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.informatik.tu-darmstadt.de

Abstract

In this paper we evaluate the performance of multilabel classification algorithms on the EUR-Lex database of legal documents of the European Union. On the same set of underlying documents, we defined three different large-scale multilabel problems with up to 4000 classes. On these datasets, we compared three algorithms: (i) the well-known one-against-all approach (OAA); (ii) the multiclass multilabel perceptron algorithm (MMP), which modifies the OAA ensemble by respecting dependencies between the base classifiers in the training protocol of the classifier ensemble; and (iii) the multilabel pairwise perceptron algorithm (MLPP), which unlike the previous algorithms trains one base classifier for each pair of classes. All algorithms use the simple but very efficient perceptron algorithm as the underlying classifier. This makes them very suitable for large-scale multilabel classification problems. While previous work has already shown that the latter approach outperforms the other two approaches in terms of predictive accuracy, its key problem is that it has to store one classifier for each pair of classes. The key contribution of this work is to demonstrate a novel technique that makes the pairwise approach feasible for problems with large number of classes, such as those studied in this work. Our results on the EUR-Lex database illustrate the effectiveness of the pairwise approach and the efficiency of the MMP algorithm. We also show that it is feasible to efficiently and effectively handle very large multilabel problems.

1. Introduction

Recently, *multilabel classification* problems, where the task is to associate an object with an unbounded set of classes instead of exactly one, have received increased attention in the literature. As a consequence, new algorithms have been developed or adapted to automatically solve the task of multilabel classification. But simultaneously an increased number of new scenarios have been identified and higher demands are continuously made to the existing algorithms. This concerns not only challenges due to large scale instance spaces, large numbers of instances and numbers of features, but particularly due to the number of possible classes. In particular in text classification, these type of problems are very common. The number of possible categories that can typically be assigned to each document varies from a few dozen to several hundred. In this paper, we study a challenging new domain, namely assigning documents of the EUR-Lex database to a few of $\approx 4,000$ possible labels. The EUR-Lex database is a freely accessible document management system for legal documents of the European Union. We chose this database 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.

The simplest strategy to tackle the multilabel problem with existing techniques is to use *one-against-all* binarization, in the multilabel setting also referred to as the *binary relevance* method. It decomposes the original problem into less complex, binary problems, by learning one classifier for each class, using all objects of this class as positive examples and all other objects as negative examples. At query time, each binary classifier predicts whether its class is relevant for the query example or not, resulting in a set of relevant labels. While this technique can potentially be used to transform any binary classifier into a multilabel classifier and it is often used in practical applications, the question remains whether this general approach can fully adapt to the particular needs of multilabel classification, because it trains each class independently of all other classes.

A recently proposed alternative that tries to tackle this problem is the *multilabel multiclass perceptron* (MMP) algorithm developed by Crammer and Singer (2003), which adapts the one-against-all approach to the multilabel case. Instead of learning the relevance of each class individually and independently, MMP incrementally trains the entire classifier ensemble as a whole so that it predicts a real-valued relevance value for each class. This is done by always evaluating the performance of the entire ensemble, and only producing training examples for the individual classifiers when their corresponding classes are misplaced in the ranking. It uses perceptrons as base classifiers, because they are simple and efficient, and because for high-dimensional problems such as text classification, linear discriminants are sufficiently expressive.

An alternative training method for an effective ensemble of perceptrons is the pairwise decomposition of the initial problem proposed by the *multilabel pairwise perceptron* (MLPP) algorithm (Loza Mencía and Fürnkranz, 2008). In this method, one perceptron is trained for each possible class pair, using the examples belonging to the two classes as positive or negative examples respectively. During prediction, an overall ranking of the classes is determined by combining the predictions of the individual base perceptrons, e.g. by voting. The first main advantage of the pairwise approach is its effectiveness: decomposing the problem into smaller subproblem will yield simpler, often linear decision boundaries, and this usually leads to an increased prediction performance (Knerr et al., 1992; Hsu and Lin, 2002; Fürnkranz, 2002). However, the second advantage of increased efficiency in training compared to one-against-all ensembles (Fürnkranz, 2002) is counteracted by the quadratic number of base classifiers required in proportion to the number of classes. While the pairwise decomposition in combination with perceptrons as base classifiers is well applicable for a large feature space and a large amount of training instances (Loza Mencía and Fürnkranz, 2008), the large number of classes in the EUR-Lex database constitutes an insurmountable obstacle: in the most complex setting approximately 8,000,000 perceptrons would be needed to represent a pairwise ensemble.

