An Evaluation of Efficient Multilabel Classification Algorithms for Large-Scale Problems in the Legal Domain

Eneldo Loza Mencía and Johannes Fürnkranz
Knowledge Engineering Group
Technische Universität Darmstadt
[loza,juffi]@ke.informatik.tu-darmstadt.de

Abstract

In this paper we evaluate the performance of multilabel classification algorithms on two classification tasks related to documents of the EUR-Lex database of legal documents of the European Union. It permits different settings of large-scale multilabel problems with up to 4000 classes with the same underlying documents. We compared the well known one-against-all approach (OAA) and its recently proposed improvement, 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. Both use the simple but very efficient perceptron algorithm as underlying classifier. This makes them very suitable for large-scale multilabel classification problems, in particular when the number of classes is high. Our results on the EUR-Lex database confirm that the MMP algorithm has a better response to an increasing number of classes than the one-against-all approach. We also show that it is principally possible 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 unrestricted set of classes instead of exactly one, have received increased attention in the literature. With the increased attention in recent times in this type of setting, 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,
- and the data is freely accessible.

The simplest strategy to tackle the multilabel problem with existing techniques is to use the *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 algorithm (MMP)* 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 as text classification linear discriminants are sufficiently expressive.

A shortcoming of the proposed 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 topic ranking task, which also enables a more detailed evaluation of the classification performance.

The MMP algorithm has already been used on large scale data sets such as the Reuters Corpus Volume 1 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 this paper we will analyze if these results can be repeated on two different settings of the EUR-Lex database, one with approx. 200 and the other with 4000 possible classes. Note that the latter problem has one order of magnitude more classes than other known applications of this algorithm.

2 Preliminaries

We represent an instance or object as a vector $\bar{\mathrm{x}} = (x_1, \ldots, x_N)$ in a feature space $\mathcal{X} \subseteq \mathbb{R}^N$. Each instance $\bar{\mathrm{x}}_i$ is assigned to a set of relevant labels $\mathcal{Y}_i$, 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 $y_i \in \{1, -1\}$, $\mathcal{Y}_i = \{y_i\}$.

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 $\bar{\mathrm{x}}$, a relevant label set $\mathcal{Y}$, a negative label set $\overline{\mathcal{Y}} = \mathcal{Y} \backslash \mathcal{Y}$ and a given predicted ranking $r(\bar{\mathrm{x}})$ 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.

ErrSetSize The error set size loss returns the number of pairs of labels which are not correctly ordered. As IsErr, it is 0 for a perfect ranking, but it additionally differentiates between different degrees of errors.

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

$$\delta_{\text{ErrSetSize}} \stackrel{\text{def}}{=} |E| \tag{2}$$

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.

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

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

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.

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

2.2 Perceptrons

A perceptron is a binary classifier initially developed as a model of the biological neuron [Rosenblatt, 1958]. Internally, it computes a linear combination of a real-valued input vector and predicts the positive class if the result is positive, and the negative class otherwise. More precisely, given an input vector $\bar{\mathrm{x}}$, the predicted class of a perceptron is computed as

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

with the weight vector $\bar{\mathrm{w}}$, threshold ω and $sgn(t) = 1$ for $t \geq 0$ and -1 otherwise. We can interpret a perceptron as a hyperplane with the formula $\bar{\mathrm{x}} \cdot \bar{\mathrm{w}} = -\omega$ that divides the N-dimensional space into two halves. An instance is a point in this space and its position determines its class membership. If the two sets of positive and negative points, respectively, can be separated by a hyperplane, they are called *linearly separable*. As a consequence, irrespective of the training algorithm used, linear classifiers as the perceptron cannot arrive at correct predictions for all potential instances unless the negative and positive instances are linearly separable. In order to find a possibly existing *separating hyperplane*, the weights are adapted according to the following perceptron training rule:

$$\begin{aligned} \theta_i &= (y_i - o'(\bar{\mathrm{x}}_i)) \\ \bar{\mathrm{w}}_{i+1} &= \bar{\mathrm{w}}_i + \eta \theta_i \bar{\mathrm{x}}_i \\ \omega_{i+1} &= \omega_i + \eta \theta_i \delta \end{aligned} \tag{7}$$

with δ usually being set to 1 and the initial weights set to zero without loss of generality. The learning rate η can be ignored if set to be constant [Bishop, 1995], as it will be the case in this work. When a N-dimensional point is misclassified, the hyperplane is moved towards this point (indicated by θ). If the training examples can be seen iteratively and the data is linearly separable, the algorithm provably finds a dividing hyperplane. This is called the perceptron convergence criterion (see, e.g., [Bishop, 1995]). Irrespective of training until convergence not always being desirable, this property does not reveal anything about the performance on unseen data.

