Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain[⋆]

Eneldo Loza Mencía and Johannes Fürnkranz

Knowledge Engineering Group
Technische Universität Darmstadt
{eneldo,juffi}@ke.tu-darmstadt.de

Abstract In this paper we applied multilabel classification algorithms to the EUR-Lex database of legal documents of the European Union. On this document collection, we studied three different multilabel classification problems, the largest being the categorization into the EUROVOC concept hierarchy with almost 4000 classes. We evaluated three algorithms: (i) the binary relevance approach which independently trains one classifier per label; (ii) the multiclass multilabel perceptron algorithm, which respects dependencies between the base classifiers; and (iii) the multilabel pairwise perceptron algorithm, which trains one classifier for each pair of labels. All algorithms use the simple but very efficient perceptron algorithm as the underlying classifier, which makes them very suitable for large-scale multilabel classification problems. The main challenge we had to face was that the almost 8,000,000 perceptrons that had to be trained in the pairwise setting could no longer be stored in memory. We solve this problem by resorting to the dual representation of the perceptron, which makes the pairwise approach feasible for problems of this size. The results on the EUR-Lex database confirm the good predictive performance of the pairwise approach and demonstrates the feasibility of this approach for large-scale tasks.

Key words: text classification, multilabel classification, legal documents, EUR-Lex database, learning by pairwise comparison

1 Introduction

The EUR-Lex text collection is a collection of documents about European Union law. It contains many several different types of documents, including treaties, legislation, case-law and legislative proposals, which are indexed according to several orthogonal categorization schemes to allow for multiple search facilities. The most important categorization is provided by the EUROVOC descriptors,

[⋆] This is the authors' version of the work from www.ke.tu-darmstadt.de. The original publication is available at www.springerlink.com, DOI: 10.1007/978-3-642-12837-0_11, and appeared in E. Francesconi et al. (Eds.): *Semantic Processing of Legal Texts*, LNAI 6036, pp. 192-215, 2010, www.springerlink.com/content/63251w4h540l2r24/.

which is a topic hierarchy with almost 4000 categories regarding different aspects of European law.

This document collection provides an excellent opportunity to study text classification techniques for several reasons:

- it contains multiple classifications of the same documents, making it possible to analyze the effects of different classification properties using the same underlying reference data without resorting to artificial or manipulated classifications,
- the overwhelming number of produced documents make the legal domain a very attractive field for employing supportive automated solutions and therefore a machine learning scenario in step with actual practice,
- the documents are available in several European languages and are hence very interesting e.g. for the wide field of multi- and cross-lingual text classification,
- and, finally, the data is freely accessible (at `http://eur-lex.europa.eu/`)

In this paper, we make a first step towards analyzing this database by applying multilabel classification techniques on three of its categorization schemes. The database is a very challenging multilabel scenario due to the high number of possible labels (up to 4000), which, for example, exceeds the number of labels in the REUTERS databases by one order of magnitude. The EUR-Lex dataset is now publicly available under `http://www.ke.tu-darmstadt.de/resources/eurlex/`.

We evaluated three methods on this task:

- the conventional binary relevance approach (BR), which trains one binary classifier per label
- the *multilabel multiclass perceptron* (MMP), which also trains one classifier per label but does not treat them independently, instead it tries to minimize a ranking loss function of the entire ensemble [3]
- the *multilabel pairwise perceptron* (MLPP), which trains one classifier for each pair of classes [12]

Previous work on using these algorithms for text categorization [12] has shown that the MLPP algorithm outperforms the other two algorithms, while being slightly more expensive in training (by a factor that corresponds to the average number of labels for each example). However, another key disadvantage of the MLPP algorithm is its need for storing one classifier for each pair of classes. For the EUROVOC categorization, this results in almost 8,000,000 perceptrons, which would make it impossible to solve this task in main memory.

To solve this problem, we introduce and analyze a novel variant that addresses this problem by representing the perceptron in its dual form, i.e. the perceptrons are formulated as a combination of the documents that were used during training instead of explicitly as a linear hyperplane. This reduces the dependence on the number of classes and therefore allows the *Dual MLPP* algorithm to handle the tasks in the EUR-Lex database.

Originally, the MLPP accepts a multilabel information but only outputs a ranking over all possible labels, following [3] and their MMP algorithm. In order to find a delimiter between relevant and irrelevant labels within a provided ranking of the labels, we have recently introduced the idea of using an artificial label that encodes the boundary between relevant and irrelevant labels for each example [2], which has also been successfully applied to the Reuters-RCV1 text categorization task [6], a large collection of news texts. This approach was adapted to work with the dual variant and we present first results in this paper. However, we will focus our analysis on the produced ranking. There are three reasons for this: (i) the MMP, to which we directly compare, and the pairwise method naturally provide such a ranking, (ii) the ranking allows to evaluate the performance differences on a finer scale, (iii) our key motivation is to study the scalability of these approaches which is determined by the rankings, and (iv) although several different thresholding techniques exist that can be applied to the rankings produced by both MMP and MLPP (a good overview is provided in [20]), it was not the intention of this work to provide a comparison between them.

The outline of the paper is as follows: We start with a presentation of the EUR-Lex respository and the datasets that we derived from it (Section 2). Section 3 briefly recapitulates the algorithms that we study, followed by the presentation of the dual version of the MLPP classifier (Section 4). In Section 5, we compare the computational complexity of all approaches, and present the experimental results in Section 6.

2 The EUR-Lex Repository

The EUR-Lex/CELEX (Communitatis Europeae LEX) Site[1] provides a freely accessible repository for European Union law texts. The documents include the official Journal of the European Union, treaties, international agreements, legislation in force, legislation in preparation, case-law and parliamentary questions. They are available in most of the languages of the EU, and in the HTML and PDF format. We retrieved the HTML versions with bibliographic notes recursively from all (non empty) documents in the English version of the *Directory of Community legislation in force*[2], in total 19,348 documents. Only documents related to secondary law (in contrast to primary law, the constitutional treaties of the European Union) and international agreements are included in this repository. The legal form of the included acts are mostly *decisions* (8,917 documents), *regulations* (5,706), *directives* (1,898) and *agreements* (1,597). This version of the dataset differs slightly from that presented in previous works [14, 13], which still contained 19,596 documents. Some empty documents that were missed in the previous version and all corrigendums (they contained the same standard text except for one document since they were concerned with translations of the

[1] `http://eur-lex.europa.eu`

[2] `http://eur-lex.europa.eu/en/legis/index.htm`

Title and reference

Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs

Classifications

EUROVOC descriptor
- *data-processing law, computer piracy, copyright, software, approximation of laws*

Directory code
- 17.20.00.00 *Law relating to undertakings /* Intellectual property law

Subject matter
- *Internal market, Industrial and commercial property*

Text

COUNCIL DIRECTIVE of 14 May 1991 on the legal protection of computer programs (91/250/EEC)

THE COUNCIL OF THE EUROPEAN COMMUNITIES,

Having regard to the Treaty establishing the European Economic Community and in particular Article 100a thereof,

Having regard to the proposal from the Commission (1),

In cooperation with the European Parliament (2),

. . .

Figure 1. Excerpt of a EUR-Lex sample document with the CELEX ID 31991L0250. The original document contains more meta-information. We trained our classifiers to predict the EUROVOC descriptors, the directory code and the subject matters based on the text of the document.

law into other languages than English) have been removed. The updated version can be found under `http://www.ke.tu-darmstadt.de/resources/eurlex/`.

The bibliographic notes of the documents contain information such as dates of effect and validity, authors, relationships to other documents and classifications. The classifications include the assignment to several EUROVOC descriptors, directory codes and subject matters, hence all classifications are multilabel ones. EUROVOC is a multilingual thesaurus providing a controlled vocabulary for European Institutions[3]. Documents in the documentation systems of the EU are indexed using this thesaurus. The directory codes are classes of the official classification hierarchy of the *Directory of Community legislation in force*. It contains 20 chapter headings with up to four sub-division levels.