We will therefore 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.

The MMP algorithm has already been used on large scale data sets such as the Reuters Corpus Volume 1 (Lewis et al., 2004) with its over 800,000 examples and approx. 100 classes (Crammer and Singer, 2003). In the experiments the MMP algorithm was able to substantially improve the performance of the one-against-all method with perceptrons as base classifiers. In a recent evaluation on the same data the pairwise approach showed an even higher performance than the MMP algorithm (Loza Mencía and Fürnkranz, 2008), demonstrating the applicability of MLPP to large-scale problems. In this paper we will analyze if these results can be repeated with the EUR-Lex database in three different settings, one with approx. 200, another with 400 and finally one with 4000 possible classes. Note that the latter problem has more classes by an order of magnitude than other known applications of these algorithms.

A shortcoming of MMP and the pairwise method is that the resulting prediction is not any more a set of classes as expected for a multilabel task, but a ranking of class relevance scores. However, it is possible to obtain the desired output in an additional step that selects classes which exceed a determined relevance value. Different methods exist for determining the threshold, a good overview can be found in Sebastiani (2002). Recently, Brinker et al. (2006) introduced the idea of using an artificial label that encodes the boundary between relevant and irrelevant labels for each example. In this paper, we will concentrate on the label ranking task, which also enables a more detailed evaluation of the classification performance.

2. Preliminaries

We represent an instance or object as a vector $\bar{x} = (x_1, \ldots, x_M)$ in a feature space $\mathcal{X} \subseteq \mathbb{R}^N$. Each instance $\bar{x}_i$ is assigned to a set of relevant labels y_i, a subset of the K possible classes $\mathcal{Y} = \{c_1, \ldots, c_K\}$. For multilabel problems, the cardinality $|y_i|$ of the label sets is not restricted, whereas for binary problems $|y_i| = 1$. For the sake of simplicity we use the following notation for the binary case: we define $\mathcal{Y} = \{1, -1\}$ as the set of classes so that each object $\bar{x}_i$ is assigned to a $y_i \in \{1, -1\}$, $y_i = \{y_i\}$.

2.1. 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 $\bar{x}$, a relevant label set y, a negative label set $\overline{y} = \mathcal{Y} \backslash y$ and a given predicted ranking $r : \mathcal{Y} \to \{1 \ldots K\}$, with $r(c)$ returning the position of class c in the ranking, the different loss functions are computed as follows:

ISERR The is-error loss determines whether $r(c) < r(c')$ for all relevant classes $c \in y$ and all irrelevant classes $c' \in \overline{y}$. It returns 0 for a completely correct, *perfect ranking*, and 1 for an incorrect ranking, irrespective of 'how wrong' the ranking is.

ONEERR The one error loss 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.

RANKLOSS The ranking loss 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.

$$E \stackrel{\text{def}}{=} \{(c, c') \mid r(c) > r(c')\} \subseteq y \times \overline{y} \qquad (1)$$

$$\delta_{\text{RankLoss}} \stackrel{\text{def}}{=} \frac{|E|}{|y||\overline{y}|} \qquad (2)$$

MARGIN The 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.

$$\begin{aligned} F \stackrel{\text{def}}{=} &\{c \in y \mid r(c) > r(c'), c' \in \overline{y}\} \\ &\cup \{c' \in \overline{y} \mid r(c) > r(c'), c \in y\} \end{aligned} \qquad (3)$$

$$\begin{aligned} \delta_{\text{Margin}} \stackrel{\text{def}}{=} &\max(0, \max\{r(c) \mid c \in y\} \\ &- \min\{r(c') \mid c' \notin y\}) \end{aligned} \qquad (4)$$

AVGP Average Precision 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.

$$\delta_{\text{AVGP}} \overset{\text{def}}{=} 1 - \frac{1}{\mathcal{Y}} \sum_{c \in \mathcal{Y}} \frac{|\{c^* \in \mathcal{Y} \mid r(c^*) \leq r(c)\}|}{r(c)} \quad (5)$$

2.2. Perceptrons

We use the simple but fast perceptrons as base classifiers (Rosenblatt, 1958). 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:

$$o'(\bar{\mathrm{x}}) = sgn(\bar{\mathrm{x}} \cdot \bar{\mathrm{w}}) \quad (6)$$

