Efficient Pairwise Multilabel Classification 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 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.

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, 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,

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 50–65, 2008.

- 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.

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.

We must note that in this work we do not solve the entire multilabel classification problem, but, following [3], we only provide a ranking of all possible labels. There are three reasons for this: (i) the MMP and the pairwise method naturally provide such a ranking, (ii) the ranking allows to evaluate the performance differences on a finer scale, and (iii) our key motivation is to study the scalability of these approaches which is determined by the rankings. However, there are several methods for finding a delimiter between relevant and irrelevant labels within a provided ranking of the labels, a good overview can be found in [17]. For the pairwise approach, 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 text categorization task [7].

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.

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, ...

Fig. 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.

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. They are available in most of the languages of the EU. 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 19,596 documents. Only documents related to secondary law (in contrast to primary law, the constitutional treaties of the European Union) and international agreements are included. 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, authors, etc. 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[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

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

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

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

Directory of Community legislation in force. It contains 20 chapter headings with up to four sub-division levels.

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.

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.

3 Preliminaries

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

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 $\mathcal{Y}$, a negative label set $\overline{\mathcal{Y}} = \mathcal{Y} \backslash \mathcal{Y}$ 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 $c \in \mathcal{Y}$ and all irrelevant classes $c' \in \overline{\mathcal{Y}}$. 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.

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

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.

$$F \stackrel{\text{def}}{=} \{c \in y \mid r(c) > r(c'), c' \in \overline{y}\} \cup \{c' \in \overline{y} \mid r(c) > r(c'), c \in y\} \tag{2}$$

$$\delta_{\text{Margin}} \stackrel{\text{def}}{=} \max(0, \max\{r(c) \mid c \in y\} - \min\{r(c') \mid c' \notin y\}) \tag{3}$$

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.

$$\delta_{\text{AvgP}} \stackrel{\text{def}}{=} 1 - \frac{1}{y} \sum_{c \in y} \frac{|\{c^* \in y \mid r(c^*) \leq r(c)\}|}{r(c)}. \tag{4}$$

3.2 Perceptrons

We use the simple but fast perceptrons as base classifiers [16]. 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:

$$o'(\bar{\mathrm{x}}) = sgn(\bar{\mathrm{x}} \cdot \bar{\mathrm{w}}) \tag{5}$$

with the internal weight vector $\bar{\mathrm{w}}$ and $sgn(t) = 1$ for $t \geq 0$ and -1 otherwise. If there exists a *separating hyperplane* between the two set of points, i.e. they are linearly separable, the following update rule provably finds it (cf., e.g., [1]).

$$\alpha_i = (y_i - o'(\bar{\mathrm{x}}_i)) \qquad \bar{\mathrm{w}}_{i+1} = \bar{\mathrm{w}}_i + \alpha_i \bar{\mathrm{x}}_i \tag{6}$$

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

$$\bar{\mathrm{w}} = \sum_{i=1}^{M} \alpha_i \bar{\mathrm{x}}_i \qquad o'(\bar{\mathrm{x}}) = sgn(\sum_{i=1}^{M} \alpha_i \cdot \bar{\mathrm{x}}_i \bar{\mathrm{x}}) \tag{7}$$

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

Require: Training example pair $(\bar{\mathrm{x}}, \mathcal{Y})$, perceptrons $\bar{\mathrm{w}}_1, \ldots, \bar{\mathrm{w}}_K$
calculate $\bar{\mathrm{x}}\bar{\mathrm{w}}_1, \ldots, \bar{\mathrm{x}}\bar{\mathrm{w}}_K$, loss δ
if $\delta > 0$ **then** ▷ only if ranking is not perfect
  calculate error sets E, F
  for each $c \in F$ **do** $\tau_c \leftarrow 0$, $\sigma \leftarrow 0$ ▷ initialize τ's, σ
  for each $(c, c') \in E$ **do**
    $p \leftarrow \textsc{Penalty}(\bar{\mathrm{x}}\bar{\mathrm{w}}_1, \ldots, \bar{\mathrm{x}}\bar{\mathrm{w}}_K)$
    $\tau_c \leftarrow \tau_c + p$ ▷ push up pos. classes
    $\tau_{c'} \leftarrow \tau_{c'} - p$ ▷ push down neg. classes
    $\sigma \leftarrow \sigma + p$ ▷ for normalization
  for each $c \in F$ **do**
    $\bar{\mathrm{w}}_c \leftarrow \bar{\mathrm{w}}_c + \delta \frac{\tau_c}{\sigma} \cdot \bar{\mathrm{x}}$ ▷ update perceptrons
return $\bar{\mathrm{w}}_1 \ldots \bar{\mathrm{w}}_K$ ▷ return updated perceptrons

