**Noname manuscript No.**
(will be inserted by the editor)

# Classifier Chains for Multi-label Classification

**Jesse Read · Bernhard Pfahringer · Geoff Holmes · Eibe Frank**

Received: date / Accepted: date

**Abstract** The widely known binary relevance method for multi-label classification, which considers each label as an independent binary problem, has often been overlooked in the literature due to the perceived inadequacy of not directly modelling label correlations. Most current methods invest considerable complexity to model interdependencies between labels. This paper shows that binary relevance-based methods have much to offer, and that high predictive performance can be obtained without impeding scalability to large datasets. We exemplify this with a novel classifier chains method that can model label correlations while maintaining acceptable computational complexity. We extend this approach further in an ensemble framework. An extensive empirical evaluation covers a broad range of multi-label datasets with a variety of evaluation metrics. The results illustrate the competitiveness of the chaining method against related and state-of-the-art methods, both in terms of predictive performance and time complexity.

**Keywords** multi-label classification · problem transformation · ensemble methods · scalable methods

Jesse Read*, Bernhard Pfahringer, Geoff Holmes, Eibe Frank
Department of Computer Science
The University of Waikato
Hamilton
New Zealand
Phone: +64 7 838 4466 ext. 8766, Fax: +64 7 838 5095
E-mail: {jmr30,bernhard,geoff,eibe}@cs.waikato.ac.nz
* *Present address of Jesse Read:*
Department of Signal Theory and Communications
Universidad Carlos III
Madrid
Spain
Phone: +34 91 624 6005, Fax: +34 91 624 8749
E-mail: jesse@tsc.uc3m.es

## 1 Introduction

Multi-label classification is the supervised learning problem where an instance may be associated with multiple labels. This is opposed to the traditional task of single-label classification (i.e. multi-class, or binary) where each instance is only associated with a single class label. The multi-label context is receiving increased attention and is applicable to a wide variety of domains, including text classification [29,14,13], scene and video classification [1,8], and bioinformatics [9,35].

A common approach to multi-label classification is to perform *problem transformation*, whereby a multi-label problem is transformed into one or more single-label (i.e. binary, or multi-class) problems. In this way, single-label classifiers are employed; and their single-label predictions are transformed into multi-label predictions. Problem transformation is attractive on account of both scalability and flexibility: any off-the-shelf single-label classifier can be used to suit requirements. Previous work has used Support Vector Machines [14], Naive Bayes [20], $k$ Nearest Neighbor methods [30], and Perceptrons [13].

The alternative to problem transformation is to modify an algorithm directly to make multi-label predictions. Well-known approaches include AdaBoost [29], decision trees [35], and lazy methods [39]. Such methods are usually chosen to work specifically in certain domains. Decision trees are especially popular in bioinformatics, for example. Some adaptations involve problem transformations internally which may be generalisable.

There are several families of problem transformation methods in the multi-label literature. These methods arise from one or more fundamental problem transformation approaches that either form the core of more complex methods or are used in algorithm adaptations. Here we review these fundamental methods.

The most common problem transformation method is the *binary relevance* method (`BR`) [33,14,38]. `BR` transforms a multi-label problem into multiple binary problems; one problem for each label, such that each binary model is trained to predict the relevance of one of the labels.

Although `BR` is mentioned throughout the literature, it has often been overlooked on the grounds that it does not directly model correlations which exist between labels in the training data. The argument is that, due to this loss of information, `BR`'s predictive performance suffers.

A binary *pairwise* classification approach (`PW`) can also be applied to multi-label classification as a problem transformation method, where a binary model is trained for each *pair* of labels. The predictions of these models result more naturally in a set of pairwise preferences than a multi-label prediction (thus becoming popular in ranking schemes), but this method has been adapted in [13] to make multi-label predictions. Although `PW` performs well in several domains, it faces quadratic complexity in terms of the number of labels and, for this reason, is usually intractable for large problems.

Another well-known problem transformation method is the *label combination* or *label power-set* method (`LC`), which has formed the basis for methods in several publications, for example [1,34,25]. `LC` transforms a multi-label problem into a single-label (multi-class) problem by treating all label sets as atomic labels, i.e. each label set is treated as a single label in a single-label multi-class problem. Although able to model label correlations in the training data, cited disadvantages of `LC` include its worst-case computational complexity (exponential with the number of labels) and tendency to over-fit the training data because it can only model label sets observed in the training data [34,25].