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., (Bishop, 1995)).

$$\alpha_i = (y_i - o'(\bar{\mathrm{x}}_i)) \qquad \bar{\mathrm{w}}_{i+1} = \bar{\mathrm{w}}_i + \alpha_i \bar{\mathrm{x}}_i \quad (7)$$

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

$$\bar{\mathrm{w}} = \sum_{i=1}^{M} \alpha_i \bar{\mathrm{x}}_i \qquad o'(\bar{\mathrm{x}}) = sgn(\sum_{i=1}^{M} \alpha_i \cdot \bar{\mathrm{x}}_i \bar{\mathrm{x}}) \quad (8)$$

assuming M as 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 . The main reason for choosing Perceptron 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 RCV1 benchmark (Lewis et al., 2004), without forfeiting too much accuracy though SVMs find the *maximum-margin hyperplane* (Freund and Schapire, 1999; Crammer and Singer, 2003; Shalev-Shwartz and Singer, 2005).

2.3. Binary Relevance Ranking

In the binary relevance (BR) or one-against-all (OAA) method, a multilabel training set with K possible classes is decomposed into K binary training sets of the same size that are then used to train K binary classifiers. So for each pair $(\bar{\mathrm{x}}_i, \mathcal{Y}_i)$ in the original training set K different pairs of instances and binary class assignments $(\bar{\mathrm{x}}_i, y_{i_j})$ with $j = 1 \ldots K$ are generated as follows:

$$y_{i_j} = \begin{cases} 1 & c_j \in \mathcal{Y}_i \\ -1 & otherwise \end{cases} \quad (9)$$

Supposing we use perceptrons as base learners, K different o'_j classifier are trained in order to determine the relevance of c_j. In consequence, the combined prediction of the binary relevance classifier for an instance $\bar{\mathrm{x}}$ would be the set $\{c_j \mid o'_j(\bar{\mathrm{x}}) = 1\}$. If, in contrast, we want to obtain a ranking of classes according to their relevance, we can simply use the result of the internal computation of the perceptrons as a measure of relevance. According to Equation 6 the desired linear combination is the inner product $o_j(\bar{\mathrm{x}}) = \bar{\mathrm{x}} \cdot \bar{\mathrm{w}}_j$ (ignoring ω as mentioned above). So the result of the prediction is a vector $\bar{\mathrm{o}}(\bar{\mathrm{x}}) = (\bar{\mathrm{x}}\bar{\mathrm{w}}_1, \ldots, \bar{\mathrm{x}}\bar{\mathrm{w}}_K)$ where component j corresponds to the relevance of class c_j. Ties are broken randomly to not favor any particular class.

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

Figure 1: Pseudocode of the training method of the MMP algorithm

2.4. Multiclass Multilabel Perceptrons

MMPs were proposed as an extension of the one-against-all algorithm with perceptrons as base learners (Crammer and Singer, 2003). 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. 1 describes the MMP training algorithm. When the MMP algorithm receives a training instance $\bar{\mathrm{x}}$, it calculates the inner products, the ranking and the loss on this ranking in order to determine whether the current model needs an update. For determining the ranking loss, any of the methods of Sec. 2.1. is appropriate, since they all return a low value on good rankings. If the ranking is perfect, the algorithm is done, otherwise it calculates the error set of wrongly ordered class pairs E. The wrongly ranked classes are also represented in F. In the next step, each class that is present in a pair of E receives a penalty score. This is done according to a selectable penalty function, being the uniform update method, where each pair in E receives the same score, the most successful one (Crammer and Singer, 2003). In the next step, the update weights τ are

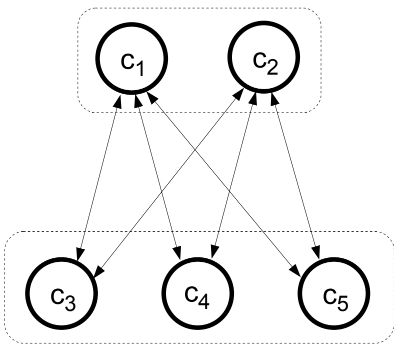


Figure 2: MLPP training: training example $\bar{x}$ belongs to $y = \{c_1, c_2\}$, $\bar{y} = \{c_3, c_4, c_5\}$ are the irrelevant classes, the arrows represent the trained perceptrons.

```
Require: Training example pair $(\bar{x}, y)$,
    perceptrons $\{\bar{w}_{u,v} \mid u < v, c_u, c_v \in \mathcal{Y}\}$
1: for each $(c_u, c_v) \in y \times \bar{y}$ 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 $\{\bar{w}_{u,v} \mid u < v, c_u, c_v \in \mathcal{Y}\}$
                                        ▷ updated perceptrons
```

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

normalized and each perceptron whose class was wrongly ordered is updated.