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

the dual form and is crucial for developing the memory efficient variant in Section 4. The main reason for choosing the 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 [10], without forfeiting too much accuracy though SVMs find the *maximum-margin hyperplane* [5, 3, 18].

3.3 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 setting $y_{i_j} = 1$ if $c_j \in \mathcal{Y}_i$ and $y_{i_j} = -1$ otherwise. Supposing we use perceptrons as base learners, K different o'_j classifiers 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 desire a class ranking, we simply use the inner products and obtain a vector $\bar{\mathrm{o}}(\bar{\mathrm{x}}) = (\bar{\mathrm{x}}\bar{\mathrm{w}}_1, \ldots, \bar{\mathrm{x}}\bar{\mathrm{w}}_K)$. Ties are broken randomly to not favor any particular class.

3.4 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

Require: Training example pair $(\bar{\mathrm{x}}, \mathcal{Y})$,
perceptrons $\{\bar{\mathrm{w}}_{u,v} \mid u < v, c_u, c_v \in \mathcal{Y}\}$
for each $(c_u, c_v) \in \mathcal{Y} \times \overline{\mathcal{Y}}$ **do**
  if $u < v$ **then**
    $\bar{\mathrm{w}}_{u,v} \leftarrow \textsc{TrainPerceptron}(\bar{\mathrm{w}}_{u,v}, (\bar{\mathrm{x}}, 1))$ ▷ train as positive example
  else
    $\bar{\mathrm{w}}_{v,u} \leftarrow \textsc{TrainPerceptron}(\bar{\mathrm{w}}_{v,u}, (\bar{\mathrm{x}}, -1))$ ▷ train as negative example
return $\{\bar{\mathrm{w}}_{u,v} \mid u < v, c_u, c_v \in \mathcal{Y}\}$ ▷ updated perceptrons

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

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 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.

3.5 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., $(u, v) \in \mathcal{Y} \times \overline{\mathcal{Y}}$ (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 $\frac{K(K-1)}{2}$ 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 c_1 receives one vote and class c_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 $\bar{\mathrm{x}}$ belongs to c_1 or c_2 since it was neither trained on this example

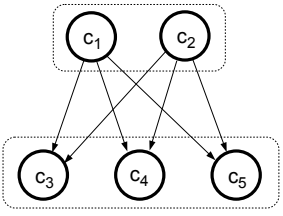

Fig. 4. MLPP training: training example $\bar{x}$ belongs to $\mathcal{Y} = \{c_1, c_2\}$, $\overline{\mathcal{Y}} = \{c_3, c_4, c_5\}$ are the irrelevant classes, the arrows represent the trained perceptrons

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

Fig. 5. 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.

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 class, 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 $K - |\mathcal{Y}|$ and a negative at most $K - |\mathcal{Y}| - 1$ votes. Class c_3 in Figure 5 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 [6, 8]. 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-RCV1 corpus [10], 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 the next section.

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 7. 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:

$$\begin{aligned}
\bar{\mathrm{w}}_{1,2}\bar{\mathrm{x}} &= \alpha_{1,2}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,2}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,2}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
\bar{\mathrm{w}}_{1,3}\bar{\mathrm{x}} &= \alpha_{1,3}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,3}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,3}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
&\vdots \\
\bar{\mathrm{w}}_{1,K}\bar{\mathrm{x}} &= \alpha_{1,K}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{1,K}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{1,K}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
\bar{\mathrm{w}}_{2,3}\bar{\mathrm{x}} &= \alpha_{2,3}^{1}\bar{\mathrm{x}}_1\bar{\mathrm{x}} + \alpha_{2,3}^{2}\bar{\mathrm{x}}_2\bar{\mathrm{x}} + \ldots + \alpha_{2,3}^{M}\bar{\mathrm{x}}_M\bar{\mathrm{x}} \\
&\vdots
\end{aligned} \tag{8}$$

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 6 and 7. 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.