The consensus view in the literature is that it is crucial to take into account label correlations during the classification process [17,14,34,25,31,36,19], and most work has turned away from `BR` to more complex methods. However, as the size of multi-label datasets grows, many methods are challenged by the growth in the number of possible correlations; even methods that only try to model a subset of possible label correlations can become intractable. Consequently, although these methods can be very accurate on small datasets, they are very slow or even intractable on larger datasets, like [13] and [3]. This necessarily restricts their usefulness, as many multi-label contexts involve large numbers of examples and labels.

In this paper we extend our work on classifier chains for multi-label classification, which we introduced in [27]. Our classifier chains method (`CC`), which is based on the `BR` method, overcomes the disadvantages of `BR` and achieves higher predictive performance, but still retains important advantages of `BR`, most importantly low time complexity. `CC` offers a general problem transformation method that inherits the efficiency of `BR` and competes with the high accuracy of more computationally complex methods. In [27] we applied classifier chains in an ensemble framework (`ECC`) and empirically demonstrated high predictive performance as compared with other modern multi-label methods.

Further to the presentation in [27], this work investigates, develops, and evaluates the classifier chains framework along various lines. Specifically, it provides:

– a deeper discussion of the performance of the `CC` framework, where we look at the probabilistic interpretation of classifier chains presented in [3]
– a new, more competitive ensemble scheme
– a new, more scalable ensemble variation for larger datasets
– a thorough discussion of the time complexity of different multi-label methods
– a more extensive empirical evaluation, extended across several dimensions: new datasets, additional measures of multi-label evaluation, a discussion on threshold selection, and comparison with additional methods from the multi-label literature
– a more in-depth discussion of results, showing how the above provides additional contributions to multi-label classification.

## 2 Preliminaries

In this section, we present the formal notation that we use throughout.

In all that follows, $X^d \subset \mathbb{R}$ is the input domain of all possible attribute values. An instance is represented as a vector of $d$ attribute values $\mathbf{x} = [x_1, \ldots, x_d]$. The set $\mathcal{L} = \{1, \ldots, L\}$ is the output domain of possible labels. Each instance $\mathbf{x}$ is associated with a subset of these labels. This set is represented by an $L$-vector $\mathbf{y} = [y_1, \ldots, y_L]$, where $y_1 = 1$ if and only if label $j$ is associated with instance $\mathbf{x}$, and 0 otherwise.

We assume a set of training data $D$ of $N$ labelled examples $D = \{(\mathbf{x}^i, \mathbf{y}^i) | i = 1, \ldots, N\}$. We use superscripts here to avoid ambiguity with the label dimension, such that $y_j^i$ represents the binary relevance of the $j$th label pertaining to the $i$th example.

Hence, in the `BR`-context, a classifier $\mathbf{h}$ is comprised of $L$ binary classifiers $h_1, \ldots, h_L$, where each $h_j$ learns from $D$ to predict the relevance of $\hat{y}_j \in \{0, 1\}$, i.e. whether an example $\mathbf{x}$ belongs to the $j$th label (1) or not (0). Thus, the output of $\mathbf{h}$, i.e. the prediction, is a vector $\hat{\mathbf{y}} \in \{0, 1\}^L$ for any instance $\mathbf{x}$.

Table 1: Worst case computational complexity for common methods

| method | #models | class labels / model | examples / model |
|---|---|---|---|
| BR | $L$ | 2 | $N$ |
| PW | $\frac{L(L-1)}{2}$ | 2 | $\leq N$ |
| LC | 1 | $\min(N, 2^L - 1)$ | $N$ |

## 3 In Defence of the Binary Relevance Method

Although BR's disadvantages are widely acknowledged, its advantages have rarely been mentioned. BR is theoretically simple and intuitive. It is highly resistant to overfitting label combinations, since it does not expect examples to be associated with previously-observed combinations of labels (as LC methods do). It can therefore handle very irregular labelling. Since labels have a one-to-one relationship with binary models, labels can be added and removed without affecting the rest of the model (making it applicable to a variety of contexts such as evolving data and other dynamic scenarios).

However the most important advantage of BR is arguably its low computational complexity as compared with other methods. Let us consider the worst case computational complexity of the fundamental problem transformation methods. In Table 1, complexity is displayed in terms of the number of single-label models involved in the transformation, and the class labels and examples associated with these models.