The high number of 3,956 different EUROVOC descriptors were identified in the retrieved documents, each document is associated to 5.31 descriptors on average. In contrast there are only 201 different subject matters appearing in the dataset, with a mean of 2.21 labels per document, and 410 different directory

[3] `http://europa.eu/eurovoc/`

Table 1. Statistics of datasets. The attribute number in parenthesis denotes the actual used number of features, i.e. for *scene* and *yeast* the number of features after adding the pairwise products and for the text collections the amount after feature selection. *Label density* indicates the average number of labels per instance d relative to the total number of classes n, and *distinct* counts the distinct label-sets found in the dataset $|\{P_i \mid i = 0 \ldots m\}|$.

dataset name	#classes n	avg. label-set size d	density $\frac{d}{n}$	distinct
EUR-Lex *subject matter*	201	2.213	1.101 %	2540
EUR-Lex *directory code*	410	1.292	0.315 %	1615
EUR-Lex *EUROVOC*	3956	5.310	0.134 %	16467

codes, with a label set size of on average 1.29. Note that for the directory codes we used only the assignment to the leaf category as the parent nodes can be deduced from the leaf node assignment. For the document in Figure 1 this would mean a set of labels of $\{17.20\}$ instead of $\{17, 17.20\}$. An overview of the properties of the different views on the dataset are given in Table 1.

Figure 1 shows an excerpt of a sample document with all information that has not been used removed. The full document can be viewed at `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:EN:NOT`. We extracted the text body from the HTML documents, excluding HTML tags, bibliographic notes or other additional information that could distort the results. The text was tokenized into lower case, stop words were excluded, and the Porter stemmer algorithm was applied. In order to perform cross validation, the instances were randomly distributed into ten folds. The tokens were projected for each fold into the vector space model using the common TF-IDF term weighting.In order to reduce the memory requirements, of the approx. 200,000 resulting features we selected the first 5,000 ordered by their document frequency. This feature selection method is very simple and efficient and independent from class assignments, although it performs comparably to more sophisticated methods using chi-square or information gain computation [21]. In order to ensure that no information from the test set enters the training phase, the TF-IDF transformation and the feature selection were conducted only on the training sets of the ten cross-validation splits.

The EUROVOC thesaurus has already been presented as set of classes for a multilabel classification task in [18]. The authors use several refined text and linguistig processing techniques and statistical computations in order to return a list of associated lemmas from the EUROVOC thesaurus for documents of the EU. However, their results are not comparable since a different resource was used for the documents, resulting also in a different number of EUROVOC descriptors used, namely around 2900

3 Preliminaries

We represent an instance or object as a vector $\bar{\mathrm{x}} = (x_1, \ldots, x_N)$ in a feature space $\mathcal{X} \subseteq \mathbb{R}^N$. Each instance $\bar{\mathrm{x}}_i$ is assigned to a set of relevant labels P_i, a subset of the n possible classes $\mathcal{L} = \{\lambda_1, \ldots, \lambda_n\}$. For multilabel problems, the cardinality $|P_i|$ of the label sets is not restricted, whereas for binary problems $|P_i| = 1$. For the sake of simplicity we use the following notation for the binary case: we define $\mathcal{L} = \{1, -1\}$ as the set of classes so that each object $\bar{\mathrm{x}}_i$ is assigned to a $\lambda_i \in \{1, -1\}$, $P_i = \{\lambda_i\}$.

3.1 Ranking Loss Functions

In order to evaluate the predicted ranking we use different *ranking losses*. The losses are computed comparing the ranking with the true set of relevant classes, each of them focusing on different aspects. For a given instance $\bar{\mathrm{x}}$, a relevant label set P, a negative label set $N = \mathcal{L} \backslash P$ and a given predicted ranking $r : \mathcal{L} \rightarrow \{1 \ldots n\}$, with $r(\lambda)$ returning the position of class λ in the ranking, the different loss functions are computed as follows:

- The is-error loss (IsErr) determines whether $r(\lambda) < r(\lambda')$ for all relevant classes $\lambda \in P$ and all irrelevant classes $\lambda' \in \overline{P}$. It returns 0 for a completely correct, *perfect ranking*, and 1 for an incorrect ranking, irrespective of 'how wrong' the ranking is.
- The one-error loss (OneErr) is 1 if the top class in the ranking is not a relevant class, otherwise 0 if the top class is relevant, independently of the positions of the remaining relevant classes.
- The ranking loss (RankLoss) returns the number of pairs of labels which are not correctly ordered normalized by the total number of possible pairs. As IsErr, it is 0 for a perfect ranking, but it additionally differentiates between different degrees of errors.

$$E \overset{\text{def}}{=} \{(\lambda, \lambda') \mid r(\lambda) > r(\lambda')\} \subseteq P \times N \qquad \delta_{\text{RankLoss}} \overset{\text{def}}{=} \frac{|E|}{|P||N|} \tag{1}$$

- The margin (Margin) loss returns the number of positions between the worst ranked positive and the best ranked negative classes. This is directly related to the number of wrongly ranked classes, i.e. the positive classes that are ordered below a negative class, or vice versa. We denote this set by F.

$$F \overset{\text{def}}{=} \{\lambda \in P \mid r(\lambda) > r(\lambda'), \lambda' \in N\} \cup \{\lambda' \in N \mid r(\lambda) > r(\lambda'), \lambda \in P\} \tag{2}$$

$$\delta_{\text{Margin}} \overset{\text{def}}{=} \max(0, \max\{r(\lambda) \mid \lambda \in P\} - \min\{r(\lambda') \mid \lambda' \notin P\}) \tag{3}$$

- Average Precision (AvgP) is commonly used in Information Retrieval and computes for each relevant label the percentage of relevant labels among all labels that are ranked before it, and averages these percentages over all relevant labels. In order to bring this loss in line with the others so that an optimal ranking is 0, we revert the measure.

$$\delta_{\text{AvgP}} \overset{\text{def}}{=} 1 - \frac{1}{P} \sum_{\lambda \in P} \frac{|\{\lambda^* \in P \mid r(\lambda^*) \leq r(\lambda)\}|}{r(\lambda)} \tag{4}$$

3.2 Multilabel Evaluation Measures

There is no generally accepted procedure for evaluating multilabel classifications. Our approach is to consider a multilabel classification problem as a meta-classification problem where the task is to separate the set of possible labels into relevant labels and irrelevant labels. Let $\hat{P}_i$ denote the set of labels predicted by the multilabel classifier and $\hat{N}_i = \mathcal{L} \setminus \hat{P}_i$ the set of labels that are not predicted by the classifier for an instance $\bar{x}_i$. Thus, we can, for each individual instance $\bar{x}_i$, compute a two-by-two confusion matrix C_i of relevant/irrelevant vs. predicted/not predicted labels:

C_i	predicted	not predicted	
relevant	$\lvert P_i \cap \hat{P}_i\rvert$	$\lvert P_i \cap \hat{N}_i\rvert$	$\lvert P_i\rvert$
irrelevant	$\lvert N_i \cap \hat{P}_i\rvert$	$\lvert N_i \cap \hat{N}_i\rvert$	$\lvert N_i\rvert$
	$\lvert\hat{P}_i\rvert$	$\lvert\hat{N}_i\rvert$	$\lvert\mathcal{L}\rvert$

From such a confusion matrix C_i, we can compute several well-known measures:

– The *Hamming loss* (HamLoss) computes the percentage of labels that are misclassified, i.e., relevant labels that are not predicted or irrelevant labels that are predicted. This basically corresponds to the error in the confusion matrix.

$$\text{HamLoss}(C_i) \overset{\text{def}}{=} 1 - \frac{1}{|\mathcal{L}|} |\hat{P}_i \triangle P_i| \tag{5}$$

The operator $\triangle$ denotes the symmetric difference between two sets and is defined as $A \triangle B \overset{\text{def}}{=} (A \setminus B) \cup (B \setminus A)$, i.e. $\hat{P}_i \triangle P_i$ has all labels that only appear in one of the two sets.