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 6 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.

5 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

Require: New training example pair $(\bar{\mathrm{x}}_M, \mathcal{Y})$,
training examples $\bar{\mathrm{x}}_1 \ldots \bar{\mathrm{x}}_{M-1}$,
weights $\{\alpha^i_{u,v} \mid c_u, c_v \in \mathcal{Y}, 0 < i < M\}$

for each $\bar{\mathrm{x}}_i \in \bar{\mathrm{x}}_1 \ldots \bar{\mathrm{x}}_{M-1}$ **do**
  $p_i \leftarrow \bar{\mathrm{x}}_i \cdot \bar{\mathrm{x}}_M$
  for each $(c_u, c_v) \in \mathcal{Y} \times \overline{\mathcal{Y}}$ **do**
    if $\alpha^i_{u,v} \neq 0$ **then**
      $s_{u,v} \leftarrow s_{u,v} + \alpha^i_{u,v} \cdot p_t$ ▷ note that $s_{u,v} = -s_{v,u}$
for each $(c_u, c_v) \in \mathcal{Y} \times \overline{\mathcal{Y}}$ **do**
  if $s_{u,v} < 0$ **then**
    $\alpha^M_{u,v} \leftarrow 1$ ▷ note that $\alpha_{u,v} = -\alpha_{v,u}$
return $\{\alpha^M_{u,v} \mid (c_u, c_v) \in \mathcal{Y} \times \overline{\mathcal{Y}}\}$ ▷ return new weights

Fig. 6. Pseudocode of the training method of the DMLPP algorithm

Require: example $\bar{\mathrm{x}}$ for classification,
training examples $\bar{\mathrm{x}}_1 \ldots \bar{\mathrm{x}}_{M-1}$,
weights $\{\alpha^i_{u,v} \mid c_u, c_v \in \mathcal{Y}, 0 < i < M\}$

for each $\bar{\mathrm{x}}_i \in \bar{\mathrm{x}}_1 \ldots \bar{\mathrm{x}}_M$ **do**
  $p \leftarrow \bar{\mathrm{x}}_i \cdot \bar{\mathrm{x}}$
  for each $(c_u, c_v) \in \mathcal{Y}_i \times \overline{\mathcal{Y}}_t$ **do**
    if $\alpha^i_{u,v} \neq 0$ **then**
      $s_{u,v} \leftarrow s_{u,v} + \alpha^i_{u,v} \cdot p$
for each $(c_u, c_v) \in \mathcal{Y} \times \mathcal{Y}$ **do**
  if $u \neq v \vee s_{u,v} > 0$ **then**
    $v_u \leftarrow v_u + 1$
return voting $\bar{v} = (v_1, \ldots, v_{|\mathcal{Y}|})$ ▷ return voting

Fig. 7. Pseudocode of the prediction phase of the DMLPP algorithm

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 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 $\mathcal{Y} \times \overline{\mathcal{Y}}$. Furthermore, it has to intermediately store the resulting scores for each base perceptron during prediction, hence the complexity is $O(MLK + MN' + K^2) = O(M(LK + N') + K^2)$.[4] We can see that MLPP is

[4] 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).

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*, $\hat{\delta}_{per}$, $\hat{\delta}_{\text{ISERR}} \leq 1$, $\hat{\delta}_{\text{MARGIN}} < K$.

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

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 training 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 4.

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 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.

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.

6.1 Ranking Performance

The results for the four algorithms and the three different classifications of EUR-Lex are presented in Table 2. MLPP results are omitted since they are equal to those of DMLPP. 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 two epochs due to time restrictions.

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 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 6.2). Please refer to [13] for a more detailed comparison between MMP and BR.

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.