Note that the number of errors until convergence depends on the margin between the positive and negative points. The hyperplane that maximizes the margin to the closest positive and negative point is called the *optimal hyperplane*. Contrary to support vector machines, perceptrons will not necessarily find an optimal hyperplane. However, the size of the margin is an indicator for the hardness of the learning problem: the smaller the margin the harder it is for the perceptron algorithm to find a good solution. On the other hand, perceptrons can be trained efficiently in an incremental setting, which makes them particularly well-suited for large-scale classification problems such as

the Reuters 2000 (RCV1) benchmark [Lewis et al., 2004]. For this reason, the perceptron has recently received increased attention [Freund and Schapire, 1999, Li et al., 2002, Shalev-Shwartz and Singer, 2005, Dekel et al., 2005].

Certainly, the δ value becomes important when the perceptron is trained in only one epoch: it is easily shown that $|\bar{w}| \leq |\mathcal{W}| \cdot \max_{\bar{x} \in \mathcal{W}} |\bar{x}|$ and $|\omega| \leq |\mathcal{W}| \cdot \delta$ holds for misclassified training examples $\mathcal{W} = \{\bar{x} \mid o'(\bar{x}) \neq y\}$. A disproportion between $\max_{\bar{x}} |\bar{x}|$ and δ can obviously lead to an excessive predominance of the threshold and make the scalar product even superfluous. To circumvent the problem of determining the right value for δ, we can set it to zero sacrificing one dimension in the hypothesis space (thus ω=0). Graphically this means that only separating hyperplanes through the origin are considered, reducing the number of potentially solvable problems. In practice, especially in high dimensional spaces as for text documents, this is usually not a very significant restriction, and it additionally renders online learning possible.

2.3 Binary Relevance Ranking

In the binary relevance 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 $(\bar{x}_i, \mathcal{Y}_i)$ in the original training set K different pairs of instances and binary class assignments $(\bar{x}_i, y_{i_j})$ with $j = 1 \ldots K$ are generated as follows:

$$y_{i_j} = \begin{cases} 1 & c_j \in \mathcal{Y}_i \\ -1 & otherwise \end{cases} \tag{8}$$

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 one-against-all classifier for an instance $\bar{x}$ would be the set $\{c_j \mid o'_j(\bar{x}) = 1\}$. 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 $o_j(\bar{x}) = \bar{x} \cdot \bar{w}_j$ (ignoring ω as mentioned above). So the result of the prediction is a vector $\bar{o}(\bar{x}) = (\bar{x}\bar{w}_1, \ldots, \bar{x}\bar{w}_K)$ where component j corresponds to the relevance of class c_j. We will denote the ranking function that returns the position of class c in the ranking with $r(c) \in \{1 \ldots K\}$. Ties are broken randomly to not favor any particular class.

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 one-against-all, 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 one-against-all 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 $\bar{x}$, 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. This allows to optimize the ranking in accordance with the used ranking loss. 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. Crammer and Singer [2003] propose several methods, including a function that returns a value proportional to the difference of the scalar products of both classes. The most successful one, however, seemed to be the uniform update method, where each pair in E receives the same score. In the next step, the update weights τ are normalized and each perceptron whose class was wrongly ordered is updated.