2.5. Multilabel Pairwise Perceptrons

In the pairwise binarization method, one classifier is trained for each pair of classes, i.e., a problem with K different classes is decomposed into $\frac{K(K-1)}{2}$ smaller subproblems. For each pair of classes (c_u, c_v), only examples belonging to either c_u or c_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., $(u, v) \in y \times \bar{y}$ (cf. Figure 2). 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 $\frac{K(K-1)}{2}$ perceptrons the classes are ordered according to their sum of votes.[1]

Figure 4 shows a possible result of classifying the sample instance of Figure 2. Perceptron $o'_{1,5}$ predicts (correctly) the first class, consequently c_1 receives one vote and class c_5

[1] Ties are broken randomly in our case.

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

Figure 4: MLPP voting: an example $\bar{x}$ is classified by all 10 base perceptrons $o'_{u,v}, u \neq v,\ c_u, c_v \in \mathcal{Y}$. Note the redundancy given by $o'_{u,v} = -o'_{v,u}$. The last line counts the positive outcomes for each class.

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 c_1 or c_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 on a single negative classes, which would lead to misclassifications. But note that any class can at most receive $K-1$ 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 $K - |y|$ and a negative at most $K - |y| - 1$ votes. Class c_3 in Figure 4 is an example for this: It receives all possible noisy votes but still loses against the positive classes c_1 and c_2.

The pairwise binarization method is often regarded as superior to binary relevance because it profits from simpler decision boundaries in the subproblems (Fürnkranz, 2002; Hsu and Lin, 2002). In the case of an equal class distribution, the subproblems have $\frac{2}{K}$ 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 Corpus Volume 1 (Lewis et al., 2004), which contains over 100 classes and 800,000 documents, showed a significant and substantial improvement over the MMP method (Loza Mencía and Fürnkranz, 2008). 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 the next section.

3. 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 huge amount of 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 8. 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:

$$\begin{aligned}
\bar{\mathrm{w}}_{1,2}\bar{\mathrm{x}} &= \alpha_{1,2}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,2}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,2}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
\bar{\mathrm{w}}_{1,3}\bar{\mathrm{x}} &= \alpha_{1,3}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,3}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,3}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
&\vdots \\
\bar{\mathrm{w}}_{1,K}\bar{\mathrm{x}} &= \alpha_{1,K}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,K}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,K}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
\bar{\mathrm{w}}_{2,3}\bar{\mathrm{x}} &= \alpha_{2,3}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{2,3}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{2,3}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
&\vdots
\end{aligned} \tag{10}$$

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 5 and 6. We denote this variant of training the pairwise perceptrons the dual multilabel pairwise perceptrons algorithm (DMLPP).

Despite the consequences for the memory requirements and the run-time analyzed in detail in Section 4., 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.

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 5 to an additive update $\alpha_{u,v}^{t} = \alpha_{u,v}^{t} + 1$ for a revisited example $\bar{\mathrm{x}}_t$. 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.

```
Require: New training example pair $(\bar{x}_M, y)$,
    training examples $\bar{x}_1 \ldots \bar{x}_{M-1}$,
    weights $\{\alpha_{u,v}^i \mid c_u, c_v \in \mathcal{Y}, 0 < i < M\}$
1: for each $\bar{x}_i \in \bar{x}_1 \ldots \bar{x}_{M-1}$ do
2:     $p_i \leftarrow \bar{x}_i \cdot \bar{x}_M$
3:     for each $(c_u, c_v) \in y \times \bar{y}$ do
4:         if $\alpha_{u,v}^i \neq 0$ then
5:             $s_{u,v} \leftarrow s_{u,v} + \alpha_{u,v}^i \cdot p_t$
                                          ▷ note that $s_{u,v} = -s_{v,u}$
6: for each $(c_u, c_v) \in y \times \bar{y}$ do
7:     if $s_{u,v} < 0$ then
8:         $\alpha_{u,v}^M \leftarrow 1$              ▷ note that $\alpha_{u,v} = -\alpha_{v,u}$
9: return $\{\alpha_{u,v}^M \mid (c_u, c_v) \in y \times \bar{y}\}$     ▷ return new weights
```

