Large-Scale Multi-label Text Classification — Revisiting Neural Networks

Jinseok Nam[1,2], Jungi Kim[1], Eneldo Loza Mencía[1],
Iryna Gurevych[1,2], and Johannes Fürnkranz[1]

[1] Department of Computer Science, Technische Universität Darmstadt, Germany
[2] Knowledge Discovery in Scientific Literature,
German Institute for Educational Research, Germany

Abstract. Neural networks have recently been proposed for multi-label classification because they are able to capture and model label dependencies in the output layer. In this work, we investigate limitations of BP-MLL, a neural network (NN) architecture that aims at minimizing pairwise ranking error. Instead, we propose to use a comparably simple NN approach with recently proposed learning techniques for large-scale multi-label text classification tasks. In particular, we show that BP-MLL's ranking loss minimization can be efficiently and effectively replaced with the commonly used cross entropy error function, and demonstrate that several advances in neural network training that have been developed in the realm of deep learning can be effectively employed in this setting. Our experimental results show that simple NN models equipped with advanced techniques such as rectified linear units, dropout, and AdaGrad perform as well as or even outperform state-of-the-art approaches on six large-scale textual datasets with diverse characteristics.

1 Introduction

As the amount of textual data on the web and in digital libraries is increasing rapidly, the need for augmenting unstructured data with metadata is also increasing. Time- and cost-wise, a manual extraction of such information from ever-growing document collections is impractical.

Multi-label classification is an automatic approach for addressing such problems by learning to assign a suitable subset of categories from an established classification system to a given text. In the literature, one can find a number of multi-label classification approaches for a variety of tasks in different domains such as bioinformatics [1], music [28], and text [9]. In the simplest case, multi-label classification may be viewed as a set of binary classification tasks that decides for each label independently whether it should be assigned to the document or not. However, this so-called *binary relevance* approach ignores dependencies between the labels, so that current research in multi-label classification concentrates on the question of how such dependencies can be exploited [23, 3]. One such approach is BP-MLL [32], which formulates multi-label classification problems as a neural network with multiple output nodes, one for each label. The output layer is able to model dependencies between the individual labels.

In this work, we directly build upon BP-MLL and show how a simple, single hidden layer NN may achieve a state-of-the-art performance in large-scale multi-label text classification tasks. The key modifications that we suggest are (i) more efficient and more effective training by replacing BP-MLL's pairwise ranking loss with cross entropy and (ii) the use of recent developments in the area of deep learning such as rectified linear units (ReLUs), Dropout, and AdaGrad.

Even though we employ techniques that have been developed in the realm of deep learning, we nevertheless stick to single-layer NNs. The motivation behind this is two-fold: first, a simple network configuration allows better scalability of the model and is more suitable for large-scale tasks on textual data[1]. Second, as it has been shown in the literature [15], popular feature representation schemes for textual data such as variants of *tf-idf* term weighting already incorporate a certain degree of higher dimensional features, and we speculate that even a single-layer NN model can work well with text data. This paper provides an empirical evidence to support that a simple NN model equipped with recent advanced techniques for training NN performs as well as or even outperforms state-of-the-art approaches on large-scale datasets with diverse characteristics.

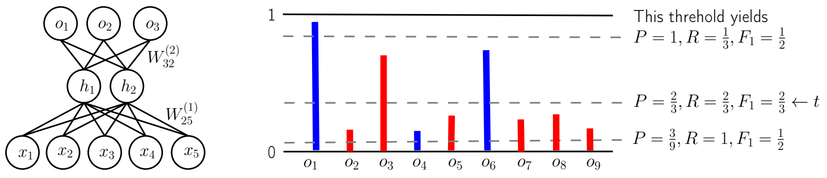
2 Multi-label Classification

Formally, multi-label classification may be defined as follows: $X \subset \mathbb{R}^D$ is a set of M instances, each being a D-dimensional feature vector, and L is a set of labels. Each instance $\mathbf{x}$ is associated with a subset of the L labels, the so-called *relevant* labels; all other labels are *irrelevant* for this example. The task of the learner is to learn a mapping function $f : \mathbb{R}^D \to 2^L$ that assigns a subset of labels to a given instance. An alternative view is that we have to predict an L-dimensional target vector $\mathbf{y} \in \{0, 1\}^L$, where $y_i = 1$ indicates that the i-th label is relevant, whereas $y_i = 0$ indicates that it is irrelevant for the given instance.

Many algorithms have been developed for tackling this type of problem. The most straightforward way is binary relevance (BR) learning; it constructs L binary classifiers, which are trained on the L labels independently. Thus, the prediction of the label set is composed of independent predictions for individual labels. However, labels often occur together, that is, the presence of a specific label may suppress or exhibit the likelihood of other labels.

To address this limitation of BR, pairwise decomposition (PW) and label powerset (LP) approaches consider label dependencies during the transformation by either generating pairwise subproblems [9, 20] or the powerset of possible label combinations [29]. Classifier chains [23, 3] are another popular approach that extend BR by including previous predictions into the predictions of subsequent labels. [7] present a large-margin classifier, RankSVM, that minimizes a ranking loss by penalizing incorrectly ordered pairs of labels. This setting can be used for multi-label classification by assuming that the ranking algorithm has to rank each relevant label before each irrelevant label.

[1] Deep NNs, in fact, scale well and work effectively by learning features from raw inputs which are usually smaller than hand-crafted features extracted from the raw inputs. However, in our case, the dimensions of raw inputs are relatively large where training deep NNs is costly.


o_1 o_2 o_3
$W_{32}^{(2)}$
h_1 h_2
$W_{25}^{(1)}$
x_1 x_2 x_3 x_4 x_5
1
0
This threhold yields
$P = 1, R = \frac{1}{3}, F_1 = \frac{1}{2}$
$P = \frac{2}{3}, R = \frac{2}{3}, F_1 = \frac{2}{3} \leftarrow t$
$P = \frac{3}{9}, R = 1, F_1 = \frac{1}{2}$
o_1 o_2 o_3 o_4 o_5 o_6 o_7 o_8 o_9

(a) A neural network (b) Threshold decision

Fig. 1. (a) a neural network with a single hidden layer of two units and multiple output units, one for each possible label. (b) shows how threshold for a training example is estimated based on prediction output $\mathbf{o}$ of the network. Consider nine possible labels, of which o_1, o_4 and o_6 are relevant labels (blue) and the rest are irrelevant (red). The figure shows three exemplary threshold candidates (dashed lines), of which the middle one is the best choice because it gives the highest F1 score. See Section 3.3 for more details.