Given a constant number of examples, BR scales linearly with the number of labels: there are $L$ binary problems. A PW approach on the other hand scales quadratically with $L$ in terms of the number of models. Although in the pairwise single-label case models may be much smaller than a BR-like one-vs rest scheme (as explained by [12]), this does not necessarily apply to the same extent in the multi-label domain where multiple labels can be associated with each example. Thus, we find that PW methods are very sensitive to large $L$. LC involves only a single model, but the worst case number of class labels scales exponentially with $L$ (limited by $N$). All methods that model all label sets will necessarily suffer this complexity.

We note that in typical multi-label settings, the label set is limited in scope, such that the number of labels can be expected to be fewer than the number of attributes in the input space (we discuss this further in Section 8.1). This is the setting we consider. Scenarios where the number of labels is very large, or not defined prior to classification, are beyond the scope of this paper. However, note that the number of examples $N$ may be large, and this is the limiting factor for LC: depending on the dataset, an LC transformation may result in thousands or even tens of thousands of class labels. This is particularly problematic where base classifiers imply greater than linear complexity with respect to the number of class labels, and the complexity of multi-class models like LC can be even greater.

Because BR's $L$ binary problems are separate, under demanding circumstances it is conceivable (and desirable) to run each label problem separately, in either parallel or serial fashion (over $L$ processors, or $L$ iterations), thus requiring only a single binary problem in memory at any point in time.

Nevertheless, despite its advantages, it is clear that not modelling label correlations can cause problems for the BR method in terms of predictive performance. In the next section we present a novel binary relevance method, CC, which can model label correlations, while maintaining acceptable computational complexity similar to BR.

Fig. 1: Transformation under BR and CC for $(\mathbf{x}, \mathbf{y})$ where $\mathbf{y} = [1, 0, 0, 1, 0]$ and $\mathbf{x} = [0, 1, 0, 1, 0, 0, 1, 1, 0]$ (assuming, for simplicity, a binary attribute space). Each classifier $h_j$ is trained to predict $\hat{y}_j \in \{0, 1\}$.

(a) BR's transformation

| $\mathbf{h}$ : | $\mathbf{x} \rightarrow$ | $\mathbf{y}$ |
|---|---|---|
| $h_1$: | [0,1,0,1,0,0,1,1,0] | 1 |
| $h_2$: | [0,1,0,1,0,0,1,1,0] | 0 |
| $h_3$: | [0,1,0,1,0,0,1,1,0] | 0 |
| $h_4$: | [0,1,0,1,0,0,1,1,0] | 1 |
| $h_5$: | [0,1,0,1,0,0,1,1,0] | 0 |

(b) CC's transformation

| $\mathbf{h}$ : | $\mathbf{x}' \rightarrow$ | $\mathbf{y}$ |
|---|---|---|
| $h_1$: | [0,1,0,1,0,0,1,1,0] | 1 |
| $h_2$: | [0,1,0,1,0,0,1,1,0,1] | 0 |
| $h_3$: | [0,1,0,1,0,0,1,1,0,1,0] | 0 |
| $h_4$: | [0,1,0,1,0,0,1,1,0,1,0,0] | 1 |
| $h_5$: | [0,1,0,1,0,0,1,1,0,1,0,0,1] | 0 |

## 4 The Classifier Chains Model (CC)

The Classifier Chains model (CC) involves $L$ binary transformations—one for each label—as in BR. In this sense, CC is also a binary relevance method, but it is different from BR in that the attribute space for each binary model is extended with the 0/1 label relevances of all previous classifiers; thus forming a *classifier chain*. The training procedure is outlined in Algorithm 1. Figure 1 illustrates the process with an example, contrasting it with BR.

**Algorithm 1** CC's training phase for training set $D$ and label set $\mathcal{L}$ of $L$ labels.

```
TRAINING($D = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$)
1  for $j = 1, \ldots, L$
2       do ▷ the $j$th binary transformation and training
3          $D'_j \leftarrow \{\}$
4          for $(\mathbf{x}, \mathbf{y}) \in D$
5               do $\mathbf{x}' \leftarrow [x_1, \ldots, x_d, y_1, \ldots, y_{j-1}]$
6                  $D'_j \leftarrow D'_j \cup (\mathbf{x}', y_j)$
7          ▷ train $h_j$ to predict binary relevance of $y_j$
8          $h_j : D'_j \rightarrow \{0, 1\}$
```

Hence a chain $\mathbf{h} = (h_1, \ldots, h_L)$ of binary classifiers is formed. Each classifier $h_j$ in the chain is responsible for learning and predicting the binary association of the $j$th label given the attribute space, augmented by all prior binary relevance predictions in the chain. Note that, although Figure 1 shows binary attributes in the original $\mathbf{x}$, it may also be composed of other types of attributes such as numerical values. The chained attributes, however, are always binary.