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 δ
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 τ's
5: **for each** $(c, c') \in E$ **do**
6: $p \leftarrow \text{PENALTY}(\bar{x}\bar{w}_1, \ldots, \bar{x}\bar{w}_K)$
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 τ'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

An example will illustrate the peculiarities of the MMP update method: Suppose that all classes are correctly ordered except for one relevant and three irrelevant classes. The three negative classes are ranked immediately over the positive. The error set contains three wrongly ordered pairs and according to the uniform update method the positive class will receive in the sum a penalty of 3 and the negatives each 1. Thus the perceptron of the positive class will be updated to a degree three times as great compared with the other three, in accordance with the degree to which it contributed to the wrong ranking. Note that regardless of the used penalty function the positive and the negative classes receive in total the same penalty scores and these are afterwards normalized, so that the degree of the overall model update only depends on δ, i.e. on the quality of the ranking. More precisely, the hyperplanes of the perceptrons of the relevant classes are translated by a total amount of $\delta\,\bar{x}$, and the remaining classes by $-\,\delta\,\bar{x}$. In summary, the degree of the update for a particular perceptron depends 1) on the used penalty method, 2) on how much it contributed to the wrong ranking, and 3) on the general ranking performance.

3 EUR-Lex Repository

The EUR-Lex/CELEX (Communitatis Europeae LEX) Site[1] provides a freely accessible repository for European Union law texts. The accessible documents include the official Journal of the European Union, treaties, international

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

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

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;

Whereas computer programs are playing an increasingly important role in a broad range of industries and computer program technology can accordingly be considered as being of fundamental importance for the Community's industrial development;

...

Figure 2: 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 and the subject matters based on the text of the document.

agreements, legislation in force, legislation in preparation, case-law and parliamentary questions. The documents are available in 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 documents in the English version of the *Directory of Community legislation in force*[2], in total 19,601 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. We restricted our view to the EUROVOC and the subject matter classifications. EUROVOC is a multilingual thesaurus providing a controlled vocabulary for European Institutions[3]. Documents in the documentation systems of the EU are indexed using this thesaurus.

Figure 2 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.

3,993 different EUROVOC descriptors were identified in the retrieved documents, each document is associated to 5.37 descriptors in average. In contrast there are only 201 different subject matters appearing in the dataset, with a mean of 2.23 labels per document.

3.1 Data Preprocessing

The main text was extracted from the HTML documents, excluding HTML tags, bibliographic notes or other additio-

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

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

	1 epoch		2 epochs		10 epochs		20 epochs	
	OAA	MMP	OAA	MMP	OAA	MMP	OAA	MMP
IsErr $\times 100$	59.44	49.31	52.35	43.17	46.92	37.32	45.71	37.34
OneErr $\times 100$	27.18	22.96	21.05	18.08	17.04	14.13	16.50	14.33
ErrSetSize	81.64	12.57	62.64	11.12	50.07	8.825	46.79	8.356
Margin	59.51	10.59	47.40	9.503	38.99	7.643	36.54	7.289
AvgP	64.59	77.95	71.18	81.73	75.21	85.13	76.07	85.25

Table 1: Average losses for the *subject matter* classification.

	1 epoch		2 epochs		5 epochs	
	OAA	MMP	OAA	MMP	OAA	MMP
IsErr $\times 100$	98.82	98.43	98.03	97.03	96.82	95.24
OneErr $\times 100$	47.29	70.14	40.52	50.37	34.78	36.47
ErrSetSize	8422.0	807.2	7320.3	926.7	6530.9	995.6
Margin	3214.0	582.4	2981.3	701.0	2794.1	756.9
AvgP	28.01	31.01	33.43	42.07	37.71	49.91

Table 2: Average losses for the *EUROVOC descriptor* classification.

nal 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.[4] 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 10,000 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.

4 Evaluation

4.1 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. Both algorithms use perceptrons without thresholds, as described in Section 2.2, and all perceptrons were initialized with random values.

4.2 Ranking Performance

The results of a direct comparison of OAA and MMP for the subject matter and EUROVOC descriptor classifications are presented in Table 1 and Table 2. The values for IsErr, OneErr 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.

For the subject matter, the results clearly show that the MMP algorithm outperforms the simple one-against-all approach (all differences are statistically significant). 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. Note also that the MMP algorithm is not able to improve its performance after 10 epochs. This partially confirms the results of Crammer and Singer. They observed that after reaching a certain amount of training examples, the improvement stops and after that point the performance even becomes worse. This point seems to be reached at the latest at ten epochs on the EUR-Lex data for subject matter classification.

The results on the EUROVOC descriptor data set confirm the previous results. The differences in ErrSetSize 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 fact that in only approximately 5% of the cases a perfect classification is achieved and in only approx. 65% the top class is correctly predicted should not lead to an underestimation of the performance of the two algorithms. Considering that with almost 4000 possible classes and only 5.3 classes per example the probability to guess a correct class is less than one percent, namely 0.13%, the performance is indeed substantial.