In order to make a prediction, the ranking has to be *calibrated* [9], i.e., a threshold has to be found that splits the ranking into relevant and irrelevant labels.

2.1 State-of-the-art Multi-label Classifiers and Limitations

The most prominent learning method for multi-label text classification is to use a BR approach with strong binary classifiers such as SVMs [24, 30] despite its simplicity. It is known that characteristics of high-dimensional and sparse data, such as text data, make decision problems linearly separable [15], and this characteristic suits the strengths of SVM classifiers well. Unlike benchmark datasets, real-world text collections consist of a large number of training examples represented in a high-dimensional space with a large amount of labels. To handle such datasets, researchers have derived efficient *linear* SVMs [16, 8] that can handle large-scale problems. However, their performance decreases as the number of labels grows and the label frequency distribution becomes skewed [19, 24]. In such cases, it is intractable to employ methods that minimize ranking errors among labels [7, 32] or that learn probability distributions of labels [11, 3].

3 Neural Networks for Multi-label Classification

In this section, we propose a neural network-based multi-label classification framework that is composed of a single hidden layer and operates with recent developments in neural network and optimization techniques, which allow the model to converge into good regions of the error surface in a few steps of parameter updates. Our approach consists of two modules (Fig. 1): a neural network that produces label scores (Sections 3.2–3.5), and a label predictor that converts label scores into binary (Section 3.3).

3.1 Rank Loss

The most intuitive objective for multi-label learning is to minimize the number of misordering between a pair of relevant label and irrelevant label, which is called *rank loss*:

$$L(\mathbf{y}, f(\mathbf{x})) = w(\mathbf{y}) \sum_{y_i < y_j} \mathbb{I}\left(f_i(\mathbf{x}) > f_j(\mathbf{x})\right) + \frac{1}{2}\mathbb{I}\left(f_i = f_j\right) \tag{1}$$

where $w(\mathbf{y})$ is a normalization factor, $\mathbb{I}(\cdot)$ is the indicator function, and $f_i(\cdot)$ is a prediction score for a label i. Unfortunately, it is hard to minimize due to non-convex property of the loss function. Therefore, convex surrogate losses have been proposed as alternatives to rank loss [26, 7, 32].

3.2 Pairwise Ranking Loss Minimization in Neural Networks

Let us assume that we would like to make a prediction on L labels from D dimensional input features. Consider the neural network model with a single hidden layer in which F hidden units are defined and input units $\mathbf{x} \in \mathbb{R}^{D\times 1}$ are connected to hidden units $\mathbf{h} \in \mathbb{R}^{F\times 1}$ with weights $\mathbf{W}^{(1)} \in \mathbb{R}^{F\times D}$ and biases $\mathbf{b}^{(1)} \in \mathbb{R}^{F\times 1}$. The hidden units are connected to output units $\mathbf{o} \in \mathbb{R}^{L\times 1}$ through weights $\mathbf{W}^{(2)} \in \mathbb{R}^{L\times F}$ and biases $\mathbf{b}^{(2)} \in \mathbb{R}^{L\times 1}$. The network, then, can be written in a matrix-vector form, and we can construct a feed-forward network $f_\Theta : \mathbf{x} \to \mathbf{o}$ as a composite of non-linear functions in the range $[0, 1]$:

$$f_\Theta(\mathbf{x}) = f_o\left(\mathbf{W}^{(2)} f_h\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}\right) \tag{2}$$

where $\Theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$, and f_o and f_h are *element-wise* activation functions in the output layer and the hidden layer, respectively. Specifically, the function $f_\Theta(\mathbf{x})$ can be re-written as follows:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \quad \mathbf{h} = f_h\left(\mathbf{z}^{(1)}\right)$$
$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)}, \quad \mathbf{o} = f_o\left(\mathbf{z}^{(2)}\right)$$

where $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ denote the weighted sum of inputs and hidden activations, respectively. Our aim is to find a parameter vector Θ that minimizes a cost function $J(\Theta; \mathbf{x}, \mathbf{y})$. The cost function measures discrepancy between predictions of the network and given targets $\mathbf{y}$.

BP-MLL [32] minimizes errors induced by incorrectly ordered pairs of labels, in order to exploit dependencies among labels. To this end, it introduces a *pairwise error function* (PWE), which is defined as follows:

$$J_{PWE}(\Theta; \mathbf{x}, \mathbf{y}) = \frac{1}{|\mathbf{y}||\bar{\mathbf{y}}|} \sum_{(p,n)\in \mathbf{y}\times\bar{\mathbf{y}}} \exp(-(o_p - o_n)) \tag{3}$$

where p and n are positive and negative label index associated with training example $\mathbf{x}$. $\bar{\mathbf{y}}$ represents a set of negative labels and $|\cdot|$ stands for the cardinality. The PWE is the relaxation of the loss function in Equation 1 that we want to minimize.

As no closed-form solution exists to minimize the cost function, we use a gradient-based optimization method.

$$\Theta^{(\tau+1)} = \Theta^{(\tau)} - \eta \nabla_{\Theta^{(\tau)}} J(\Theta^{(\tau)}; \mathbf{x}, \mathbf{y}) \tag{4}$$

The parameter Θ is updated by adding a small step of negative gradients of the cost function $J(\Theta^{(\tau)}; \mathbf{x}, \mathbf{y})$ with respect to the parameter Θ at step τ. The parameter η, called the learning rate, determines the step size of updates.

3.3 Thresholding

Once training of the neural network is finished, its output can be used to rank labels, but additional measures are needed in order to split the ranking into relevant and irrelevant labels. For transforming the ranked list of labels into a set of binary predictions, we train a multi-label threshold predictor from training data. This sort of thresholding methods are also used in [7, 32]

For each document $\mathbf{x}_m$, labels are sorted by the probabilities in decreasing order. Ideally, if NNs successfully learn a mapping function f_Θ, all correct (positive) labels will be placed on top of the sorted list and there should be a large margin between the set of positive labels and the set of negative labels. Using F_1 score as a reference measure, we calculate classification performances at every pair of successive positive labels and choose a threshold value t_m that produces the best performance (Figure 1 (b)).

Afterwards, we can train a multi-label thresholding predictor $\hat{\mathbf{t}} = T(\mathbf{x}; \theta)$ to learn $\mathbf{t}$ as target values from input pattern $\mathbf{x}$. We use linear regression with $\ell 2$-regularization to learn θ

$$J(\theta) = \frac{1}{2M} \sum_{m=1}^{M} \left(T(\mathbf{x}_m; \theta) - t_i\right)^2 + \frac{\lambda}{2} \|\theta\|_2^2 \tag{5}$$

where $\mathbf{x}_m$ is m-th document in the train data, $T(\mathbf{x}_m; \theta) = \theta^T \mathbf{x}_m$, and λ is a parameter which controls the magnitude of the $\ell 2$ penalty.