It is straightforward to obtain classifications from this chain. The classification process begins at $h_1$ and propagates along the chain: the $j$th binary classifier predicts the relevance of the $j$th label, given the attribute space augmented by the predictions of all previous binary classifiers in the chain. This classification procedure is outlined in Algorithm 2.

This chaining method passes label information between classifiers, allowing CC to take into account correlations in the label space, and thus overcoming the problem associated with BR of ignoring this information. Although the additional attributes

**Algorithm 2** CC's prediction phase for a test instance $\mathbf{x}$.

```
CLASSIFY($\mathbf{x}$)
1  ▷ global $\mathbf{h} = (h_1, \ldots, h_L)$
2  $\mathbf{y} \leftarrow [\hat{y}_1, \ldots, \hat{y}_L]$
3  for $j = 1, \ldots, L$
4        do $\mathbf{x}' \leftarrow [x_1, \ldots, x_d, \hat{y}_1, \ldots, \hat{y}_{j-1}]$
5           $\hat{y}_j \leftarrow h_j(\mathbf{x}')$
6  return $\hat{\mathbf{y}}$
```

make up a small part of the total attribute space, if strong correlations exist then these attributes give any base classifier relatively more predictive power.

The advantage of using the 0/1 label relevance predictions instead of probabilistic predictions, which some classifiers may be able to provide, is that no internal validation procedure is necessary: at training time each classifier can train directly on the 0/1 relevance values available in the training data. This is also why we use a chain model, since combining all 0/1 relevance values at once using a stacking approach would require a meta process with internal validation (we refer to such a method in Section 6). In later sections we compare empirically with such a meta classifier and show that it always requires more than twice the running time of our chain model.

Although an average of $L/2$ attributes are added to each instance, the size of the label space $L$ is limited in practice and therefore the computational complexity of CC can be very close to that of BR. BR's complexity is $O(L \times f(d, N))$, where $f(d, N)$ is the complexity of the underlying learner for $d$ attributes and $N$ examples. CC's complexity is $O(L \times f(d+L, N))$, i.e. a penalty is incurred for having up to $L$ additional attributes. Assuming a linear base learner (with respect to the number of attributes $d$), CC's complexity becomes $O(L \times d \times f(1, N) + L \times L \times f(1, N))$, where the first term dominates as long as $L < d$, which we normally expect (see Section 8), and therefore the effective complexity of CC is $O(L \times d \times f(1, N))$, which is identical to BR's complexity. Indeed, we will see empirically in Section 8 that the difference in running time between BR and CC tends to be small in practice.

Like those of BR, CC's models can both be parallelized (at least in the training procedure) and serialized, such that only a single binary problem is required in memory at any point in time—a clear advantage over methods based on a single large model.

The order of the chain itself (determined by the order of the label variables in each $\mathbf{y}$ vector) will normally have an effect on accuracy. This has also been noticed by the authors of [3], who expand the work on CC (from [27]) by formulating a probabilistic interpretation. In their analysis of the classifier chains problem, they explain how Bayes-optimal *probabilistic classifier chains* (PCC) can be formed based on probability theory. Under the product rule, the conditional probability of $\mathbf{y}$ for an instance $\mathbf{x}$ (where $P_{\mathbf{x}}(\mathbf{y}) \equiv P(\mathbf{y}|\mathbf{x})$) is:

$$P_{\mathbf{x}}(\mathbf{y}) = P_{\mathbf{x}}(y_1) \cdot \prod_{j=2}^{L} P_{\mathbf{x}}(y_j | y_1, \ldots, y_{j-1})$$

When $h_j(\cdot)$ is a probabilistic classifier, this can be rewritten as:

$$P_{\mathbf{x}}(\mathbf{y}) = h_1(\mathbf{x}) \cdot \prod_{j=2}^{L} h_j(\mathbf{x}, y_1, \ldots, y_{j-1})$$

Given $P_{\mathbf{x}}$ and a loss function to be minimised, (we will review and present several in Section 8.3), an optimum prediction can be obtained by minimising the risk of using different variable orderings in the vectors $\mathbf{y}$.