– *Precision* (Prec) computes the percentage of predicted labels that are relevant, *recall* (Rec) computes the percentage of relevant labels that are predicted, and the F1-measure is the harmonic mean between the two.

$$\text{Prec}(C_i) \overset{\text{def}}{=} \frac{|\hat{P}_i \cap P_i|}{|\hat{P}_i|} \qquad \text{Rec}(C_i) \overset{\text{def}}{=} \frac{|\hat{P}_i \cap P_i|}{|P_i|} \tag{6}$$

$$\text{F1}(C_i) \overset{\text{def}}{=} \frac{2}{\frac{1}{\text{Rec}(C_i)} + \frac{1}{\text{Prec}(C_i)}} = \frac{2\,\text{Rec}(C_i)\,\text{Prec}(C_i)}{\text{Rec}(C_i) + \text{Prec}(C_i)} \tag{7}$$

To average these values, we compute a micro-average over all values in a test set, i.e., we add up the confusion matrices C_i for examples in the test set and compute the measure from the resulting confusion matrix. Thus, for any given measure f, the average is computed as:

$$f_{\text{avg}} = f(\sum_{i=1}^{m} C_i) \tag{8}$$

To combine the results of the individual folds of a cross-validation, we average the estimates f^j_{avg} , $j = 1 \ldots q$ over all q folds.

3.3 Perceptrons

We use the simple but fast perceptrons as base classifiers [19]. Such as Support Vector Machines (SVM), their decision function describes a hyperplane that divides the N-dimensional space into two halves corresponding to positive and negative examples. We use a version that works without learning rate and threshold:

$$o'(\bar{\mathrm{x}}) = sgn(\bar{\mathrm{x}} \cdot \bar{\mathrm{w}}) \tag{9}$$

with the internal weight vector $\bar{\mathrm{w}}$ and $sgn(t) = 1$ for $t \geq 0$ and -1 otherwise. Two sets of points are called *linearly separable* if there exists a *separating hyperplane* between them. If this is the case and the examples are seen iteratively, the following update rule provably finds a separating hyperplane (cf., e.g., [1]).

$$\alpha_i = (\lambda_i - o'(\bar{\mathrm{x}}_i)) \qquad \bar{\mathrm{w}}_{i+1} = \bar{\mathrm{w}}_i + \alpha_i \bar{\mathrm{x}}_i \tag{10}$$

It is important to see that the final weight vector can also be represented as linear combination of the training examples:

$$\bar{\mathrm{w}} = \sum_{i=1}^{m} \alpha_i \bar{\mathrm{x}}_i \qquad o'(\bar{\mathrm{x}}) = sgn(\sum_{i=1}^{m} \alpha_i \cdot \bar{\mathrm{x}}_i \bar{\mathrm{x}}) \tag{11}$$

assuming m to be the number of seen training examples and $\alpha_i \in \{-1, 0, 1\}$. The perceptron can hence be coded implicitly as a vector of instance weights $\alpha = (\alpha_1, \ldots, \alpha_m)$ instead of explicitly as a vector of feature weights. This representation is denominated the dual form and is crucial for developing the memory efficient variant in Section 4.

The main reason for choosing perceptrons as our base classifier is because, contrary to SVMs, they can be trained efficiently in an incremental setting, which makes them particularly well-suited for large-scale classification problems such as the Reuters-RCV1 benchmark [9], without forfeiting too much accuracy. For this reason, the perceptron has recently received increased attention (e.g. [4, 8]).

```
Require: Training example pair $(\bar{x}, P)$, perceptrons $\bar{w}_1, \dots, \bar{w}_n$
 1: calculate $\bar{x}\bar{w}_1, \dots, \bar{x}\bar{w}_n$, loss $\delta$
 2: if $\delta > 0$ then                                   ▷ only if ranking is not perfect
 3:     calculate error sets $E$, $F$
 4:     for each $\lambda \in F$ do $\tau_\lambda \leftarrow 0$, $\sigma \leftarrow 0$          ▷ initialize $\tau$'s, $\sigma$
 5:     for each $(\lambda, \lambda') \in E$ do
 6:         $p \leftarrow$ PENALTY$(\bar{x}\bar{w}_1, \dots, \bar{x}\bar{w}_n)$
 7:         $\tau_\lambda \leftarrow \tau_\lambda + p$                          ▷ push up pos. classes
 8:         $\tau_{\lambda'} \leftarrow \tau_{\lambda'} - p$                        ▷ push down neg. classes
 9:         $\sigma \leftarrow \sigma + p$                              ▷ for normalization
10:     for each $\lambda \in F$ do
11:         $\bar{w}_\lambda \leftarrow \bar{w}_\lambda + \delta \frac{\tau_\lambda}{\sigma} \cdot \bar{x}$                  ▷ update perceptrons
12: return $\bar{w}_1 \dots \bar{w}_n$                             ▷ return updated perceptrons
```

Figure 2. Pseudocode of the training method of the MMP algorithm

3.4 Binary Relevance Ranking

In the binary relevance (BR) or one-against-all (OAA) method, a multilabel training set with n possible classes is decomposed into n binary training sets of the same size that are then used to train n binary classifiers. So for each pair $(\bar{x}_i, P_i)$ in the original training set n different pairs of instances and binary class assignments $(\bar{x}_i, \lambda_{i_j})$ with $j = 1 \dots n$ are generated setting $\lambda_{i_j} = 1$ if $\lambda_j \in P_i$ and $\lambda_{i_j} = -1$ otherwise. Supposing we use perceptrons as base learners, n different o'_j classifiers are trained in order to determine the relevance of λ_j. In consequence, the combined prediction of the binary relevance classifier for an instance $\bar{x}$ would be the set $\{\lambda_j \mid o'_j(\bar{x}) = 1\}$. If, in contrast, we desire a class ranking, we simply use the inner products and obtain a vector $\bar{o}(\bar{x}) = (\bar{x}\bar{w}_1, \dots, \bar{x}\bar{w}_n)$. Ties are broken randomly to not favor any particular class.

3.5 Multiclass Multilabel Perceptrons

MMPs were proposed as an extension of the one-against-all algorithm with perceptrons as base learners [3]. Just as in binary relevance, one perceptron is trained for each class, and the prediction is calculated via the inner products. The difference lies in the update method: while in the binary relevance method all perceptrons are trained independently to return a value greater or smaller than zero, depending on the relevance of the classes for a certain instance, MMPs are trained to produce a good ranking so that the relevant classes are all ranked above the irrelevant classes. The perceptrons therefore cannot be trained independently, considering that the target value for each perceptron depends strongly on the values returned by the other perceptrons.

The pseudocode in Fig. 2 describes the MMP training algorithm. In summary, for each new training example the MMP first computes the predicted ranking, and if there is an error according to the chosen loss function δ (e.g. any of the

```
Require: Training example pair $(\bar{x}, P)$,
    perceptrons $\{\bar{w}_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$
1: for each $(\lambda_u, \lambda_v) \in P \times N$ do
2:     if $u < v$ then
3:         $\bar{w}_{u,v} \leftarrow$ TrainPerceptron$(\bar{w}_{u,v}, (\bar{x}, 1))$        ▷ train as positive example
4:     else
5:         $\bar{w}_{v,u} \leftarrow$ TrainPerceptron$(\bar{w}_{v,u}, (\bar{x}, -1))$       ▷ train as negative example
6: return $\{\bar{w}_{u,v} \mid u < v, \lambda_u, \lambda_v \in \mathcal{L}\}$               ▷ updated perceptrons
```

Figure 3. Pseudocode of the training method of the MLPP algorithm.

losses in Sec. 3.1), it computes the set of wrongly ordered class pairs in the ranking and applies to each class in this set a penalty score according to a freely selectable function. We chose the uniform update method, where each pair in E receives the same score [3]. Please refer to [3] and [12] for a more detailed description of the algorithm.

3.6 Multilabel Pairwise Perceptrons

In the pairwise binarization method, one classifier is trained for each pair of classes, i.e., a problem with n different classes is decomposed into $\frac{n(n-1)}{2}$ smaller subproblems. For each pair of classes (λ_u, λ_v), only examples belonging to either λ_u or λ_v are used to train the corresponding classifier $o'_{u,v}$. All other examples are ignored. In the multilabel case, an example is added to the training set for classifier $o'_{u,v}$ if u is a relevant class and v is an irrelevant class, i.e., $(u, v) \in P \times N$ (cf. Figure 4). We will typically assume $u < v$, and training examples of class u will receive a training signal of $+1$, whereas training examples of class v will be classified with -1. Figure 3 shows the training algorithm in pseudocode. Of course MLPPs can also be trained incrementally.