At test time, these learned thresholds are used to predict a binary output $\hat{y}_{kl}$ for label l of a test document $\mathbf{x}_k$ given label probabilities o_{kl}; $\hat{y}_{kl} = 1$ if $o_{kl} > T(\mathbf{x}_k; \theta)$, otherwise 0. Due to the fact that the resulting parameter θ might get biased to the training data, the control parameter λ needs to be tuned via cross-validation.

3.4 Ranking Loss vs. Cross Entropy

BP-MLL is supposed to perform better in multi-label problems since it takes label correlations into consideration than the standard form of NN that does not. However, we have found that BP-MLL does not perform as expected in our preliminary experiments, particularly, on datasets in textual domain.

Consistency w.r.t Rank Loss. Recently, it has been claimed that none of convex loss functions including BP-MLL's loss function (Equation 3) is consistent with respect to *rank loss* which is non-convex and has discontinuity [2, 10]. Furthermore, univariate surrogate loss functions such as *log loss* are rather consistent with rank loss [4].

$$J_{log}(\Theta; \mathbf{x}, \mathbf{y}) = w(\mathbf{y}) \sum_{l} \log\left(1 + e^{-\dot{y}_l z_l}\right)$$

where $w(\mathbf{y})$ is a weighting function that normalizes loss in terms of $\mathbf{y}$ and z_l indicates prediction for label l. Please note that the log loss is often used for logistic regression in which $\dot{y} \in \{-1, 1\}$ is the target and z_l is the output of a linear function

$z_l = \sum_k W_{lk} x_k + b_l$ where W_{lk} is a weight from input x_k to output z_l and b_l is bias for label l. A typical choice is, for instance, $w(\mathbf{y}) = (|\mathbf{y}||\bar{\mathbf{y}}|)^{-1}$ as in BP-MLL. In this work, we set $w(\mathbf{y}) = 1$, then the log loss above is equivalent to *cross entropy* (CE), which is commonly used to train neural networks for classification tasks if we use *sigmoid* transfer function in the output layer, i.e. $f_o(z) = 1/\left(1 + \exp(-z)\right)$, or simply $f_o(z) = \sigma\left(z\right)$:

$$J_{CE}(\Theta; \mathbf{x}, \mathbf{y}) = -\sum_l \left(y_l \log o_l) + (1 - y_l)\log(1 - o_l)\right) \tag{6}$$

where o_l and y_l are the prediction and the target for label l, respectively. Let us verify the equivalence between the *log loss* and the *CE*. Consider the log loss function for only label l.

$$J_{log}(\Theta; \mathbf{x}, y_l) = \log(1 + e^{-\dot{y}_l z_l}) = -\log\left(\frac{1}{1 + e^{-\dot{y}_l z_l}}\right) \tag{7}$$

As noted, $\dot{y}$ in the log loss takes either -1 or 1, which allows us to split the above equation as follows:

$$-\log\left(\frac{1}{1 + e^{-\dot{y}_l z_l}}\right) = \begin{cases} -\log\left(\sigma\left(z_l\right)\right) & \text{if } \dot{y} = 1 \\ -\log\left(\sigma\left(-z_l\right)\right) & \text{if } \dot{y} = -1 \end{cases} \tag{8}$$

Then, we have the corresponding CE by using a property of the sigmoid function $\sigma\left(-z\right) = 1 - \sigma\left(z\right)$

$$J_{CE}\left(\Theta; \mathbf{x}, y_l\right) = -\left(y_l \log o_l + (1 - y_l)\log\left(1 - o_l\right)\right) \tag{9}$$

where $y \in \{0, 1\}$ and $o_l = \sigma\left(z_l\right)$.

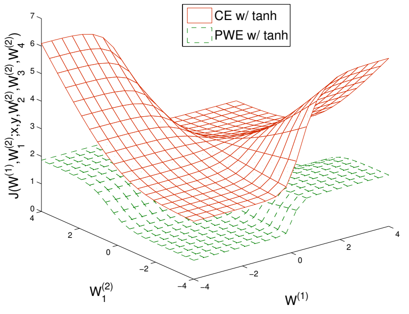
Computational Expenses. In addition to consistency with rank loss, CE has an advantage in terms of computational efficiency; computational cost for computing gradients of parameters with respect to PWE is getting more expensive as the number of labels grows. The error term $\delta_l^{(2)}$ for label l which is propagated to the hidden layer is defined as

$$\delta_l^{(2)} = \begin{cases} -\frac{1}{|\mathbf{y}||\bar{\mathbf{y}}|}\sum_{n \in \bar{\mathbf{y}}} \exp(-(o_l - o_n)) f_o'(z_l^{(2)}), & \text{if } l \in \mathbf{y} \\ \frac{1}{|\mathbf{y}||\bar{\mathbf{y}}|}\sum_{p \in \mathbf{y}} \exp(-(o_p - o_l)) f_o'(z_l^{(2)}), & \text{if } l \in \bar{\mathbf{y}} \end{cases} \tag{10}$$

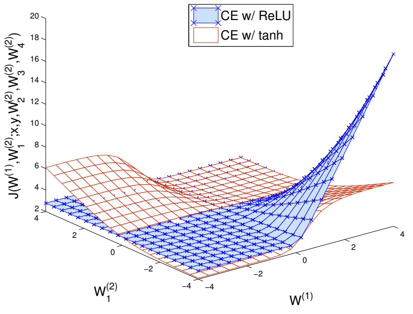
Whereas the computation of $\delta_l^{(2)} = -y_l/o_l + (1 - y_l)/(1 - o_l) f_o'(z_l^{(2)})$ for the CE can be performed efficiently, obtaining error terms $\delta_l^{(2)}$ for the PWE is L times more expensive than one in ordinary NN utilizing the cross entropy error function. This also shows that BP-MLL scales poorly w.r.t. the number of unique labels.

Plateaus. To get an idea of how differently both objective functions behave as a function of parameters to be optimized, let us draw graphs containing cost function values. Note that it has been pointed out that the slope of the cost function as a function of the parameters plays an important role in learning parameters of neural networks [27, 12].