Figure 5: Pseudocode of the training method of the DMLPP algorithm.

```
Require: example $\bar{x}$ for classification,
    training examples $\bar{x}_1 \ldots \bar{x}_{M-1}$,
    weights $\{\alpha_{u,v}^i \mid c_u, c_v \in \mathcal{Y}, 0 < i < M\}$
1: for each $\bar{x}_i \in \bar{x}_1 \ldots \bar{x}_M$ do
2:     $p \leftarrow \bar{x}_i \cdot \bar{x}$
3:     for each $(c_u, c_v) \in y_i \times \bar{y}_t$ do
4:         if $\alpha_{u,v}^i \neq 0$ then
5:             $s_{u,v} \leftarrow s_{u,v} + \alpha_{u,v}^i \cdot p$
6: for each $(c_u, c_v) \in \mathcal{Y} \times \mathcal{Y}$ do
7:     if $u \neq v \vee s_{u,v} > 0$ then
8:         $v_u \leftarrow v_u + 1$
9: return voting $\bar{v} = (v_1, \ldots, v_{|\mathcal{Y}|})$          ▷ return voting
```

Figure 6: Pseudocode of the prediction phase of the DMLPP algorithm.

4. Computational Complexity

The notation used in this section is the following: K denotes the number of possible classes, L 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 $\frac{K(K-1)}{2}$ perceptrons, each with N weights in memory, hence we need $O(K^2N)$ memory. The DMLPP algorithms 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 $y \times \bar{y}$. Furthermore, it has to intermediately store the resulting scores for each base perceptron during prediction, hence the complexity is $O(MLK + MN' + K^2) = O(M(LK + N') + K^2)$.[2] 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, $O(LK)$ 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 $O(LKN')$ per trai-

[2] Note that we do not estimate L as $O(K)$ 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).

	training time	testing time	memory requirement
MMP, BR	$O(KN')$	$O(KN')$	$O(KN)$
MLPP	$O(LKN')$	$O(K^2N')$	$O(K^2N)$
DMLPP	$O(M(LK + N'))$	$O(M(LK + N'))$	$O(M(LK + N') + K^2)$

Table 1: Computational complexity given in expected number of addition and multiplication operations. K: *#classes*, L: *avg. #labels per instance*, M: *#training examples*, N: *#attributes*, N': *#attributes*$\neq 0$, $\hat{\delta}$: *avg. Loss*, $\hat{\delta}_{per}, \hat{\delta}_{\text{IsErr}} \leq 1, \hat{\delta}_{\text{Margin}} < K$.

ning example. Similarly, the DMLPP spends M dot product computations. In addition the summation of the scores costs $O(LK)$ per training instance, leading to $O(M(LK + N'))$ operations. It is obvious that MLPP has a clear advantage over DMLPP in terms of training time, unless K is of the order of magnitude of M or the model is trained over several epochs, as already outlined in the previous Section 3.
During prediction the MLPP evaluates all perceptrons, leading to $O(K^2N')$ computations. The dual variant again iterates over all training examples and associated weights, hence the complexity is $O(M(LK + N'))$. At this phase DMLPP 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 1, together with the complexities of MMP and BR. A more detailed comparison between MMP and MLPP is available from Loza Mencía and Fürnkranz (2008).
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, which was the main obstacle to applying the pairwise approach to large scale problems in terms of classes.

5. EUR-Lex Repository

The EUR-Lex/CELEX (Communitatis Europeae LEX) Site[3] 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. The documents are available in most of the languages of the EU, and in the HTML and PDF formats. 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*[4], in total 19,596 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).
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[5]. 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.
Figure 7 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.
The high number of 3,993 different EUROVOC descriptors were identified in the retrieved documents, each document is associated to 5.37 descriptors on average. In contrast there are only 201 different subject matters appearing in the dataset, with a mean of 2.23 labels per document, and 412 different directory 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 7 this would mean a set of labels of $\{17.20\}$ instead of $\{17, 17.20\}$.