In order to return a class ranking we use a simple voting strategy, known as *max-wins*. Given a test instance, each perceptron delivers a prediction for one of its two classes. This prediction is decoded into a vote for this particular class. After the evaluation of all $\frac{n(n-1)}{2}$ perceptrons the classes are ordered according to their sum of votes. Ties are broken randomly in our case.

Figure 5 shows a possible result of classifying the sample instance of Figure 4. Perceptron $o'_{1,5}$ predicts (correctly) the first class, consequently λ_1 receives one vote and class λ_5 zero (denoted by $o'_{1,5} = 1$ in the first and $o'_{5,1} = -1$ in the last row). All 10 perceptrons (the values in the upper right corner can be deduced due to the symmetry property of the perceptrons) are evaluated though only six are 'qualified' since they were trained with the original example.

This may be disturbing at first sight since many 'unqualified' perceptrons are involved in the voting process: $o'_{1,2}$ is asked for instance though it cannot know anything relevant in order to determine if $\bar{x}$ belongs to λ_1 or λ_2 since it was neither trained on this example nor on other examples belonging simultaneously to both classes (or to none of both). In the worst case the noisy votes concentrate

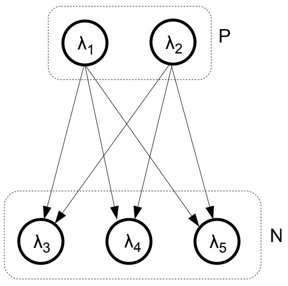

Figure 4. MLPP training: training example $\bar{x}$ belongs to $P = \{\lambda_1, \lambda_2\}$, $N = \{\lambda_3, \lambda_4, \lambda_5\}$ are the irrelevant classes, the arrows represent the trained perceptrons.

$o'_{1,2} = 1$	$o'_{2,1} = -1$	$o'_{3,1} = -1$	$o'_{4,1} = -1$	$o'_{5,1} = -1$
$o'_{1,3} = 1$	$o'_{2,3} = 1$	$o'_{3,2} = -1$	$o'_{4,2} = -1$	$o'_{5,2} = -1$
$o'_{1,4} = 1$	$o'_{2,4} = 1$	$o'_{3,4} = 1$	$o'_{4,3} = -1$	$o'_{5,3} = -1$
$o'_{1,5} = 1$	$o'_{2,5} = 1$	$o'_{3,5} = 1$	$o'_{4,5} = 1$	$o'_{5,4} = -1$
$v_1 = 4$	$v_2 = 3$	$v_3 = 2$	$v_4 = 1$	$v_5 = 0$

Figure 5. MLPP voting: an example $\bar{x}$ is classified by all 10 base perceptrons $o'_{u,v}, u \neq v\,,\ \lambda_u, \lambda_v \in \mathcal{L}$. Note the redundancy given by $o'_{u,v} = -o'_{v,u}$. The last line counts the positive outcomes for each class.

on a single negative class, which would lead to misclassifications. But note that any class can at most receive $n-1$ votes, so that in the extreme case when the qualified perceptrons all classify correctly and the unqualified ones concentrate on a single class, a positive class would still receive at least $n-|P|$ and a negative at most $n-|P|-1$ votes. Class λ_3 in Figure 5 is an example for this: It receives all possible noisy votes but still loses against the positive classes λ_1 and λ_2.

The pairwise binarization method is often regarded as superior to binary relevance because it profits from simpler decision boundaries in the subproblems [5, 7]. In the case of an equal class distribution, the subproblems have $\frac{2}{n}$ times the original size whereas binary relevance maintains the size. Typically, this goes hand in hand with an increase of the space where a separating hyperplane can be found. Particularly in the case of text classification the obtained benefit clearly exists. An evaluation of the pairwise approach on the Reuters-RCV1 corpus [9], which contains over 100 classes and 800,000 documents, showed a significant and substantial improvement over the MMP method [12]. This encourages us to apply the pairwise decomposition to the EUR-Lex database, with the main obstacle of the quadratic number of base classifier in relationship to the number of classes. Since this problem can not be coped for the present classifications in EUR-Lex we propose to reformulate the MLPP algorithm in the way described in Section 3.6.

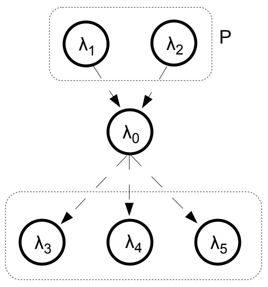

Figure 6. calibration: introducing virtual label λ_0 that separates P an N. Perceptrons $\bar{\mathrm{w}}_{1,0}, \bar{\mathrm{w}}_{2,0}, \bar{\mathrm{w}}_{0,3}, \bar{\mathrm{w}}_{0,4}, \bar{\mathrm{w}}_{0,5}$ are additionally trained.

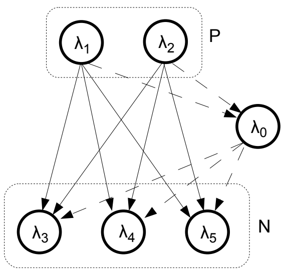

Figure 7. CMLPP training: the complete set of trained perceptrons.

Note that MLPP can potentially be used with any binary base learner. In particular it is possible to use advanced perceptron variants that especially consider the case of unbalanced classification problems [10, 4], since this is commonly the case for problems with a high number of classes [16]. In our opinion, this problem is not too severe for the pairwise decomposition, since it does not compare one class against the accumulation of all remaining examples, achieving in average a more balanced factor between positive and negative examples. Moreover, since we mainly evaluate the ranking quality, MMP and BR should not be discriminated by unbalanced classes.

3.7 Calibrated Label Ranking

To convert the resulting ranking of labels into a multilabel prediction, we use the *calibrated label ranking* approach [6]. This technique avoids the need for learning a threshold function for separating relevant from irrelevant labels, which is often performed as a post-processing phase after computing a ranking of all possible classes. The key idea is to introduce an artificial *calibration label* λ_0, which represents the split-point between relevant and irrelevant labels. Thus, it is assumed to be preferred over all irrelevant labels, but all relevant labels are preferred over λ_0. This introduction of an additional label during training is depicted in Figure 6, the combination with the normal pairwise base classifiers is shown in Figure 7.

As it turns out, the resulting n additional binary classifiers $\{\, o'_{0,u} \mid u = 1 \dots n\}$ are identical to the classifiers that are trained by the binary relevance approach. Thus, each classifier $o'_{0,u}$ is trained in a one-against-all fashion by using the whole dataset with $\{\, \bar{\mathrm{x}}_i \mid \lambda_u \in P_i\} \subseteq \mathcal{X}$ as positive examples and $\{\, \bar{\mathrm{x}}_i \mid \lambda_u \in N_i\} \subseteq \mathcal{X}$ as negative examples. At prediction time, we will thus get a ranking over $n+1$ labels (the n original labels plus the calibration label). Then, the projection of voting aggregation of pairwise perceptrons with a calibrated label to a multilabel output is quite straight-forward:

$$\hat{P} = \{\lambda \in \mathcal{L} \,|\, v(\lambda) > v(\lambda_0)\}$$

where $v(\lambda)$ is the amount of votes class λ has received.

We denote the MLPP algorithm adapted in order to support the calibration technique as CMLPP. This algorithm was again applied to the large Reuters-RCV1 corpus, outperforming the binary relevance and MMP approach [6].