Consider two-layer neural networks consisting of $W^{(1)} \in \mathbb{R}$ for the first layer, $\mathbf{W}^{(2)} \in \mathbb{R}^{4 \times 1}$ for the second, that is, the output layer. Since we are interested in function values with respect to two parameters $W^{(1)}$ and $W_1^{(2)}$ out of 5 parameters, $\mathbf{W}_{\{2,3,4\}}^{(2)}$



(a) Comparison of CE and PWE



(b) Comparison of $\tanh$ and ReLU, both for CE

Fig. 2. Landscape of cost functions and a type of hidden units. $W^{(1)}$ represents a weight connecting an input unit to a hidden unit. Likewise, $W_1^{(2)}$ denotes a weight from the hidden unit to output unit 1. The z-axis stands for a value for the cost function $J(W^{(1)}, W_1^{(2)}; \mathbf{x}, \mathbf{y}, W_2^{(2)}, W_3^{(2)}, W_4^{(2)})$ where instances $\mathbf{x}$, targets $\mathbf{y}$ and weights $W_2^{(2)}, W_3^{(2)}, W_4^{(2)}$ are fixed.

is set to a fixed value c. In this paper we use $c = 0$.[2] Figure 2 (a) shows different shapes of the functions and slope steepness. In figure 2 (a) both curves have similar shapes, but the curve for PWE has plateaus in which gradient descent can be very slow in comparison with the CE. Figure 2 (b) shows that CE with ReLUs, which is explained the next Section, has a very steep slope compared to CE with tanh. Such a slope can accelerate convergence speed in learning parameters using gradient descent. We conjecture that these properties might explain why our set-up converges faster than the other configurations, and BP-MLL performs poorly in most cases in our experiments.

3.5 Recent Advances in Deep Learning

In recent neural network and deep learning literature, a number of techniques were proposed to overcome the difficulty of learning neural networks efficiently. In particular, we make use of ReLUs, AdaGrad, and Dropout training, which are briefly discussed in the following.

Rectified Linear Units. *Rectified linear units* (ReLUs) have been proposed as activation units on the hidden layer and shown to yield better generalization performance [22, 13, 31]. A ReLU disables negative activation ($\text{ReLU}(x) = \max(0, x)$) so that the number of parameters to be learned decreases during the training. This sparsity characteristic makes ReLUs advantageous over the traditional activation units such as *sigmoid* and *tanh* in terms of the generalization performance.

[2] The shape of the functions is not changed even if we set c to arbitrary value since it is drawn by function values in z-axis with respect to only $W^{(1)}$ and $W_1^{(2)}$.

Learning Rate Adaptation with AdaGrad. *Stochastic gradient descent* (SGD) is a simple but effective technique for minimizing the objective functions of NNs (Equation 4). When SGD is considered as an optimization tool, one of the problems is the choice of the learning rate. A common approach is to estimate the learning rate which gives lower training errors on subsamples of training examples [17] and then decrease it over time. Furthermore, to accelerate learning speed of SGD, one can utilize momentum [25].

Instead of a fixed or scheduled learning rate, an *adaptive learning rate* method, namely AdaGrad, was proposed [6]. The method determines the learning rate at iteration τ by keeping previous gradients $\Delta_{1:\tau}$ to compute the learning rate for each dimension of parameters $\eta_{i,\tau} = \eta_0 / \sqrt{\sum_{t=1}^{\tau} \Delta_{i,t}^2}$ where i stands for an index of each dimension of parameters and η_0 is the initial learning rate and shared by all parameters. For multi-label learning, it is often the case that a few labels occur frequently, whereas the majority only occurs rarely, so that the rare ones need to be updated with larger steps in the direction of the gradient. If we use AdaGrad, the learning rates for the frequent labels decreases because the gradient of the parameter for the frequent labels will get smaller as the updates proceed. On the other hand, the learning rates for rare labels remain comparatively large.

Regularization using Dropout Training. In principle, as the number of hidden layers and hidden units in a network increases, its expressive power also increases. If one is given a large number of training examples, training a larger networks will result in better performance than using a smaller one. The problem when training such a large network is that the model is more prone to getting stuck in local minima due to the huge number of parameters to learn. Dropout training [14] is a technique for preventing overfitting in a huge parameter space. Its key idea is to decouple hidden units that activate the same output together, by randomly dropping some hidden units' activations. Essentially, this corresponds to training an ensemble of networks with smaller hidden layers, and combining their predictions. However, the individual predictions of all possible hidden layers need not be computed and combined explicitly, but the output of the ensemble can be approximately reconstructed from the full network. Thus, dropout training has a similar regularization effect as ensemble techniques.

4 Experimental Setup

We have shown the reason why the structure of NNs needs to be reconsidered in the previous Sections. In this Section, we describe evaluation measures to show how effectively NNs perform by combining recent development in learning neural networks based on the fact that the *univariate* loss is consistent with respect to rank loss on large-scale textual datasets.

Evaluation Measures. Multi-label classifiers can be evaluated in two groups of measures: bipartition and ranking. Bipartition measures operate on classification results, i.e. a set of labels assigned by classifiers to each document, while ranking measures operate on the ranked list of labels. In order to evaluate the quality of a ranked list, we consider several ranking measures [26]. Given a document $\mathbf{x}$ and associated label information $\mathbf{y}$, consider a multi-label learner $f_\theta(\mathbf{x})$ that is able to produce scores for each label. These scores, then, can be sorted in descending order. Let $r(l)$ be the rank of a label l in the sorted list of labels. We already introduced *Rank loss*, which is concerned primarily in this work, in Section 3.1. *One-Error* evaluates whether the topmost ranked label with the highest score is a positive label or not: $\mathbb{I}\left(r^{-1}(1)(f_\theta(x)) \notin \mathbf{y}\right)$ where $r^{-1}(1)$ indicates the index of a label positioning on the first place in the sorted list. *Coverage* measures on average how far one needs to go down the ranked list of labels to achieve recall of 100%: $\max_{l_i \in \mathbf{y}} r(l_i) - 1$. *Average Precision* or AP measures the average fraction of labels preceding relevant labels in the ranked list of labels: $\frac{1}{|\mathbf{y}|} \sum_{l_i \in \mathbf{y}} \frac{|\{l_j \in \mathbf{y} | r(l_j) \leq r(l_i)\}|}{r(l_i)}$.