6.2 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 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 included in this table also the results for the non-dual MLPP, however no values have been received for the EUROVOC problem due to the discussed memory space problem: MLPP requires 539 MB memory for the subject matter and already 1825 MB for the directory code problem, whereas DMLPP consumes 203 MB and resp. 217 MB It is remarkable that MMP uses similar MMP 151 MB resp. 165MB. Also for EUROVOC the usage of 1143 MB is comparable to DMLPP's 2246 MB.

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

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

		1 epoch			2 epochs			5 epochs			10 epochs		
		MLNB	BR	MMP	DMLPP	BR	MMP	DMLPP	BR	MMP	DMLPP	BR	MMP	DMLPP
subject matter	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	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	IsErr ×100	99.58	98.57	98.84	97.92	98.19	97.47	96.64	97.23	95.96				
	OneErr ×100	99.99	48.69	75.89	35.50	41.50	54.41	29.50	37.30	40.15				
	RankLoss	22.88	40.35	3.906	2.779	35.46	4.350	2.504	30.96	4.701				
	Margin	1644.00	3230.68	597.59	433.89	3050.07	694.10	397.11	2842.63	761.24				
	AvgP	1.06	26.90	29.28	46.67	31.58	39.54	52.27	35.90	47.27				

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

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.
MLPP	91.77 s	196.93 s
	3,870 M op.	19,210 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.
MLPP	149.49 s	775.79 s
	4,712 M op.	80,938 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.
MLPP	–	–
	–	–

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 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 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. A deeper analysis of the contrary behavior of both algorithms when the number of classes increases can be found in [11].

7 Conclusions

In this paper, we introduced the EUR-Lex text collection as a promising test bed for studies in text categorization. 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 decomposition 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. 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) and than convert it to the dual representation by means of the collected information during training the perceptrons. The use of SVMs during training is also an interesting option.

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 [2]. 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 [4, 9], or by employing a voting technique that takes the prediction weights into account such as the weighted voting technique by Price et al. [15]. Finally, we note that we are currently working on an adaptation of the efficient voting technique introduced in [14] to the multilabel case, of which a further significant reduction in classification time can be expected.

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

References

[1] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)

[2] Brinker, K., Fürnkranz, J., Hüllermeier, E.: A Unified Model for Multilabel Classification and Ranking. In: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006) (2006)

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

[4] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585 (2006)
[5] Freund, Y., Schapire, R.E.: Large Margin Classification using the Perceptron Algorithm. Machine Learning 37(3), 277–296 (1999)
[6] Fürnkranz, J.: Round Robin Classification. Journal of Machine Learning Research 2, 721–747 (2002)
[7] Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning (to appear, 2008)
[8] Hsu, C.-W., Lin, C.-J.: A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural Networks 13(2), 415–425 (2002)
[9] Khardon, R., Wachman, G.: Noise tolerant variants of the perceptron algorithm. Journal of Machine Learning Research 8, 227–248 (2007)
[10] Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A New Benchmark Collection for Text Categorization Research. Journal of Machine Learning Research 5, 361–397 (2004)
[11] Loza Mencía, E., Fürnkranz, J.: An evaluation of efficient multilabel classification algorithms for large-scale problems in the legal domain. In: LWA 2007: Lernen -Wissen - Adaption, Workshop Proceedings, pp. 126–132 (2007)
[12] Loza Mencía, E., Fürnkranz, J.: Pairwise learning of multilabel classifications with perceptrons. In: Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN 2008), Hong Kong (2008)
[13] Loza Mencía, E., Fürnkranz, J.: 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, Marrakech, Morocco (2008)
[14] Park, S.-H., Fürnkranz, J.: Efficient pairwise classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 658–665. Springer, Heidelberg (2007)
[15] Price, D., Knerr, S., Personnaz, L., Dreyfus, G.: Pairwise Neural Network Classifiers with Probabilistic Outputs. In: Advances in Neural Information Processing Systems, vol. 7, pp. 1109–1116. MIT Press, Cambridge (1995)
[16] Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386–408 (1958)
[17] Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
[18] Shalev-Shwartz, S., Singer, Y.: A New Perspective on an Old Perceptron Algorithm. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 264–278. Springer, Heidelberg (2005)