-mails and web pages.

3.4 Transformations of Frequency Vectors and Kernel Functions

The mapping of text to the SVM input space consists of three parts. First the type-frequencies (i.e. number of occurrences of words) are transformed by a bijective mapping. The resulting vector is multiplied by a vector of importance weights, and is finally normalized to unit length. We used those transformations that yielded the best results in [11].

We define the vector of logarithmic type frequencies of document d_i by

$$\mathbf{l}_i \quad = \quad \Big(\log\big(1 + f(w_1, d_i)\big), \ldots, \log\big(1 + f(w_n, d_i)\big) \Big), \quad (3)$$

where $f(w_k, d_i)$ is the frequency of occurrence of term w_k in document d_i, and n is the number of different terms in all documents of the collection. Logarithmic frequencies are combined with different importance weights. They are normalized with respect to L_2.

Importance weights can be used to quantify how specific a given type is to the documents of a text collection. A type which is evenly distributed across the document collection should be given a low importance weight because it is judged to be less specific for the documents it occurs in. A type which is used in only a few documents should be given a high importance weight. Redundancy quantifies the skewness of a probability distribution, and r_k is a measure of how much the distribution of a term w_k in the various documents deviates from the uniform distribution.

We therefore consider the empirical distribution of a type over the documents in the collection and define the importance weight of type w_k by

$$r_k \quad = \quad \log N + \sum_{i=1}^{N} \frac{f(w_k, d_i)}{f(w_k)} \log \frac{f(w_k, d_i)}{f(w_k)}, \tag{4}$$

where N is the number of documents in the collection. This yields a vector of importance weights for the whole document collection:

$$\mathbf{r} \quad = \quad (r_1, \ldots, r_n). \tag{5}$$

The advantage of redundancy over inverse document frequency is that it does not simply count the documents a type occurs in but takes into account the frequencies of occurrence in each of the document. The difference between redundancy and idf is larger for longer documents.

From the standpoint of the SVM learning algorithm the best normalization rule is the L_2-normalization because it yields the best error bounds. L_2-normalization has been used by Joachims [9] and Dumais et al. [5]. So the complete frequency transformation we used is defined as

$$\mathbf{x}_i \quad = \quad \frac{\mathbf{l}_i * \mathbf{r}}{\|\mathbf{l}_i * \mathbf{r}\|_{L_2}}$$

We used only the linear kernel function $K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x} \cdot \boldsymbol{x}'$ of the SVM, because increased computing times prevented tests with nonlinear kernels. Furthermore linear kernels performed very well in our pilot study [10].

4 Performance Measures

We applied performance measures which are widely used in information retrieval, because we think they are more adequate to text categorization than plain error rates.

4.1 Definitions

Precision (equation 6) is the percentage of documents in the target category i of all those documents which are categorized (perhaps wrongly) as category i.

$$prc(i) = 100 * \frac{c_t(i)}{c_t(i) + e_o(i)}, \tag{6}$$

$$prc(i) = 0 \text{ if } c_t(i) = 0$$

Recall (equation 7) is the percentage of documents in target category i which are recognized correctly.

$$rec(i) = 100 * \frac{c_t(i)}{c_t(i) + e_t(i)} \tag{7}$$

$c_t(i)$ correctly categorized documents of target class
$e_t(i)$ wrongly categorized documents of target class
$c_o(i)$ correctly categorized documents of other classes
$e_o(i)$ wrongly categorized documents of other classes

Since *precision* and *recall* may be inadequate to measure the performance on very small categories, we also computed a performance measure combined from *precision* and *recall*. It is called the *F-measure* (see [12, p.269]):

$$F(i) = \left(\alpha \frac{1}{prc(i)} + (1 - \alpha) \frac{1}{rec(i)} \right)^{-1} \tag{8}$$

α is a weighting factor. In this paper we set $\alpha = 0.5$, because we wanted to put equal importance on both *precision* and *recall*. Thus we get $F(i) = 2 \cdot \frac{prc(i) \cdot rec(i)}{prc(i) + rec(i)}$ The mean performance over all N categories of a corpus is

$$prc = \sum_{i=1}^{N} p_t(i) \cdot prc(i) \tag{9}$$

$$rec = \sum_{i=1}^{N} p_t(i) \cdot rec(i) \tag{10}$$

$$F = \sum_{i=1}^{N} p_t(i) \cdot F(i) \tag{11}$$

$p_t(i)$ is the probability of occurrence of category i in the corpus. We determined $p_t(i)$ as the fraction of documents of category i in all documents of the corpus. Weighted in this way, prc, rec, and F again can vary between 0% and 100%.