By employing different loss functions PCC can be tailored to a specific multi-label evaluation measure. In contrast, CC, as we present it, does not optimise a particular loss function and is thus not tailored towards a specific evaluation measure. When labels are independent, CC will tend to function similarly to BR. However, given the presence of label correlations, as explained in [3], CC will approximate the mode of the joint distribution, that is to say, that it will tend to function more like LC.

CC is neither theoretically optimal nor is it tailored to a specific evaluation measure. There is a reason for this: an optimal classifier like PCC comes at an intractable computational cost in practice. While PCC must look at each of $2^L$ possible combinations, CC needs only consider a single (default or random) order of the $L$ label variables in $\mathbf{y}$. For PCC, this implies an upper limit for $L$ of around 10—15 [3]. For example, this is nearly two orders of magnitude less than the number of labels in the *Delicious* dataset (where $L = 917$) which we consider in our experimental evaluation in Section 8. Furthermore, we claim that by not being tailored towards a specific evaluation measure, the methods we present provide all-round good performance—a desirable quality in practice, especially where "off-the-shelf" methods are sought after.

It is true that a randomly arranged chain can be poorly ordered. We overcome this issue with a different strategy; by employing ensembles of CC, with a random chain order for each iteration. This limits complexity to be linear with respect to the number of iterations and provides reliably high predictive performance, as we show empirically. In the next section, we present this ensemble framework.

## 5 Ensembles of Classifier Chains (ECC)

In this section we preset Ensembles of Classifier Chains (ECC). As indicated in the previous section, a single standalone CC model can be poorly ordered. Moreover there is the possible effect of error propagation along the chain at classification time, when one (or more) of the first classifiers predict poorly. Using an ensemble of chains, each with a random label order, greatly reduces the risk of these events having an overall negative effect on classification accuracy, and at only a linear time cost with respect to the number of iterations. An ensemble framework will not interfere with the ability to build CC's binary models in a parallel or serial fashion where memory space is a priority. Moreover a common advantage of ensembles is their well-known effect of generally increasing overall predictive performance.

Note that binary methods are occasionally referred to as ensemble methods because they employ multiple binary models. However, none of these models is multi-label capable and therefore we use the term *ensemble* strictly in the sense of an *ensemble of multi-label methods.*

In [27] we used an ensemble taking simple subsets of the examples in the training set, for each ensemble member, sampled without replacement. Subsequently we discovered that a *bagging* scheme [2] can achieve higher predictive performance. This comes at a marginal cost in terms of time complexity, which can be justified by the gains in predictive performance. Moreover, in Section 7 we present new variations of ECC which give much improved efficiency, and provide scalability for large datasets.

ECC trains $m$ CC classifiers $\mathbf{h}_1, \dots, \mathbf{h}_m$; each classifier is given a random chain ordering, and is trained on a random selection of $N$ training instances *sampled with replacement.*

ECC initially produces a vector of confidence outputs $\hat{\mathbf{w}} = [\hat{w}_1, \dots, \hat{w}_L] \in \mathbb{R}^L$ where $\hat{w}_j$ represents the confidence for the $j$th label. Given prediction vectors $\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_m$ from iterations $1, \dots, m$, this confidence is calculated as:

$$\hat{w}_j = \frac{1}{m} \sum_{k=1}^{m} \hat{y}_{j,k}$$

A threshold function can be applied to $\hat{\mathbf{w}}$ to create a bipartition of relevant and irrelevant labels:

$$\hat{\mathbf{y}} = f_t(\hat{\mathbf{w}})$$

for some threshold function $f$. Threshold functions are discussed in Section 8.4.

The ensemble voting scheme is demonstrated with an example in Figure 2. This ensemble scheme is generic and thus straightforward to apply to any multi-label method. We can therefore apply this scheme to BR to create an *ensemble of binary relevance* classifiers (EBR). The ensemble process is carried out identically to ECC, except that chain ordering has no effect on BR.