5.1. Data Preprocessing

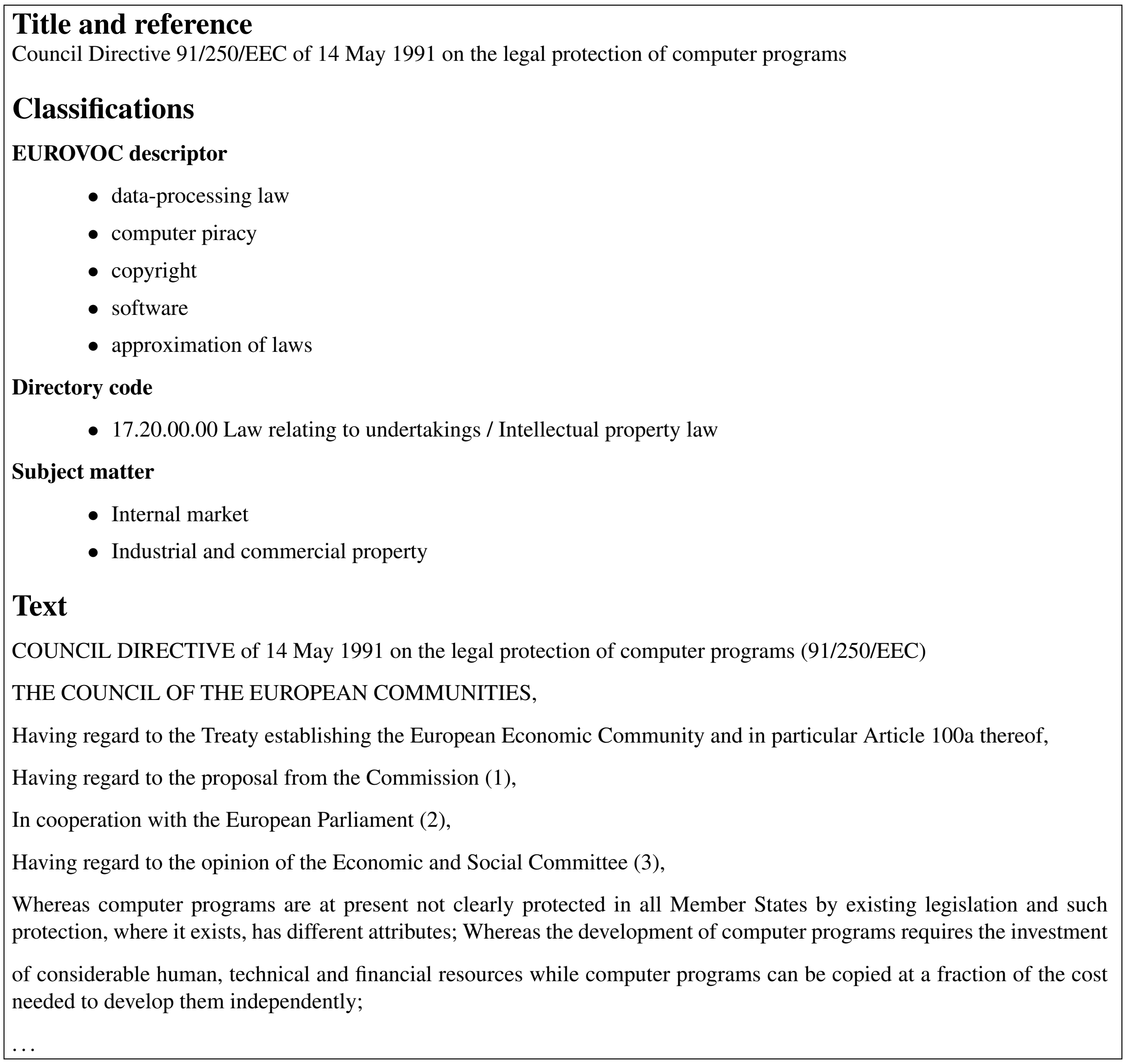
The main text was extracted from the HTML documents, excluding HTML tags, bibliographic notes or other additional information that could distort the results, and was then finally tokenized. The tokens were transformed to lower case, stop words were excluded, and the Porter stemmer algorithm was applied.[6] In order to perform cross validation, the instances were randomly distributed into ten folds. The tokens were projected into the vector space model using the common TF-IDF term weighting (Sebastiani, 2002). 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 (Yang and Pedersen, 1997). 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.

[3]http://eur-lex.europa.eu

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

[5]http://europa.eu/eurovoc/

[6]The implementation from the Apache Lucene Project (http://lucene.apache.org/java/docs/index.html) was used.