4.3 Computational Costs

In order to allow a comparison independent from external factors such as logging activities and the run-time environment, we measured the computational cost in terms of vector additions and scalar multiplications. We also ignored minor operations that have to be performed by both algorithms, such as sorting or internal real value operations. An overview is given in Table 4.3, together with the CPU-times that were measured on a AMD Dual Core Opteron 2000

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

subject	training	testing
OAA	48.74 s 3,545,841 mult. + 44,113 add.	5.42 s 393,960 mult.
MMP	54.08 s 3,545,841 mult. + 304,245 add.	5.39 s 393,960 mult.

EUROVOC	training	testing
OAA	621.814 s 70,440,513 mult. + 224,615 add.	98.354 s 7,826,280 mult.
MMP	818.147 s 70,440,513 mult. + 15,305,659 add.	93.467 s 7,826,280 mult.

Table 3: Computational costs in CPU-time and vector multiplications and additions on both data sets.

MHz for additional reference information.

As per design, for both algorithms the number of scalar multiplications is equal, namely 3,545,841 for each training iteration and 393,960 for testing on the subject matter data and 70,440,513 and 7,826,280 respectively for the EUROVOC data. In contrast, the number of vector addition operations differ: while the one-against-all method requires 44,113 operations for the subject matter and 224,615 for the EUROVOC classifications (for the first iteration, cross-validated), the MMP has higher costs with 304,245 and 15,305,659 operations respectively.

If we analyze the number of perceptron updates, i.e. additions, as a function of the number of classes, it is interesting to see the contrary behavior of both algorithms: while the one-against-all algorithm reduces the ratio of updated perceptrons per training example from 1.23% to 0.32% when increasing the number of classes from 202 to 3993, the MMP algorithm doubles the rate from 9.08% to 21,81%.

For the MMP this behavior is natural: with increasing numbers of classes the error set size increases and as a consequence the number of updated perceptrons. The MMP adapts itself to the increased scale and complexity. An explanation for the one-against-all case could be that due to the decreased number of positive examples for each base classifier (in average 23 = classes per example / (total number of classes * number of training examples) for the EUROVOC classes) the perceptrons quickly adopt the generally good rule to always return a negative score, which leads to only a few errors and consequently to little corrective updates. Note that a base classifier that always predicts the negative class would make approx. 95,000 errors on the training set, compared to the 224,615 of the perceptrons in the training phase.

5 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 (OAA) 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 increase in predictive performance has to be paid by the MMP algorithm with a small increase in computational complexity that in our opinion is not important in practice.

The average precision rate for the EUROVOC classification task, a multilabel classification task with 4000 possible labels, approaches 50%. Rougly 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.

For future research, on the one hand we see space for improvement and extension of the MMP algorithm for example 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 class, separates the relevant from irrelevant labels. On the other hand, we are in the process of evaluating different binarization approaches such as pairwise learning [Fürnkranz, 2002]. An evaluation of this pairwise approach on the Reuters Corpus Volume 1 [Lewis et al., 2004], which contains over 100 classes and 800,000 documents, showed a substantial improvement over the MMP method [Loza Mencía and Fürnkranz, 2007]. The main obstacle to the applicability is the large memory requirement of the approximately 8,000,000 perceptrons that are needed to represent such an ensemble. We are currently investigating ways for reducing that number without losing the effectiveness of pairwise approaches. Furthermore, we also need to compare our performance to those of established algorithms that are potentially capable of handling large-scale multilabel data, such as the Naive Bayes algorithm.

Acknowledgements

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

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.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The Forgetron: A Kernel-Based Perceptron on a Fixed Budget. In *Advances in Neural Information Processing Systems 18*, 2005.

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.

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.

Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S. Kandola. The Perceptron Algorithm

with Uneven Margins. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002)*, pages 379–386, 2002.

Eneldo Loza Mencía and Johannes Fürnkranz. Pairwise learning of multilabel classifications with perceptrons. Technical Report TUD-KE-2007-05, Technische Universität Darmstadt, Knowledge Engineering Group, 2007. http://www.ke.informatik.tu-darmstadt.de/publications/reports/tud-ke-2007-05.pdf.

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.