For bipartition measures, Precision, Recall, and F_1 score are conventional methods to evaluate effectiveness of information retrieval systems. There are two ways of computing such performance measures: *Micro-averaged* measures and *Macro-averaged* measures[3][21].

$$P_{micro} = \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fp_l}, R_{micro} = \frac{\sum_{l=1}^{L} tp_l}{\sum_{l=1}^{L} tp_l + fn_l}, F_{1-micro} = \frac{\sum_{l=1}^{L} 2tp_l}{\sum_{l=1}^{L} 2tp_l + fp_l + fn_l}$$

$$P_{macro} = \frac{1}{L}\sum_{l=1}^{L} \frac{tp_l}{tp_l + fp_l}, R_{macro} = \frac{1}{L}\sum_{l=1}^{L} \frac{tp_l}{tp_l + fn_l}, F_{1-macro} = \frac{1}{L}\sum_{l=1}^{L} \frac{2tp_l}{2tp_l + fp_l + fnl}$$

Datasets. Our main interest is in large-scale text classification, for which we selected six representative domains, whose characteristics are summarized in Table 1. For Reuters21578, we used the same training/test split as previous works [30]. Training and test data were switched for RCV1-v2 [18] which originally consists of 23,149 train and 781,265 test documents. The EUR-Lex, Delicious and Bookmarks datasets were taken from the MULAN repository.[4] Except for Delicious and Bookmarks, all documents are represented with *tf-idf* features with cosine normalization such that length of the document vector is 1 in order to account for the different document lengths.

In addition to these standard benchmark datasets, we prepared a large-scale dataset from documents of the German Education Index (GEI).[5] The GEI is a database of links to more than 800,000 scientific articles with metadata, e.g. title, authorship, language of an article and index terms. We consider a subset of the dataset consisting of approximately 300,000 documents which have an abstract as well as the metadata. Each document has multiple index terms which are carefully hand-labeled by human experts with respect to the content of the articles. We processed plain text by removing stopwords and stemming each token. To avoid the computational bottleneck from a large number of labels, we chose the 1,000 most common labels out of about 50,000. We then randomly split the dataset into 90% for training and 10% for test.

[3] Note that scores computed by micro-averaged measures might be much higher than that by macro-averaged measures if there are many rarely-occurring labels for which the classification system does not perform well. This is because macro-averaging weighs each label equally, whereas micro-averaged measures are dominated by the results of frequent labels.

[4] `http://mulan.sourceforge.net/datasets.html`

[5] `http://www.dipf.de/en/portals/portals-educational-information/german-education-index`

Table 1. Number of documents (D), size of vocabulary (D), total number of labels (L) and average number of labels per instance (C) for the six datasets used in our study

Dataset	M	D	L	C
Reuters-21578	10789	18637	90	1.13
RCV1-v2	804414	47236	103	3.24
EUR-Lex	19348	5000	3993	5.31
Delicious	16105	500	983	19.02
Bookmarks	87856	2150	208	2.03
German Education Index	316061	20000	1000	7.16

Algorithms. Our main goal is to compare our NN-based approach to BP-MLL. $\mathrm{NN_A}$ stands for the single hidden layer neural networks which have *ReLUs* for its hidden layer and which are trained with SGD where each parameter of the neural networks has their own learning rate using *AdaGrad*. $\mathrm{NN_{AD}}$ additionally employs *Dropout* based on the same settings as $\mathrm{NN_A}$. T and R following BP-MLL indicate *tanh* and *ReLU* as a transfer function in the hidden layer. For both NN and BP-MLL, we used 1000 units in the hidden layer over all datasets. [6] As Dropout works well as a regularizer, no additional regularization to prevent overfitting was incorporated. The base learning rate η_0 was also determined among $[0.001, 0.01, 0.1]$ using validation data.

We also compared the NN-based algorithms to binary relevance (BR) using SVMs (Liblinear) as a base learner, as a representative of the state-of-the-art. The penalty parameter C was optimized in the range of $[10^{-3}, 10^{-2}, \ldots, 10^{2}, 10^{3}]$ based on either average of micro- and macro-average $\mathrm{F_1}$ or rankloss on validation set. $\mathrm{BR_B}$ refers to linear SVMs where C is optimized with bipartition measures on the validation dataset. BR models whose penalty parameter is optimized on ranking measures are indicated as $\mathrm{BR_R}$. In addition, we apply the same thresholding technique which we utilize in our NN approach (Section 3.3) on a ranked list produced by BR models ($\mathrm{BR_R}$). Given a document, the distance of each predicted label to the hyperplane is used to determine the position of the label in the ranked list.

5 Results

We evaluate our proposed models and other baseline systems on datasets with varying statistics and characteristics. We first show experimental results that confirm that the techniques discussed in Section 3.5 actually contribute to an increased performance of NN-based multi-label classification, and then compare all algorithms on the six above-mentioned datasets in order to get an overall impression of their performance.

Better Local Minima and Acceleration of Convergence Speed. First we intend to show the effect of ReLUs and AdaGrad in terms of convergence speed and rank loss. The left part of Figure 3 shows that all three results of AdaGrad (red lines) show a lower

[6] The optimal number of hidden units of BP-MLL and NN was tested among 20, 50, 100, 500, 1000 and 4000 on validation datasets. Usually, the more units are in the hidden layer, the better performance of networks is. We chose it in terms of computational efficiency.

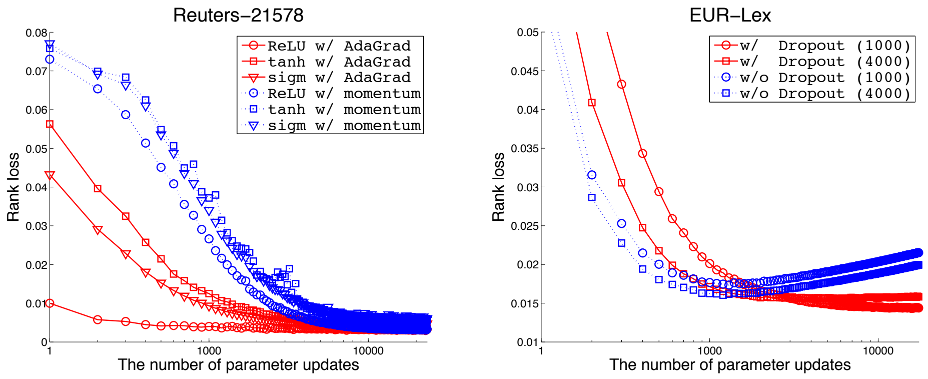


Fig. 3. (*left*) effects of AdaGrad and momentum on three types of transfer functions in the hidden layers in terms of rank loss on Reuters-21578. The number of parameter updates in x-axis corresponds to the number of evaluations of Eq. (4). (*right*) effects of dropout with two different numbers of hidden units in terms of rank loss on EUR-Lex.

rank loss than all three versions of momentum. Moreover, within each group, ReLUs outperform the versions using $\tanh$ or sigmoid activation functions. That NNs with ReLUs at the hidden layer converge faster into a better weight space has been previously observed for the speech domain [31].[7] This faster convergence is a major advantage of combining recently proposed learning components such as ReLUs and AdaGrad, which facilitates a quicker learning of the parameters of NNs. This is particularly important for the large-scale text classification problems that are the main focus of this work.