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),

Having regard to the opinion of the Economic and Social Committee (3),

Whereas computer programs are at present not clearly protected in all Member States by existing legislation and such protection, where it exists, has different attributes; Whereas the development of computer programs requires the investment

of considerable human, technical and financial resources while computer programs can be copied at a fraction of the cost needed to develop them independently;

. . .

Figure 7: 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.

6. Evaluation

6.1. Algorithm Setup

For the MMP algorithm we used the ISERR loss function and the uniform penalty function. This setting showed the best results in (Crammer and Singer, 2003) 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. For the MLNB we counted the TF-IDF instead of the term frequency values as we obtained improved results by using this additional information about the overall relevance of each term.

6.2. Ranking Performance

The results for the four algorithms and the three different classifications of EUR-Lex are presented in Table 2. 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 DMLPP on EUROVOC for more than one epoch.

The first appreciable characteristic is that DMLPP dominates all other algorithms on all three views of the EUR-Lex data, regardless of the number of epochs or losses. For the directory code DMLPP achieve a result in epoch 2 that is still beyond the reach of the other algorithms in epoch 10, except for MMP's ISERR. 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. It is also interesting to note that the improvement between epoch 5 and epoch 10 is rather small compared to the previous steps. We can observe this effect also for the MMP algorithm advancing from 10 to 20 epochs (e.g. 40.01 for ISERR and 9.73 % for RANKLOSS

subject matter		1 epoch			2 epochs			5 epochs			10 epochs		
	MLNB	BR	MMP	DMLPP	BR	MMP	DMLPP	BR	MMP	DMLPP	BR	MMP	DMLPP
IsErr ×100	99.15	61.71	53.61	52.28	57.39	48.83	45.48	52.36	43.21	39.74	49.87	40.89	37.69
OneErr ×100	98.63	30.67	27.95	23.62	26.44	24.4	18.64	22.17	18.82	15.01	20.41	17.08	13.7
RankLoss	8.979	16.36	2.957	1.160	14.82	2.785	0.988	11.40	2.229	0.885	10.29	2.000	0.849
Margin	25.34	59.33	13.04	4.611	54.27	11.94	4.001	44.05	9.567	3.615	40.39	8.636	3.488
AvgP	12.26	62.9	74.71	77.98	66.61	78.05	82.05	71.28	81.71	84.76	73.06	83.19	85.79

directory code		1 epoch			2 epochs			5 epochs			10 epochs		
	MLNB	BR	MMP	DMLPP	BR	MMP	DMLPP	BR	MMP	DMLPP	BR	MMP	DMLPP
IsErr ×100	99.25	52.46	46.08	37.58	45.89	40.78	33.31	40.97	34.27	30.41	37.67	32.52	29.58
OneErr ×100	99.28	44.61	39.42	29.41	37.46	34.27	25.60	32.09	27.62	23.04	28.86	25.83	22.40
RankLoss	7.785	19.30	2.749	1.109	15.12	2.294	0.999	12.10	2.199	0.961	10.17	1.783	0.953
Margin	35.89	96.16	15.47	6.271	77.31	13.30	5.690	62.98	12.30	5.478	53.82	10.20	5.436
AvgP	6.565	57.10	70.00	77.21	63.49	74.61	80.10	68.27	78.83	81.93	71.18	80.28	82.37

EUROVOC		1 epoch			2 epochs		5 epochs	
	MLNB	BR	MMP	DMLPP	BR	MMP	BR	MMP
IsErr ×100	99.58	98.57	98.84	97.92	98.19	97.47	97.23	95.96
OneErr ×100	99.99	48.69	75.89	35.50	41.50	54.41	37.30	40.15
RankLoss	22.88	40.35	3.906	2.779	35.46	4.350	30.96	4.701
Margin	1644.00	3230.68	597.59	433.89	3050.07	694.10	2842.63	761.24
AvgP	1.06	26.90	29.28	46.67	31.58	39.54	35.90	47.27

Table 2: Average losses for the three views on the data and for the different algorithms.

on subject matter for epoch 20, similar behavior for directory code). This partially confirms the results of Crammer and Singer (2003). They observed that after reaching a certain amount of training examples the improvement stops and after that point the performance can even become worse. This point seems to be reached for MMP at the latest at ten epochs on the subject matter and directory code data. It is also interesting to note the behavior on the EUROVOC data as the ranking losses RankLoss and Margin increases from the first epoch on whereas for the other losses it still decreases.

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 five 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. The price to pay for the BR algorithm 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.