4.2 Results

To exploit the training set S_0 of each corpus in a better way, we used the following *cross-testing* procedure. S_0 was randomly divided into 3 subsets S_1, S_2, S_3 of nearly equal size. Then three different SVM classification systems were determined using $S_0 \setminus S_i$ as training set and S_i was used as test set. The numbers of correctly and wrongly classified documents were added up yielding an effective test set of all documents in S_0. The SVM implementation of Joachims (`http://ais.gmd.de/~thorsten/svm_light/`) was used for our experiments.

Table 4 shows the mean performance with respect to *precision* (equation 9), *recall* (equation 10), and *F-measure* (equation 11). The first four rows give the values for the error correcting codes of different bit length (Sect. 3.1). The last row gives the value of the corresponding 1-of-N codes. The best values are printed in bold font. Table 5 shows the corresponding values for error correcting codes with optimized KL-distance.

In both tables the 1-of-N coding has better *precision* for corpora *bz* and *reuters* and better *recall* for for corpora *sjm* and *wsj*. The combined *F-measure* however is better for error correcting codes on all four corpora.

Comparing tables with and without KL-optimization, there is no clear tendency. It seems that for corpora *bz* and *sjm* there generally is a slight improvement. A slight degradation can be seen in most cases of *reuters* and *wsj*.

Table 6 shows the percentage of categories on which error correcting codes perform better than the 1-of-N code with respect to *precision* (equation 9), *recall* (equation 10), and *F-measure* (equation 11). Table 7 shows the corresponding percentages for error correcting codes with optimized KL-distance.

We observe converse trends for *precision* and *recall*: Those corpora with many good results for error correcting codes on *precision* tend to perform bad on *recall* and vice versa.

Comparing tables with and without KL-optimization, there is a mixed tendency for *precision* and *recall*. Regarding the *F-measure*, there is improvement for most KL-optimized codes.

5 Conclusion

We investigated the potential of error correcting codes for text categorization on several large text collections. The error correcting code classifier was implemented as a combination of binary SVMs. We compared the results with those of a 1-of-N classifier. The main result is that long error correcting codes perform better than 1-of-N codes for a combination of *precision* and *recall* error measures.

We also investigated the effects of optimization of the Kullback-Leibler distance of the text categories grouped together in the code-words. The classification performance could be improved slightly for the corpora *bz* and *sjm*, but the latter result seems to be of mostly theoretical interest.

Acknowledgments. We would like to thank Thorsten Joachims and Tamas Horvath for inspiring discussions on this topic.

Table 4. Mean performance on all corpora - *no* KL-optimization

code length	bz	reuters	sjm	wsj
50%	80.4	91.5	58.3	**59.0**
75%	79.8	91.4	57.7	58.6
100%	79.8	91.4	58.6	58.2
150%	79.6	91.8	**59.8**	58.7
1-of-n	**89.0**	**93.2**	53.2	47.6

precision

code length	bz	reuters	sjm	wsj
50%	78.2	**91.8**	58.5	57.7
75%	79.2	91.5	59.0	58.5
100%	79.7	91.5	59.8	58.3
150%	**80.2**	91.7	59.6	58.8
1-of-n	60.7	89.1	**63.9**	**69.0**

recall

code length	bz	reuters	sjm	wsj
50%	76.9	**91.4**	55.5	55.7
75%	76.8	91.0	55.5	56.5
100%	77.2	91.0	**56.5**	56.5
150%	**78.0**	91.3	56.1	**57.0**
1-of-n	69.5	90.4	55.9	54.9

F-measure

Table 5. Mean performance on all corpora - *with* KL-optimization.