| | [ | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | ] |
|---|---|---|---|---|---|---|---|---|
| $\hat{\mathbf{y}}_1 =$ | [ | 0 | 0 | 1 | 1 | 0 | 0 | ] |
| $\hat{\mathbf{y}}_2 =$ | [ | 1 | 0 | 0 | 1 | 0 | 0 | ] |
| $\hat{\mathbf{y}}_3 =$ | [ | 0 | 0 | 1 | 1 | 0 | 0 | ] |
| $\hat{\mathbf{y}}_4 =$ | [ | 0 | 0 | 1 | 1 | 0 | 0 | ] |
| $\hat{\mathbf{y}}_5 =$ | [ | 1 | 0 | 1 | 0 | 0 | 0 | ] |
| $\hat{\mathbf{w}} =$ | [ | $\frac{2}{5}$ | 0 | $\frac{4}{5}$ | $\frac{4}{5}$ | 0 | 0 | ] |
| $\hat{\mathbf{w}} =$ | [ | 0.4 | 0.0 | 0.8 | 0.8 | 0.0 | 0.0 | ] |
| $\hat{\mathbf{y}} = f_{t=0.5}(\hat{\mathbf{w}}) =$ | [ | 0 | 0 | 1 | 1 | 0 | 0 | ] |

Fig. 2: An example of ECC's voting procedure, where $L = 6$, $m = 5$ and $f_t(\cdot)$ is a simple threshold function as in Section 8.4 under a threshold of $t = 0.5$.

## 6 Related Work

Before embarking on an empirical evaluation of the methods presented in this paper, let us review existing work on multi-label learning.

## 6.1 Related methods

Adding label information to the attribute space as we have done in `CC` is not a new idea. Godbole and Sarawagi [14] stacked `BR` classification outputs along with the full original attribute space, creating a two-stage classification process. We refer to this method as *Meta-BR* (`MBR`). Similarly to `CC`, `MBR` can take into account label correlations. However, using a meta classifier implies an extra iteration on both training and test data as well as internal classifications on the training data to acquire the label outputs for the meta training step. In contrast, `CC` only requires a single training iteration like `BR`, and uses labels directly from the training data without any internal classification.

Another multi-label method, described in [28], uses a meta scheme with a Hamming distance metric to map the confidence predictions of a single-label classifier to label combinations that have been observed in the training data. The label vector from the training set with the shortest Hamming distance to the predicted vector is chosen as the prediction. When applied to the binary outputs of `BR` we refer to this method as *Subset-Mapped BR* (`SMBR`). Thus, this method also involves a meta step; one which is related to `LC` in terms of mapping label combinations from the training data.

A `BR`-based boosting algorithm was introduced in [36]. Binary models are trained on subsets of the example and attribute spaces. By sharing binary models between labels, redundancy in the learning space is exploited, and thereby complexity is reduced. This is a good example of how the complexity of a binary relevance approach can be significantly reduced, and supports our message that binary methods have been underrated.

In [24], the issue of *class-label imbalance* is identified with respect to large text categorisation problems tackled with `BR`, caused by the inevitable label sparsity, which yields a relatively small number of positive examples in the transformed problems. This work addresses this issue by overweighting positive examples in each of the `BR` models. The authors also report that classification speed can be improved with marginal effect on predictive performance by ignoring rare class labels altogether and consider pruning `BR` based on performance. Results are positive with respect to their text data corpus, although applying these strategies to other domains (where class skew is not as prevalent) may be less effective.

The work in [17] presents a general framework for extracting shared subspaces in a `BR` approach, which adds a second part to the `BR` method. This addition models label correlations. However, despite using an approximation algorithm, it is computationally expensive, which is reflected in the experimental setup used in [36] where only relatively small samples of 1000 training instances are used. [31] uses a computationally complex hypergraph method to model label correlations that also, despite a proposed approximate formulation, induces high computational complexity.

## 6.2 Alternative methods

In Section 1 we mentioned pairwise (`PW`) problem transformation methods. `CLR` [13] is a well-known example (many other pairwise methods are specific to the label ranking problem, which is beyond the scope of this paper). In [21], each pairwise classifier is accompanied by two probabilistic models to isolate the overlapping attribute space. This adds further complexity, and causes a computational bottleneck on large datasets.

Several multi-label methods have been developed around the LC problem transformation paradigm; improving its performance as well as reducing its time complexity. A well-known example is the RAkEL system [34], which trains $m$ LC models using random subsets of $k$ labels. A simple ensemble-based voting process determines the final classification set over the $m$ sub-models. Using appropriate values of $m$ and $k$, RAkEL has been shown to be more accurate than BR and LC. In more recent work, we presented EPS [25], an ensemble method that uses pruning to reduce the computational complexity of LC in practice by up to an order of magnitude, and a label-set subsampling method to preserve high predictive performance. This method proved to be competitive in terms of efficiency. EPS uses a similar voting scheme to ECC for producing multi-label predictions from the ensemble, but takes simple subsets of the training set *without* replacement, rather than the bagging scheme we present here.