Decorrelating Hidden Units While Output Units Remain Correlated. One major goal of multi-label learners is to minimize rank loss by leveraging inherent correlations in a label space. However, we conjecture that these correlations also may cause overfitting because if groups of hidden units specialize in predicting particular label subsets that occur frequently in the training data, it will become harder to predict novel label combinations that only occur in the test set. Dropout effectively fights this by randomly dropping individual hidden units, so that it becomes harder for groups of hidden units to specialize in the prediction of particular output combinations, i.e., they decorrelate the hidden units, whereas the correlation of output units still remains. Particularly, a subset of output activations $\mathbf{o}$ and hidden activations $\mathbf{h}$ would be correlated through $\mathbf{W}^{(2)}$.

We observed overfitting across all datasets except for Reuters-21578 and RCV1-v2 under our experimental settings. The right part of Figure 3 shows how well Dropout prevents NNs from overfitting on the test data of EUR-Lex. In particular, we can see that with increasing numbers of parameter updates, the performance of regular NNs eventually got worse in terms of rank loss. On the other hand, when dropout is employed, convergence is initially slower, but eventually effectively prevents overfitting.

[7] However, unlike the results of [31], in our preliminary experiments adding more hidden layers did not further improve generalization performance.

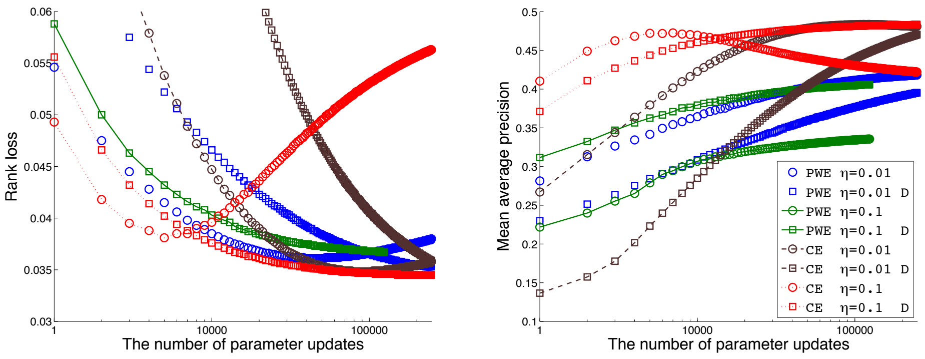


Fig. 4. Rankloss (left) and mean average precision (right) on the German Education Index test data for the different cost functions. η denotes the base learning rate and D indicates that Dropout is applied. Note that *x*-axis is in log scale.

Limiting Small Learning Rates in BP-MLL. The learning rate strongly influences convergence and learning speed [17]. As we have already seen in the Figure 2, the slope of PWE is less steep than CE, which implies that smaller learning rates should be used. Specifically, we observed PWE allows only smaller learning rate 0.01 (blue markers) in contrast with CE that works well a relatively larger learning rate 0.1 (red markers) in Figure 4. In the case of PWE with the larger learning rate (green markers), interestingly, dropout (rectangle markers in green) makes it converge towards much better local minima, yet it is still worse than the other configurations. It seems that the weights of BP-MLL oscillates in the vicinity of local minima and, indeed, converges *worse* local minima. However, it makes learning procedure of BP-MLL slow compared to NNs with CE making bigger steps for parameter updates.

With respect to Dropout, Figure 4 also shows that for the same learning rates, networks without Dropout converge much faster than ones working with Dropout in terms of both rank loss and MAP. Regardless of the cost functions, overfitting arises over the networks without Dropout and it is likely that overfitting is avoided effectively as discussed earlier.[8]

Comparison of Algorithms. Table 3 shows detailed results of all experiments with all algorithms on all six datasets, except that we could not obtain results of BP-MLL on EUR-Lex within a reasonable time frame. In an attempt to summarize the results, Table 2 shows the average rank of each algorithm in these six datasets according to all ranking an bipartition measures discussed in Section 4 [9].

We can see that although BP-MLL focuses on minimizing pairwise ranking errors, thereby capturing label dependencies, the single hidden layer NNs with cross-entropy

[8] A trajectory for PWE $\eta = 0.1$ is missing in the figure because it got 0.2 on the rankloss measure which is much worse than the other configurations.

[9] The Friedman test is passed for $\alpha = 1\%$ except for micro and macro recall ($\alpha = 10\%$) [5]. Nemenyi's critical distance between the average ranks, for which a statistical difference can be detected, is 4.7 for $\alpha = 5\%$ (4.3 for $\alpha = 10\%$).

Table 2. Average ranks of the algorithms on ranking and bipartition measures

Eval. measures	Ranking				Bipartition					
	rankloss	oneError	Coverage	MAP	miP	miR	miF	maP	maR	maF
Average Ranks										
$\mathrm{NN_A}$	2.2	2.4	2.6	2.2	**2.0**	6.0	2.4	**1.8**	5.6	**2.0**
$\mathrm{NN_{AD}}$	**1.2**	**1.4**	**1.2**	**1.6**	**2.0**	5.8	**1.8**	2.0	5.6	2.2
$\mathrm{BP\text{-}MLL_{TA}}$	5.2	7.2	6.0	6.4	7.0	3.2	7.0	6.2	**2.0**	5.6
$\mathrm{BP\text{-}MLL_{TAD}}$	4.1	6.0	4.4	5.9	7.4	**2.8**	7.4	7.2	3.2	7.0
$\mathrm{BP\text{-}MLL_{RA}}$	5.9	6.7	5.6	6.4	5.2	3.2	4.6	5.6	3.8	4.8
$\mathrm{BP\text{-}MLL_{RAD}}$	4.0	6.0	3.6	5.6	5.6	3.6	5.4	5.4	4.4	5.8
$\mathrm{BR_B}$	7.4	3.3	6.9	4.3	3.2	6.8	4.6	4.4	6.8	5.6
$\mathrm{BR_R}$	6.0	3.0	5.7	3.6	3.6	4.6	2.8	3.4	4.6	3.0

minimization (i.e., $\mathrm{NN_A}$ and $\mathrm{NN_{AD}}$) work much better not only on rank loss but also on other ranking measures. The binary relevance (BR) approaches show acceptable performance on ranking measures even though label dependency was ignored during the training phase. In addition, $\mathrm{NN_A}$ and $\mathrm{NN_{AD}}$ perform as good as or better than other methods on bipartition measures as well as on ranking measures.