4 Dual Multilabel Pairwise Perceptrons

With an increasing number of classes the required memory by the MLPP algorithm grows quadratically and even on modern computers with a large memory this problem becomes unsolvable for a high number of classes. For the EUROVOC classification, the use of MLPP would mean maintaining approximately 8,000,000 perceptrons in memory. In order to circumvent this obstacle we reformulate the MLPP ensemble of perceptrons in dual form as we did with one single perceptron in Equation 11. In contrast to MLPP, the training examples are thus required and have to be kept in memory in addition to the associated weights, as a base perceptron is now represented as $\bar{\mathrm{w}}_{u,v} = \sum_{i=1}^{m} \alpha_{u,v}^{t} \bar{\mathrm{x}}_i$. This makes an additional loop over the training examples inevitable every time a prediction is demanded. But fortunately it is not necessary to recompute all $\bar{\mathrm{x}}_i\bar{\mathrm{x}}$ for each base perceptron since we can reuse them by iterating over the training examples in the outer loop, as can be seen in the following equations:

$$\begin{aligned}
\bar{\mathrm{w}}_{1,2}\bar{\mathrm{x}} &= \alpha_{1,2}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,2}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,2}^{m}\bar{\mathrm{x}}_m\bar{\mathrm{x}} \\
\bar{\mathrm{w}}_{1,3}\bar{\mathrm{x}} &= \alpha_{1,3}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,3}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,3}^{m}\bar{\mathrm{x}}_m\bar{\mathrm{x}} \\
&\vdots \\
\bar{\mathrm{w}}_{1,n}\bar{\mathrm{x}} &= \alpha_{1,n}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,n}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,n}^{m}\bar{\mathrm{x}}_m\bar{\mathrm{x}} \\
\bar{\mathrm{w}}_{2,3}\bar{\mathrm{x}} &= \alpha_{2,3}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{2,3}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{2,3}^{m}\bar{\mathrm{x}}_m\bar{\mathrm{x}} \\
&\vdots
\end{aligned} \tag{12}$$

By advancing column by column it is not necessary to repeat the dot products computations, however it is necessary to store the intermediate values, as can also be seen in the pseudocode of the training and prediction phases in Figures 8 and 9. Note also that the algorithm preserves the property of being incrementally trainable. We denote this variant of training the pairwise perceptrons the *dual multilabel pairwise perceptrons* algorithm (DMLPP).

In addition to the savings in memory and run-time, analyzed in detail in Section 5, the dual representation allows for using the kernel trick, i.e. to replace the dot product by a kernel function, in order to be able to solve originally not linearly separable problems. However, this is not necessary in our case since text problems are in general linearly separable.

```
Require: New training example pair $(\bar{x}_m, P_m)$,
    training examples $\bar{x}_1 \ldots \bar{x}_{m-1}$, $P_1 \ldots P_{m-1}$,
    weights $\{\alpha^i_{u,v} \mid \lambda_u, \lambda_v \in \mathcal{L}, 0 < i < m\}$
1: for each $\bar{x}_i = \bar{x}_1 \ldots \bar{x}_{m-1}$ do        ▷ iterate over previous training examples
2:     $p_i \leftarrow \bar{x}_i \cdot \bar{x}_m$
3:     for each $(\lambda_u, \lambda_v) \in P_m \times N_m$ do        ▷ $\bar{x}_m$ only relevant for training these pairs
4:         if $\alpha^i_{u,v} \neq 0$ then
5:             $s_{u,v} \leftarrow s_{u,v} + \alpha^i_{u,v} \cdot p_t$        ▷ note that $s_{u,v} = -s_{v,u}$
6: for each $(\lambda_u, \lambda_v) \in P_m \times N_m$ do        ▷ update only concerned perceptrons
7:     if $s_{u,v} < 0$ then        ▷ and only if they misspredicted
8:         $\alpha^m_{u,v} \leftarrow 1$        ▷ note that $\alpha_{u,v} = -\alpha_{v,u}$
9: return $\{\alpha^m_{u,v} \mid (\lambda_u, \lambda_v) \in P \times N\}$        ▷ return new weights
```

Figure 8. Pseudocode of the training method of the DMLPP algorithm.

```
Require: example $\bar{x}$ for classification,
    training examples $\bar{x}_1 \ldots \bar{x}_{m-1}$, $P_1 \ldots P_{m-1}$,
    weights $\{\alpha^i_{u,v} \mid \lambda_u, \lambda_v \in \mathcal{L}, 0 < i < m\}$
1: for each $\bar{x}_i = \bar{x}_1 \ldots \bar{x}_{m-1}$ do        ▷ iterate over training examples
2:     $p \leftarrow \bar{x}_i \cdot \bar{x}$
3:     for each $(\lambda_u, \lambda_v) \in P_i \times N_i$ do        ▷ $\bar{x}_i$ was only be part of training these pairs
4:         if $\alpha^i_{u,v} \neq 0$ then        ▷ consider only if $\bar{x}$ is actually part of $\bar{w}_{u,v}$
5:             $s_{u,v} \leftarrow s_{u,v} + \alpha^i_{u,v} \cdot p$        ▷ add intermediate score to $\bar{w}_{u,v}\bar{x}$
6: for each $(\lambda_u, \lambda_v) \in \mathcal{L} \times \mathcal{L}$ do
7:     if $u \neq v \wedge s_{u,v} > 0$ then
8:         $v_u \leftarrow v_u + 1$        ▷ add up a vote for winning class $\lambda_u$
9: return voting $\bar{v} = (v_1, \ldots, v_{|\mathcal{L}|})$        ▷ return voting
```

Figure 9. Pseudocode of the prediction phase of the DMLPP algorithm.

Note also that the pseudocode needs to be slightly adapted when the DMLPP algorithm is trained in more than one epoch, i.e. the training set is presented to the learning algorithm more than once. It is sufficient to modify the assignment in line 8 in Figure 8 to an additive update $\alpha^m_{u,v} \leftarrow \alpha^m_{u,v} + 1$ for a revisited example $\bar{x}_m$. This setting is particularly interesting for the dual variant since, when the training set is not too big, memorizing the inner products can boost the subsequent epochs in a substantial way, making the algorithm interesting even if the number of classes is small.

4.1 Calibration

There exist two ways of adapting the calibration approach described in Section 3.7 for DMLPP: processing the additional subproblems internally or externally.

The first version trains the additional base classifiers also in dual form. However, we believe that this approach could decrease the advantage that DMLPP obtains through the *sparseness* of the pairwise decomposition.

Therefore, the second version considered simply trains an external (non-dual) binary relevance classifier (as described in Section 3.4) in parallel. During classification, the predictions of the base perceptrons of the BR classifier are incorporated in the voting process. We will denote this algorithm as DCMLPP.

5 Computational Complexity

The notation used in this section is the following: n denotes the number of possible classes, d the average number of relevant classes per instance in the training set, N the number of attributes and N' the average number of attributes not zero (size of the sparse representation of an instance), and m denotes the size of the training set. For each complexity we will give an upper bound O in Landau notation. We will indicate the runtime complexity in terms of real value additions and multiplications ignoring operations that have to be performed by all algorithms such as sorting or internal real value operations. Additionally, we will present the complexities per instance as all algorithms are incrementally trainable. We will also concentrate on the comparison between MLPP and the implicit representation DMLPP.

The MLPP algorithm has to keep $\frac{n(n-1)}{2}$ perceptrons, each with N weights in memory, hence we need $O(n^2N)$ memory. The DMLPP algorithm keeps the whole training set in memory, and additionally requires for each training example $\bar{\mathrm{x}}$ access to the weights of all class pairs $P \times N$. Furthermore, it has to intermediately store the resulting scores for each base perceptron during prediction, hence the complexity is $O(mdn + mN' + n^2) = O(m(dn + N') + n^2)$.[4] We can see that MLPP is applicable especially if the number of classes is low and the number of examples high, whereas DMLPP is suitable when the number of classes is high, however it does not handle huge training sets very well.

For processing one training example, $O(dn)$ dot products have to be computed by MLPP, one for each associated perceptron. Assuming that a dot product computation costs $O(N')$, we obtain a complexity of $O(dnN')$ per training example. Similarly, the DMLPP spends m dot product computations. In addition the summation of the scores costs $O(dn)$ per training instance, leading to $O(m(dn + N'))$ operations. It is obvious that MLPP has a clear advantage over DMLPP in terms of training time, unless n is of the order of magnitude of m or the model is trained over several epochs, as already outlined in the previous Section 4.