code length	bz	reuters	sjm	wsj
50%	80.6	91.2	58.2	58.3
75%	80.3	91.2	58.4	**58.8**
100%	79.6	91.3	58.3	58.3
150%	80.9	91.5	**58.9**	58.7
1-of-n	**89.0**	**93.2**	53.2	47.6

precision

code length	bz	reuters	sjm	wsj
50%	79.0	91.4	58.7	57.5
75%	80.0	91.3	59.3	58.6
100%	80.4	91.5	59.7	58.6
150%	**80.7**	**91.6**	59.9	58.6
1-of-n	60.7	89.1	**63.9**	**69.0**

recall

code length	bz	reuters	sjm	wsj
50%	77.1	91.0	55.8	55.7
75%	78.0	90.9	55.8	**56.8**
100%	78.2	91.1	56.4	56.6
150%	**78.5**	**91.3**	**56.7**	**56.8**
1-of-n	69.5	90.4	55.9	54.9

F-measure

References

1. Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proc. 17th International Conf. on Machine Learning*, pages 9–16. Morgan Kaufmann, San Francisco, CA, 2000.
2. Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Computational Learing Theory*, pages 35–46, 2000.
3. J. Diederich, K. Kindermann, E. Leopold, and G. Paass. Authorship attribution with support vector machines. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Techniques*, 2001. in press.
4. T.G. Dietterich and G. Bakiri. Solving multiclass learning via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.
5. S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *7th International Conference on Information and Knowledge Managment*, 1998.
6. Stephen E. Fienberg. *The analysis of cross-classified categorical data*. 1980.

Table 6. Percentage of categories on which EC-codes perform better than 1-of-N codes - *no* KL-optimization

code length	bz	reuters	sjm	wsj
50%	15.6	23.8	74.3	47.7
75%	20.3	14.3	83.2	60.6
100%	18.8	11.9	82.2	56.9
150%	17.2	23.8	89.1	63.3

precision

code length	bz	reuters	sjm	wsj
50%	84.4	66.7	17.8	44.0
75%	85.9	76.2	21.8	43.1
100%	90.6	83.3	16.8	48.6
150%	89.1	85.7	19.8	37.6

recall

code length	bz	reuters	sjm	wsj
50%	73.4	45.2	37.6	41.3
75%	75.0	52.4	43.6	45.9
100%	65.6	59.5	48.5	56.0
150%	75.0	69.0	60.4	46.8

F-measure

Table 7. Percentage of categories on which EC-codes perform better than 1-of-N codes - *with* KL-optimization

code length	bz	reuters	sjm	wsj
50%	20.3	14.3	73.3	56.0
75%	17.2	16.7	84.2	63.3
100%	14.1	19.0	83.2	58.7
150%	20.3	16.7	91.1	65.1

precision

code length	bz	reuters	sjm	wsj
50%	89.1	59.5	12.9	40.4
75%	90.6	78.6	15.8	40.4
100%	90.6	81.0	16.8	46.8
150%	95.3	92.9	12.9	51.4

recall

code length	bz	reuters	sjm	wsj
50%	79.7	31.0	35.6	45.0
75%	78.1	52.4	46.5	46.8
100%	75.0	61.9	47.5	57.8
150%	79.7	76.2	53.5	61.5

F-measure

7. Y. Guermeur, A. Eliseeff, and H. Paugam-Moisy. A new multi-class svm based on a uniform convergence result. In S.-I. Amari, C.L. Giles, M. Gori, and V. Piuri, editors, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN 2000*, pages IV–183 – IV–188, Los Alamitos, 2000. IEEE Computer Society.
8. C.-W. Hsu and C.J. Lin. A comparison on methods for multi-class support vector machines. unpulished manuscript, see `http://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.ps.gz`, April 2001.
9. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nedellec and C. Rouveirol, editors, *European Conference on Machine Learning (ECML)*, 1998.
10. J. Kindermann, E. Leopold, and G. Paass. Multi-class classification with error correcting codes. Technical report, GMD, Oct 2000. Beiträge zum Treffen der GI Fachgruppe 1.1.3 Maschinelles Lernen.
11. E. Leopold and J. Kindermann. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 2001. in press.
12. C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
13. J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.
14. V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.