We did not observe significant improvements by replacing hidden units of BP-MLL from tanh to ReLU. However, if we change the cost function in the previous setup from PWE to CE, significant improvements were obtained. Because $\mathrm{BP\text{-}MLL_{RAD}}$ is the same architecture as $\mathrm{NN_{AD}}$ except for its cost function,[10] we can say that the differences in the effectiveness of NNs and BP-MLL are due to the use of different cost functions. This also implies that the main source of improvements for NNs against BP-MLL is replacement of the cost function. Again, Figure 4 shows the difference between two cost functions more explicitly.

6 Conclusion

This paper presents a multi-label classification framework based on a neural network and a simple threshold label predictor. We found that our approach outperforms BP-MLL, both in predictive performance as well as in computational complexity and convergence speed. We have explored why BP-MLL as a multi-label text classifier does not perform well, and provided an empirical confirmation of a recent theoretical result that univariate losses might be more useful than bivariate losses for optimizing rank performance. Our experimental results showed the proposed framework is an effective method for the multi-label text classification task. Also, we have conducted extensive analysis to characterize the effectiveness of combining ReLUs with AdaGrad for fast convergence rate, and utilizing Dropout to prevent overfitting which results in better generalization.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments. This work has been supported by the German Institute for Educational Research (DIPF) under the Knowledge Discovery in Scientific Literature

[10] For PWE we use *tanh* in the output layer, but *sigmoid* is used for CE because predictions $\mathbf{o}$ for computing CE with targets $\mathbf{y}$ needs to be between 0 and 1.

Table 3. Results on ranking and bipartition measures. Results for BP-MLL on EUR-Lex are missing because the runs could not be completed in a reasonably short time.

Eval. measures	Ranking				Bipartition					
	rankloss	oneError	Coverage	MAP	miP	miR	miF	maP	maR	maF
Reuters-21578										
NN$_{\mathrm{A}}$	0.0037	0.0706	0.7473	0.9484	0.8986	0.8357	0.8660	**0.6439**	0.4424	0.4996
NN$_{\mathrm{AD}}$	**0.0031**	0.0689	**0.6611**	0.9499	0.9042	0.8344	0.8679	0.6150	0.4420	0.4956
BP-MLL$_{\mathrm{TA}}$	0.0039	0.0868	0.8238	0.9400	0.7876	0.8616	0.8230	0.5609	**0.4761**	0.4939
BP-MLL$_{\mathrm{TAD}}$	0.0039	0.0808	0.8119	0.9434	0.7945	0.8654	0.8284	0.5459	0.4685	0.4831
BP-MLL$_{\mathrm{RA}}$	0.0054	0.0808	1.0987	0.9431	0.8205	0.8582	0.8389	0.5303	0.4364	0.4624
BP-MLL$_{\mathrm{RAD}}$	0.0063	0.0719	1.2037	0.9476	0.8421	0.8416	0.8418	0.5510	0.4292	0.4629
BR$_{\mathrm{B}}$	0.0040	**0.0613**	0.8092	**0.9550**	**0.9300**	0.8096	0.8656	0.6050	0.3806	0.4455
BR$_{\mathrm{R}}$	0.0040	**0.0613**	0.8092	**0.9550**	0.8982	**0.8603**	**0.8789**	0.6396	0.4744	**0.5213**
RCV1-v2										
NN$_{\mathrm{A}}$	0.0040	0.0218	3.1564	0.9491	0.9017	0.7836	0.8385	0.7671	0.5760	0.6457
NN$_{\mathrm{AD}}$	**0.0038**	**0.0212**	**3.1108**	**0.9500**	**0.9075**	0.7813	0.8397	**0.7842**	0.5626	0.6404
BP-MLL$_{\mathrm{TA}}$	0.0058	0.0349	3.7570	0.9373	0.6685	0.7695	0.7154	0.4385	0.5803	0.4855
BP-MLL$_{\mathrm{TAD}}$	0.0057	0.0332	3.6917	0.9375	0.6347	0.7497	0.6874	0.3961	0.5676	0.4483
BP-MLL$_{\mathrm{RA}}$	0.0058	0.0393	3.6730	0.9330	0.7712	0.8074	0.7889	0.5741	0.6007	0.5823
BP-MLL$_{\mathrm{RAD}}$	0.0056	0.0378	3.6032	0.9345	0.7612	0.8016	0.7809	0.5755	0.5748	0.5694
BR$_{\mathrm{B}}$	0.0061	0.0301	3.8073	0.9375	0.8857	0.8232	**0.8533**	0.7654	0.6342	0.6842
BR$_{\mathrm{R}}$	0.0051	0.0287	3.4998	0.9420	0.8156	**0.8822**	0.8476	0.6961	**0.7112**	**0.6923**
EUR-Lex										
NN$_{\mathrm{A}}$	0.0195	0.2016	310.6202	0.5975	0.6346	0.4722	0.5415	0.3847	0.3115	0.3256
NN$_{\mathrm{AD}}$	**0.0164**	**0.1681**	**269.4534**	**0.6433**	**0.7124**	0.4823	**0.5752**	**0.4470**	0.3427	0.3687
BR$_{\mathrm{B}}$	0.0642	0.1918	976.2550	0.6114	0.6124	0.4945	0.5471	0.4260	**0.3643**	**0.3752**
BR$_{\mathrm{R}}$	0.0204	0.2088	334.6172	0.5922	0.0329	**0.5134**	0.0619	0.2323	0.3063	0.2331
German Education Index										
NN$_{\mathrm{A}}$	**0.0350**	0.2968	138.5423	**0.4828**	0.4499	0.4200	**0.4345**	0.4110	0.3132	**0.3427**
NN$_{\mathrm{AD}}$	0.0352	**0.2963**	138.3590	0.4797	0.4155	0.4472	0.4308	0.3822	0.3216	0.3305
BP-MLL$_{\mathrm{TA}}$	0.0386	0.8309	150.8065	0.3432	0.1502	**0.6758**	0.2458	0.1507	**0.5562**	0.2229
BP-MLL$_{\mathrm{TAD}}$	0.0371	0.7591	139.1062	0.3281	0.1192	0.5056	0.1930	0.1079	0.4276	0.1632
BP-MLL$_{\mathrm{RA}}$	0.0369	0.4221	143.4541	0.4133	0.2618	0.4909	0.3415	0.3032	0.3425	0.2878
BP-MLL$_{\mathrm{RAD}}$	0.0353	0.4522	**135.1398**	0.3953	0.2400	0.5026	0.3248	0.2793	0.3520	0.2767
BR$_{\mathrm{B}}$	0.0572	0.3052	221.0968	0.4533	**0.5141**	0.2318	0.3195	0.3913	0.1716	0.2319
BR$_{\mathrm{R}}$	0.0434	0.3021	176.6349	0.4755	0.4421	0.3997	0.4199	**0.4361**	0.2706	0.3097
Delicious										
NN$_{\mathrm{A}}$	0.0860	0.3149	396.4659	0.4015	**0.3637**	0.4099	0.3854	0.2488	0.1721	0.1772
NN$_{\mathrm{AD}}$	**0.0836**	**0.3127**	**389.9422**	**0.4075**	0.3617	0.4399	**0.3970**	**0.2821**	0.1777	**0.1824**
BP-MLL$_{\mathrm{TA}}$	0.0953	0.4967	434.8601	0.3288	0.1829	0.5857	0.2787	0.1220	0.2728	0.1572
BP-MLL$_{\mathrm{TAD}}$	0.0898	0.4358	418.3618	0.3359	0.1874	0.5884	0.2806	0.1315	0.2427	0.1518
BP-MLL$_{\mathrm{RA}}$	0.0964	0.6157	427.0468	0.2793	0.2070	**0.5894**	0.3064	0.1479	**0.2609**	0.1699
BP-MLL$_{\mathrm{RAD}}$	0.0894	0.6060	411.5633	0.2854	0.2113	0.5495	0.3052	0.1650	0.2245	0.1567
BR$_{\mathrm{B}}$	0.1184	0.4355	496.7444	0.3371	0.1752	0.2692	0.2123	0.0749	0.1336	0.0901
BR$_{\mathrm{R}}$	0.1184	0.4358	496.8180	0.3371	0.2559	0.3561	0.2978	0.1000	0.1485	0.1152
Bookmarks										
NN$_{\mathrm{A}}$	0.0663	0.4924	22.1183	0.5323	0.3919	0.3907	0.3913	0.3564	0.3069	0.3149
NN$_{\mathrm{AD}}$	**0.0629**	**0.4828**	**20.9938**	**0.5423**	**0.3929**	0.3996	**0.3962**	**0.3664**	0.3149	**0.3222**
BP-MLL$_{\mathrm{TA}}$	0.0684	0.5598	23.0362	0.4922	0.0943	0.5682	0.1617	0.1115	0.4743	0.1677
BP-MLL$_{\mathrm{TAD}}$	0.0647	0.5574	21.7949	0.4911	0.0775	**0.6096**	0.1375	0.0874	**0.5144**	0.1414
BP-MLL$_{\mathrm{RA}}$	0.0707	0.5428	23.6088	0.5049	0.1153	0.5389	0.1899	0.1235	0.4373	0.1808
BP-MLL$_{\mathrm{RAD}}$	0.0638	0.5322	21.5108	0.5131	0.0938	0.5779	0.1615	0.1061	0.4785	0.1631
BR$_{\mathrm{B}}$	0.0913	0.5318	29.6537	0.4868	0.2821	0.2546	0.2676	0.1950	0.1880	0.1877
BR$_{\mathrm{R}}$	0.0895	0.5305	28.7233	0.4889	0.2525	0.4049	0.3110	0.2259	0.3126	0.2569