During prediction the MLPP evaluates all perceptrons, leading to $O(n^2N')$ computations. The dual variant again iterates over all training examples and associated weights, hence the complexity is $O(m(dn+N'))$. At this phase DMLPP

[4] Note that we do not estimate d as $O(n)$ since both values are not of the same order of magnitude in practice. For the same reason we distinguish between N and N' since particularly in text classification both values are not linked: a text document often turns out to employ around 100 different words whereas the size of the vocabulary of a the whole corpus can easily reach 100,000 words (although this number is normally reduced by feature selection).

Table 2. Computational complexity given in expected number of addition and multiplication operations. n: *#classes*, d: *avg. #labels per instance*, m: *#training examples*, N: *#attributes*, N': *#attributes≠0.*

	training time	testing time	memory requirement
MMP, BR	$O(nN')$	$O(nN')$	$O(nN)$
MLPP	$O(dnN')$	$O(n^2N')$	$O(n^2N)$
DMLPP	$O(m(dn+N'))$	$O(m(dn+N'))$	$O(m(dn+N')+n^2)$

benefits from the linear dependence of the number of classes in contrast to the quadratic relationship of the MLPP. Roughly speaking the breaking point when DMLPP is faster in prediction is approximately when the square of the number of classes is clearly greater than the number of training documents. We can find a similar trade-off for the memory requirements with the difference that the factor between sparse and total number of attributes becomes more important, leading earlier to the breaking point when the sparseness is high. A compilation of the analysis can be found in Table 2, together with the complexities of MMP and BR. A more detailed comparison between MMP and MLPP can be found in [12].

In summary, it can be stated that the dual form of the MLPP balances the relationship between training and prediction time by increasing training and decreasing prediction costs, and especially benefits from a decreased prediction time and memory savings when the number of classes is large. Thus, this technique addresses the main obstacle to applying the pairwise approach to problems with a large number of labels.

For the complexities of the calibrated variants of MLPP and DMLPP we can simply add the corresponding complexity of BR, at least if we consider the externally calibrated variant of DCMLPP.

6 Experiments

For the MMP algorithm we used the IsErr loss function and the uniform penalty function. This setting showed the best results in [3] on the RCV1 data set. The perceptrons of the BR and MMP ensembles were initialized with random values. We performed also tests with a multilabel variant of the multinomial Naive Bayes (MLNB) algorithm in order to provide a baseline. Another baseline is depicted by FC (frequency classifier) that returns always the same ranking of classes according to the class frequency in the training set.

6.1 Ranking Quality

The results for the four algorithms and the three different classifications of EUR-Lex are presented in Table 3. DMLPP results are omitted since they differ only

Table 3. Average ranking losses for the three views on the data and for the different algorithms. For IsErr, OneErr, RankLoss and Margin low values are good, for AvgP the higher the better.

				1 epoch			2 epochs			5 epochs			10 epochs		
		FC	MLNB	BR	MMP	DCMLPP	BR	MMP	DCMLPP	BR	MMP	DCMLPP	BR	MMP	DCMLPP
subject matter	IsErr ×100	99.58	99.47	65.99	55.70	51.38	58.78	51.96	44.07	53.42	42.77	38.23	50.19	40.22	36.34
	OneErr ×100	77.83	98.68	35.71	30.58	22.78	27.13	27.09	17.29	22.69	18.38	13.49	20.64	15.97	12.55
	RankLoss	12.89	8.885	17.38	2.303	1.064	13.89	2.520	0.911	11.58	2.091	0.796	9.752	1.85	0.762
	Margin	40.16	25.04	62.31	10.11	4.316	52.28	11.22	3.757	44.77	9.366	3.337	38.45	8.177	3.214
	AvgP	22.57	11.91	59.33	74.01	78.68	66.07	76.95	82.73	70.69	82.10	85.64	73.30	83.75	86.52
directory code	IsErr ×100	91.51	99.34	52.80	47.68	36.55	46.26	40.01	32.38	40.76	33.28	29.22	37.55	31.39	28.30
	OneErr ×100	90.13	99.04	44.40	40.85	28.22	37.38	32.99	24.42	31.48	25.79	21.41	28.1	23.9	20.65
	RankLoss	14.17	7.446	19.40	2.383	0.972	15.09	2.058	0.863	11.69	1.874	0.824	9.876	1.529	0.815
	Margin	68.33	34.44	96.43	14.18	5.626	77.32	12.18	5.045	61.48	10.95	4.831	52.94	8.947	4.785
	AvgP	18.98	6.714	57.10	68.70	77.89	63.68	74.90	80.87	68.75	79.84	82.87	71.61	81.30	83.38
EUROVOC	IsErr ×100	99.82	99.82	99.25	99.14	98.20	98.70	98.00	96.75	97.46	96.14		97.06	95.13	
	OneErr ×100	93.52	99.58	53.11	78.98	34.76	44.93	56.88	28.01	36.69	39.46		33.84	34.99	
	RankLoss	12.97	22.34	39.78	3.669	2.692	35.25	4.091	2.398	30.93	4.573		28.59	4.509	
	Margin	1357.10	1623.72	3218.12	562.81	426.28	3040.01	670.65	387.51	2846.47	757.01		2716.63	740.12	
	AvgP	5.504	1.060	25.55	27.04	46.79	30.71	38.42	52.72	35.95	47.65		38.31	50.71	

Table 4. Computational costs in CPU-time and millions of real value operations (M op.)

subject matter	training	testing	*directory code*	training	testing	*EUROVOC*	training	testing
BR	29.96 s	7.09 s	BR	50.67 s	9.62 s	BR	368.02 s	53.34 s
	1,680 M op.	184 M op.		3,420 M op.	378 M op.		33,074 M op.	3,662 M op.
MMP	31.95 s	6.89 s	MMP	53.38 s	9.46 s	MMP	479.14 s	52.90 s
	1,807 M op.	184 M op.		3,615 M op.	378 M op.		40,547 M op.	3,662 M op.
DMLPP	372.14 s	151.98 s	DMLPP	383.40 s	187.65 s	DMLPP	13,058.01 s	6,780.51 s
	6,035 M op.	4,471 M op.		3,047 M op.	5,246 M op.		17,647 M op.	123,422 M op.
MLPP	69.50 s	164.04 s	MLPP	120.70 s	643.34 s	MLPP	–	–
	3,886 M op.	18,427 M op.		4,735 M op.	77,629 M op.		–	–

slightly from those of DCMLPP due to the possible additional (one) vote won against the artificial label. In the same way, we omit the results of MLPP since they differ only marginally due to a different random initialization. Note however that MLPP cannot be applied to the *EUROVOC* dataset due to the high memory requirements, which was the reason for developing the dual version.

The values for IsErr, OneErr, RankLoss and AvgP are shown $\times 100\%$ for better readability, AvgP is also presented in the conventional way (with 100% as the optimal value) and not as a loss function. The number of epochs indicates the number of times that the online-learning algorithms were able to see the training instances. No results are reported for the performance of DCMLPP on *EUROVOC* for more than two epochs due to time restrictions. Note also that the results differ slightly from those of previous experiments in [14, 13] due to the modifications to the dataset presented in Section 2.

The first appreciable characteristic is that DCMLPP dominates all other algorithms on all three views of the EUR-Lex data, regardless of the number of epochs or losses. Often DCMLPP achieves better results than the other algorithms for more epochs. Especially on the losses that directly evaluate the ranking performance the improvement is quite pronounced and the results are already unreachable after the first epoch.

In addition to the fact that the DMLPP outperforms the remaining algorithms, it is still interesting to compare the performances of MMP and BR as they have still the advantage of reduced computational costs and memory requirements in comparison to the (dual) pairwise approach and could therefore be more applicable for very complex data sets such as *EUROVOC*, which is certainly hard to tackle for DMLPP (cf. Section 6.3).

For the *subject matter* and *directory code*, the results clearly show that the MMP algorithm outperforms the simple one-against-all approach. Especially on the losses that directly evaluate the ranking performance the improvement is quite pronounced. The smallest difference can be observed in terms of OneErr, which evaluates the top class accuracy.