The fact that in only approximately 4% of the cases a perfect classification is achieved and in only approx. 60% the top class is correctly predicted (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.

6.3. 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 6.3. (measured on the first cross validation split, trained for one epoch), together with the CPU-times on an AMD Dual Core Opteron 2000 MHz as additional reference information. 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. For the directory code the rate for MMP and BR more than doubles in correspondence with the increase in number of classes, 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 4.), 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 21 weights set per instance out of 443 ($= L(K - L)$) in the worst case (a ratio of 4.47%), and on average 5.1% of them are required when a new training example arrives. For the directory code with a smaller fraction L/K 35.5 weights are stored (3.96%), of which only 1.11% are used when updating. This also explains the relatively small number of operations for training on EUROVOC, since from the 1,802 weights per instance (8.41%), 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.

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 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 (Loza Mencía and Fürnkranz, 2007).

subject matter	training	testing
BR	35.8 s	8.36 s
	1,675 M op.	192 M op.
MMP	31.35 s	6.28 s
	1,789 M op.	192 M op.
DMLPP	326.02 s	145.67 s
	6,089 M op.	4,628 M op.

directory code	training	testing
BR	49.01 s	11.99 s
	3,410 M op.	394 M op.
MMP	49.63 s	11.03 s
	3,579 M op.	394 M op.
DMLPP	313.59 s	192.99 s
	2,986 M op.	5,438 M op.

EUROVOC	training	testing
BR	405.42 s	56.71 s
	32,975 M op.	3,817 M op.
MMP	503.04 s	53.69 s
	40,510 M op.	3,817 M op.
DMLPP	11,479.81 s	7,631.86 s
	17,719 M op.	127,912 M op.

Table 3: Computational costs in CPU-time and millions of real value operations (M op.)

7. Conclusions

In this paper, we evaluated two known approaches for efficiently solving multilabel classification tasks on a large-scale text classification problem taken from the legal domain: the EUR-Lex database. The experimental results confirm that the MMP algorithm, which improves the more commonly used one-against-all or binary relevance approach by employing a concerted training protocol for the classifier ensemble, is very competitive and well applicable in practice for solving large-scale multilabel problems.

The average precision rate for the EUROVOC classification task, a multilabel classification task with 4000 possible labels, approaches 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.

In addition we presented an algorithm that reformulates the pairwise decomposition approach to a dual form so that it is capable to handle very complex problems and therefore to compete with the approaches which use 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 outperformed all other algorithms. This confirms previous results of the non-dual variant on the large Reuters Corpus Volume 1 (Loza Mencía and Fürnkranz, 2008). 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. We are currently investigating variants to further reduce the computational complexity. 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.

For future research, on the one hand we see space for improvement for the MMP and pairwise approach for instance by using a calibrated ranking approach (Brinker et al., 2006). The basic idea of this algorithm is to introduce an artificial label which, for each example, separates the relevant from irrelevant labels in order to return a set of classes instead of only a ranking. On the other hand, we see possible improvements by exploiting advancements in the perceptron algorithm and in the pairwise binarization, e.g. by using one of the several variants of the perceptron algorithm that, similar to SVMs, try to maximize the margin of the separating hyperplane in order to produce more accurate models (Crammer et al., 2006; Khardon and Wachman, 2007), or by employing a voting technique that takes the prediction weights into account such as the weighted voting technique by Price et al. (1995).

Acknowledgements

This work was supported by the EC 6th framework project ALIS (Automated Legal Information System).

8. References

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

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.

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

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.

Yoav Freund and Robert E. Schapire. Large Margin Classification using the Perceptron Algorithm. *Machine Learning*, 37(3): 277–296, 1999.

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

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.

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

Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Handwritten Digit Recognition by Neural Networks with Single-Layer Training. *IEEE Transactions on Neural Networks*, 3(6):962–968, 1992.

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.

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.

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 (to appear)*, 2008.

David Price, Stefan Knerr, Leon Personnaz, and Gerard Dreyfus. Pairwise Neural Network Classifiers with Probabilistic Outputs. In *Advances in Neural Information Processing Systems*, volume 7, pages 1109–1116. The MIT Press, 1995.

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

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

Shai Shalev-Shwartz and Yoram Singer. A New Perspective on an Old Perceptron Algorithm. In *Learning Theory, 18th Annual Conference on Learning Theory (COLT 2005)*, pages 264–278. Springer, 2005.

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. ISBN 1-55860-486-3.