(KDSL) program, in part by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, and in part by the German Research Foundation under grant 798/1-5.

References

[1] Bi, W., Kwok, J.T.: Multi-label classification on tree-and dag-structured hierarchies. In: Proceedings of the 28th International Conference on Machine Learning, pp. 17–24 (2011)

[2] Calauzènes, C., Usunier, N., Gallinari, P.: On the (non-)existence of convex, calibrated surrogate losses for ranking. In: Advances in Neural Information Processing Systems 25, pp. 197–205 (2012)

[3] Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: Proceedings of the 27th International Conference on Machine Learning, pp. 279–286 (2010)

[4] Dembczyński, K., Kotłowski, W., Hüllermeier, E.: Consistent multilabel ranking through univariate losses. In: Proceedings of the 29th International Conference on Machine Learning, pp. 1319–1326 (2012)

[5] Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research 7, 1–30 (2006)

[6] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12, 2121–2159 (2011)

[7] Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in Neural Information Processing Systems 14, pp. 681–687 (2001)

[8] Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874 (2008)

[9] Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning 73(2), 133–153 (2008)

[10] Gao, W., Zhou, Z.H.: On the consistency of multi-label learning. Artificial Intelligence 199-200, 22–44 (2013)

[11] Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 195–200 (2005)

[12] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, JMLR W&CP, pp. 249–256 (2010)

[13] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, JMLR W&CP, pp. 315–323 (2011)

[14] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. ArXiv preprint ArXiv:1207.0580 (2012)

[15] Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998)

[16] Joachims, T.: Training linear svms in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217–226 (2006)

[17] LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural Networks: Tricks of the Trade, pp. 9–48 (2012)

[18] 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)

[19] Liu, T.Y., Yang, Y., Wan, H., Zeng, H.J., Chen, Z., Ma, W.Y.: Support vector machines classification with a very large-scale taxonomy. SIGKDD Explorations 7(1), 36–43 (2005)

[20] Loza Mencía, E., Park, S.H., Fürnkranz, J.: Efficient voting prediction for pairwise multilabel classification. Neurocomputing 73(7-9), 1164–1176 (2010)

[21] Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)

[22] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning, pp. 807–814 (2010)

[23] Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning 85(3), 333–359 (2011)

[24] Rubin, T.N., Chambers, A., Smyth, P., Steyvers, M.: Statistical topic models for multi-label document classification. Machine Learning 88(1-2), 157–208 (2012)

[25] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)

[26] Schapire, R.E., Singer, Y.: BoosTexter: A boosting-based system for text categorization. Machine Learning 39(2/3), 135–168 (2000)

[27] Solla, S.A., Levin, E., Fleisher, M.: Accelerated learning in layered neural networks. Complex Systems 2(6), 625–640 (1988)

[28] Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multi-label classification of music into emotions. In: Proceedings of the 9th International Conference on Music Information Retrieval, pp. 325–330 (2008)

[29] Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Random k-labelsets for multilabel classification. IEEE Transactions on Knowledge and Data Engineering 23(7), 1079–1089 (2011)

[30] Yang, Y., Gopal, S.: Multilabel classification with meta-level features in a learning-to-rank framework. Machine Learning 88(1-2), 47–68 (2012)

[31] Zeiler, M.D., Ranzato, M., Monga, R., Mao, M.Z., Yang, K., Le, Q.V., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., Hinton, G.E.: On rectified linear units for speech processing. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3517–3521 (2013)

[32] Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering 18, 1338–1351 (2006)