The performance on the *EUROVOC* descriptor data set confirms the previous results. The differences in RankLoss and Margin are very pronounced. In contrast, in terms of OneErr the MMP algorithm is worse than one-against-all, even after ten epochs. It seems that with an increasing amount of classes, the MMP algorithm has more difficulties to push the relevant classes to the top such that the margin is big enough to leave all irrelevant classes below, although the algorithm in general clearly gives the relevant classes a higher score than the one-against-all approach. An explanation could be the dependence between the perceptrons of the MMP. This leads to a natural normalization of the scalar product, while there is no such restriction when trained independently as done in the binary relevance algorithm. As a consequence there could be some perceptrons that produce high maximum scores and thereby often arrive at top positions at the overall ranking. Furthermore, MMP's accuracy on RankLoss and Margin seems to suffer from the increased number of classes, since the loss increases from the first to the fifth epoch, and still in the tenth epoch the value

Table 5. Average multilabel losses for the three views on the data and for the label set predicting BR and DCMLPP. For HamLoss low values are good, for the remaining measures high values near 100% are good.

		1 epoch		2 epochs		5 epochs		10 epochs	
		BR	DCMLPP	BR	DCMLPP	BR	DCMLPP	BR	DCMLPP
subject matter	HamLoss	1.196	0.715	1.004	0.641	0.823	0.574	0.757	0.540
	F1	54.39	62.43	60.13	68.81	65.73	72.66	68.32	74.47
	Rec	64.64	54.02	68.66	64.26	71.62	69.25	74.11	71.56
	Prec	47.03	74.08	53.55	74.09	60.74	76.43	63.39	77.63
directory code	HamLoss	0.416	0.231	0.355	0.198	0.289	0.179	0.265	0.169
	F1	46.81	49.37	53.28	62.95	59.74	67.75	62.58	69.64
	Rec	58.31	36.05	64.51	53.55	68.36	59.87	70.54	61.89
	Prec	39.13	78.56	45.41	76.38	53.07	78.04	56.26	79.61
EUROVOC	HamLoss	0.267	0.125	0.238	0.117	0.208		0.199	
	F1	26.95	18.20	31.56	36.11	36.42		38.57	
	Rec	37.03	10.45	41.30	24.89	44.84		46.93	
	Prec	21.19	71.62	25.54	65.82	30.67		32.74	

is higher than after only one epoch. Perhaps it is indicated to use a different loss for MMP to optimize for problems with higher amount of classes, where IsErr is inevitably high (cf. Section 3.5). The price to pay for the good OneErr of BR is a decreased quality of the produced rankings, as the results for RankLoss and Margin are even beaten by Naive Bayes, which is by far the worst algorithm for the other losses.

It is interesting to note in this context that the frequency classifier often achieves a better performance than Naive Bayes and even BR, especially with increasing number of classes as with *EUROVOC*.

The fact that in only approximately 5% of the cases a perfect classification is achieved and in only approx. 65% the top class is correctly predicted in *EUROVOC* (MMP) should not lead to an underestimation of the performance of these algorithms. Considering that with almost 4000 possible classes and only 5.3 classes per example the probability of randomly choosing a correct class is less than one percent, namely 0.13%, the performance is indeed substantial.

6.2 Multilabel Classification Quality

Table 5 shows the several results for predicting a set of labels for each instance rather than a ranking of labels. Obviously, only results for BR and the calibrated version of DMLPP can be shown since MMP only produces a ranking. The first remarkable point is that DCMLPP outperforms BR in all direct comparisons for the overall measures HamLoss and F1 and also Prec. But interestingly, BR always achieves a higher Rec than DCMLPP. This is due to the fact that the calibration tends to underestimate the number of returned labels for each instance, especially for a high number of total classes and when the base classifiers are not yet that accurate such as for low numbers of epochs. A possible explanation for this behavior is the following: when the BR classifier, that is also included in DCMLPP or CMLPP, predicts that v classes are positive, this means for the remaining classes that they have to obtain at least $n-v$ votes of their maximum of n votes in order to be predicted as positive. The probability that this happens for a real positive class decreases with increasing n and increasing error of the base classifiers, since it becomes more probably that at least v base classifiers mistakenly commit a wrong decision.

The average label set size that is produced by DCMLPP demonstrates this: for *subject matter* it increases from 1.65 to 2.04 (BR from 3.05 to 2.59), for *directory code* from 0.59 to 1.0 (BR: 1.93 to 1.62) and for *EUROVOC* it increases from small 0.77 to 2.01 after the second epoch (BR from 9.28 to 7.61 in the tenth epoch). BR begins with an overestimation, reducing the predicted size subsequently.

In order to allow a comparison independent of different tendencies of the different thresholding techniques, we have computed Rec and Prec using the correct, true size of the label set of the test examples. With this trick, we obtained a Rec/Prec of 68.5% for BR and 79.3 for DCMLPP on the last epoch of *subject matter*. Note that Rec equals always Prec since we have always the same amount of false positives and false negatives in the confusion matrix. For *directory code* the values are 64.8 and 75.0, and for *EUROVOC* 33.6 and 48.0 for the second epoch, 40.8 for the tenth epoch for BR. This trick can also be applied in order to compare to MMP, which is beaten by DCMLPP but better than BR with 76.2, 71.9, 35.2 and 47.1 respectively.

6.3 Computational Costs

In order to allow a comparison independent from external factors such as logging activities and the run-time environment, we ignored minor operations that have to be performed by all algorithms, such as sorting or internal operations. An overview over the amount of real value addition and multiplication computations is given in Table 4 (averaged over the cross validation splits, trained for one epoch), together with the CPU-times on an AMD Dual Core Opteron 2000 MHz as additional reference information. We report only results of DMLPP, since DCMLPP's operations and seconds can easily be derived or estimated by adding those of BR. Furthermore, we include the results for the non-dual MLPP, however no values have been received for the *EUROVOC* problem due to the memory space problem discussed at the end of this section.

We can observe a clear advantage of the non-pairwise approaches on the *subject matter* data especially for the prediction phase, however the training costs are in the same order of magnitude. Between MLPP and DMLPP we can see an antisymmetric behavior: while MLPP requires only almost half of the amount of the DMLPP operations for training, DMLPP reduces the amount of prediction operations by a factor of more than 4. For the *directory code* the rate for MMP and BR more than doubles in correspondence with the increase in number of

Table 6. Memory requirements of the different classifiers for the EUR-Lex datasets.

dataset	BR/MMP	DMLPP	DCMLPP	MLPP
subject matter	153 MB	199 MB	210 MB	541 MB
directory code	167 MB	210 MB	229 MB	1,818 MB
EUROVOC	1,145 MB	1,242 MB	1,403 MB	–

classes, additionally the MLPP testing time substantially increases due to the quadratic dependency, while DMLPP profits from the decrease in the average number of classes per instance. It even causes less computations in the training phase than MMP/BR. The reason for this is not only the reduced maximum amount of weights per instance (cf. Section 5), but particularly the decreased probability that a training example is relevant for a new training example (and consequently that dot products and scores have to be computed) since it is less probable that both class assignments match, i.e. that both examples have the same pair of positive and negative classes. This becomes particularly clear if we observe the number of non-zero weights and actually used weights during training for each new example. The classifier for *subject matter* has on average 20 weights set per instance out of 440 ($= d(n-d)$) in the worst case (a ratio of 4.45%), and on average 4.97% of them are required when a new training example arrives. For the *directory code* with a smaller fraction d/n 35.0 weights are stored (6.66%), of which only 1.10% are used when updating. This also explains the relatively small number of operations for training on *EUROVOC*, since from the 1,781 weights per instance (8.45%), only 0.55% are relevant to a new training instance. In this context, regarding the disturbing ratio between real value operations and CPU-time for training DMLPP on *EUROVOC*, we believe that this is caused by a suboptimal storage structure and processing of the weights and we are therefore confident that it is possible to reduce the distance to MMP in terms of actual consumed CPU-time by improving the program code. Memory swapping may also have influenced the measurement.

Note that MMP and BR compute the same amount of dot products, the computational costs only differ in the number of vector additions, i.e. perceptron updates. It is therefore interesting to observe the contrary behavior of both algorithms when the number of classes increases: while the one-against-all algorithm reduces the ratio of updated perceptrons per training example from 1.33% to 0.34% when going from 202 to 3993 classes, the MMP algorithm more than doubles the rate from 8.53% to 22.22%. For the MMP this behavior is natural: with more classes the error set size increases and consequently the number of updated perceptrons. In contrast BR receives less positive examples per base classifier, the perceptrons quickly adopt the generally good rule to always return a negative score, which leads to only a few binary errors and consequently to little corrective updates. A more extensive comparison of BR and MMP can be found in a previous work [11].

The memory consumption provided by the Java Virtual Machine after training the several classifiers for one epoch is depicted in Table 6.3. Note that these sizes include the overhead caused by the virtual machine and the machine learning framework.[5] MLPP already consumes more memory than the dual variant for the first dataset with 200 classes. For the 400 classes of the *directory code* view the algorithm requires almost 2 GB, while DMLPP is able to compress the same information into slightly more than 200 MB. As expected and already mentioned in Section 6.1, MLPP is not applicable to *EUROVOC*. A simple estimation based on the number of base classifiers, number of features and bytes per float variable results in 152 GB of memory. Another remarkable fact is that the memory requirement of DMLPP is comparable to that of the one-per-class algorithms: for the smaller datasets we obtain an overhead of only 50 MB and for the bigger *EUROVOC* view it requires only double of the memory, although representing a quadratic number of base classifiers. On the other hand, a view on the memory consumption of DCMLPP reveals that great part of the space for MMP/BR and DCMLPP is caused by overhead of the JVM and the machine learning framework (instances and class mappings in memory, extensive statistics, etc.). If we compute the core memory requirements of BR by subtracting DMLPP's value from DMLPP's for *EUROVOC*, we obtain 161 MB. In consequence we obtain a general overhead of 981 MB and thus an actual memory consumption of 261 MB for DMLPP. These values seem to be realistic after a simple estimation.

7 Conclusions

In this paper, we introduced the EUR-Lex text collection as a promising test bed for studies in text categorization, available at `http://www.ke.tu-darmstadt.de/resources/eurlex/`. Among its many interesting characteristics (e.g., multilinguality), our main interest was the large number of categories, which is one order of magnitude above other frequently studied text categorization benchmarks, such as the Reuters-RCV1 collection.

On the *EUROVOC* classification task, a multilabel classification task with 4000 possible labels, the DMLPP algorithm, which decomposes the problem into training classifiers for each pair of classes, achieves an average precision rate of slightly more than 50%. Roughly speaking, this means that the (on average) five relevant labels of a document will (again, on average) appear within the first 10 ranks in the relevancy ranking of the 4,000 labels. This is a very encouraging result for a possible automated or semi-automated real-world application for categorizing EU legal documents into EUROVOC categories.

This result was only possible by finding an efficient solution for storing the approx. 8,000,000 binary classifiers that have to be trained by this pairwise approach. To this end, we showed that a reformulation of the pairwise decompositi-

[5] We used the WEKA framework (`http://www.cs.waikato.ac.nz/~ml/weka/`), but we adapted it so that it maintains a copy of a training instance in memory only when necessary for the incremental updating.

on approach into a dual form is capable of handling very complex problems and can therefore compete with the approaches that use only one classifier per class. It was demonstrated that decomposing the initial problem into smaller problems for each pair of classes achieves higher prediction accuracy on the EUR-Lex data, since DMLPP substantially outperforms all other algorithms. This confirms previous results of the non-dual variant on the large Reuters Corpus Volume 1 [12]. The dual form representation allows for handling a much higher number of classes than the explicit representation, albeit with an increased dependence on the training set size. Despite the improved ability to handle large problems, DMLPP is still less efficient than MMP, especially for the *EUROVOC* data with 4000 classes. However, in our opinion the results show that DMLPP is still competitive for solving large-scale problems in practice, especially considering the trade-off between runtime and prediction performance.

As further work, we have adapted the efficient voting technique for pairwise classification introduced in [17] to the multilabel case. This technique permits to reduce the amount of perceptron predictions of the MLPP algorithm during the classification of the *subject matter* and *directory code* views to a level competitive to BR/MMP [15]. However, the processing of the almost 4000 classes of *EUROVOC* is out of scope, making it still necessary to use techniques such as the Dual MLPP.

Additionally, we are currently investigating hybrid variants to further reduce the computational complexity. The idea is to use a different formulation in training than in the prediction phase depending on the specific memory and runtime requirements of the task. In order e.g. to combine the advantage of MLPP during training and DMLPP during predicting on the *subject matter* subproblem, we could train the classifier as in the MLPP (with the difference of iterating over the perceptrons first so that only one perceptron has to remain in memory at a point in time) and than convert it to the dual representation by means of the collected information during training the perceptrons. The use in training of SVMs or more advanced perceptron variants that, similar to SVMs, try to maximize the margin of the separating hyperplane in order to produce more accurate models [4, 8], is also an interesting option.

Acknowledgements This work was supported by the EC 6th framework project ALIS (Automated Legal Information System) and by the German Science Foundation (DFG).

8 References

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[2] Klaus Brinker, Johannes Fürnkranz, and Eyke Hüllermeier. A Unified Model for Multilabel Classification and Ranking. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI-06)*, 2006.

[3] Koby Crammer and Yoram Singer. A Family of Additive Online Algorithms for Category Ranking. *Journal of Machine Learning Research*, 3(6):1025–1058, 2003.

[4] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[5] Johannes Fürnkranz. Round Robin Classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[6] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2): 133–153, 2008.

[7] Chih-Wei Hsu and Chih-Jen Lin. A Comparison of Methods for Multi-class Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[8] Roni Khardon and Gabriel Wachman. Noise tolerant variants of the perceptron algorithm. *Journal of Machine Learning Research*, 8:227–248, 2007.

[9] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[10] Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S. Kandola. The Perceptron Algorithm with Uneven Margins. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002)*, pages 379–386, 2002.

[11] Eneldo Loza Mencía and Johannes Fürnkranz. An evaluation of efficient multilabel classification algorithms for large-scale problems in the legal domain. In *LWA 2007: Lernen - Wissen - Adaption, Workshop Proceedings*, pages 126–132, 2007.

[12] Eneldo Loza Mencía and Johannes Fürnkranz. Pairwise learning of multilabel classifications with perceptrons. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN 08)*, pages 2900–2907, Hong Kong, 2008.

[13] Eneldo Loza Mencía and Johannes Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Disocvery in Databases (ECML-PKDD-2008), Part II*, pages 50–65, Antwerp, Belgium, 2008. Springer-Verlag.

[14] Eneldo Loza Mencía and Johannes Fürnkranz. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In *Proceedings of the Language Resources and Evaluation Conference (LREC) Workshop on Semantic Processing of Legal Texts*, pages 23–32, Marrakech, Morocco, 2008.

[15] Eneldo Loza Mencía, Sang-Hyeun Park, and Johannes Fürnkranz. Efficient voting prediction for pairwise multilabel classification. In *Proceedings of the 11th European Symposium on Artificial Neural Networks (ESANN-09)*. Springer, 2009.

[16] Arturo Montejo Ráez, Luis Alfonso Ureña López, and Ralf Steinberger. Adaptive selection of base classifiers in one-against-all learning for large multi-labeled collections. In *Advances in Natural Language Processing, 4th International Conference (EsTAL 2004), Alicante, Spain, October 20-22, Proceedings*, volume 3230 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2004.

[17] Sang-Hyeun Park and Johannes Fürnkranz. Efficient pairwise classification. In J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, editors, *Proceedings of 18th European Conference on Machine Learning (ECML-07)*, pages 658–665, Warsaw, Poland, 2007. Springer-Verlag.

[18] Bruno Pouliquen, Ralf Steinberger, and Camelia Ignat. Automatic annotation of multilingual text collections with a conceptual thesaurus. In *Proceedings of the Workshop 'Ontologies and Information Extraction' at the Summer School 'The Semantic Web and Language Technology - Its Potential and Practicalities' (EU-ROLAN'2003), 28 July - 8 August 2003*, Bucharest, Romania, 2003.

[19] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[20] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[21] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420. Morgan Kaufmann Publishers Inc., 1997.