There are also several notable algorithm adaptation methods in the literature. The BoosTexter system [29] was an important milestone in multi-label classification and ranking. BoosTexter provides multi-label adaptations of the well-known AdaBoost boosting paradigm. Improving these algorithms, including threshold selection, was a focus of the work in [18]. AdaBoost-based methods have mainly been used in bioinformatics applications (where boosting and decision trees are particularly popular [18]). Although they can work with textual datasets, they scale poorly with the number of labels and can fail to perform well on sparse data [11]. This family of methods has been recognised as having high computational complexity [21].

The authors of [4] expanded on the work on lazy classification and introduced the IBLR algorithm: a combination of instance-based learning and logistic regression. IBLR takes label correlations into account by using the labels of neighboring examples as extra attributes in a logistic regression scheme. This secondary step is, in effect, a specialisation of the stacking procedure of the MBR paradigm.

In [5] decision trees were modified to allow for multi-label prediction at the leaves. A further example of a decision tree model is [35]. This method has been shown to work very well on hierarchical bioinformatics problems.

## 7 Scaling Up ECC

A notable potential drawback of ECC is that, as an ensemble of binary transformations, a potentially very large number of instances must be processed. In terms of the number of instances, each label induces $m \times N$ instances; one for each of $m$ iterations from the original $N$ instances. The binary context as used in ECC usually entails considerable redundancy in the learning space, as also noticed in the case of the multi-label binary approach in [36]. Some other methods have also been used in the literature for reducing the complexity of binary relevance problems by taking advantage of label sparsity using compressive sensing techniques [32,16].

In this section we investigate a simple strategy for reducing redundancy in the learning space of ECC; and thus further reducing its time and memory complexity, without significant loss of predictive performance. More specifically, we investigate taking random subsets of both the attribute and the instance space to generate each ensemble member and also consider the effect of the number of iterations.

Figures 3, 4 and 5 illustrate the reductions that can be made in the learning space, and the reward (in terms of reduced time complexity) and the cost (in terms of accuracy) incurred. To obtain these results, we ran ECC in different scenarios where each binary model takes random subsets of the attribute space, instance space, and different numbers of iterations. We use the J48 decision tree learner (the WEKA software's implementation of C4.5 [23]) as the base classifier, the *TMC2007* dataset, and the multi-label accuracy evaluation measure as reviewed in Section 8.

The results show that similar results in predictive performance can be obtained for significantly less complexity. For example, with a subset of 75% of the training instances, 50% of the attribute space, and only 10 iterations, accuracy is negligibly less than when using the full spectrum. Similar results can be obtained on other datasets. Such a configuration can be used on both EBR and ECC to make these methods more efficient on larger datasets without significant losses in predictive performance. We also show empirical evidence for this in the following, Section 8.

## 8 Experiments

We performed an experimental evaluation of multi-label methods to test a variety of methods in different contexts and in comparison with our classifier chains method.

Initially we carry out experiments to justify CC by comparing it to BR and BR-related methods. Following that, we compare ECC to well-known methods with proven success in multi-label classification. Finally we test scalability on large datasets.

Our set-up yields one of the most extensive evaluations in the multi-label literature. To the best of our knowledge, our collection of multi-label datasets represents the largest assembled so far in multi-label evaluation. We also employ a variety of evaluation methods all of which are supplemented by statistical significance tests.

Next we introduce the datasets and relevant statistics, evaluation measures and techniques, and following this, we review our experimental method. Then, we present and discuss the results.

### 8.1 Datasets

Table 2 displays multi-label datasets from a variety of domains, and their associated statistics. Here, *L*abel *Card*inality (LCard) is a standard measure of "multi-labelledness", introduced in [33]. It is simply the average number of labels associated with each example; defined for $N$ examples as:

$$\text{LCard} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} y_j^i$$

This measure gives a good idea of label *frequency*, but gives no indication of the *regularity* or *uniformity* of the labelling scheme.

In [27] we introduced the *P*roportion of *Uniq*ue label combinations (PUniq): the proportion of label sets which are unique across all examples. For a set of $N$ examples $D$:

$$\text{PUniq} = \frac{|\{\mathbf{y}|\exists!\mathbf{x} : (\mathbf{x},\mathbf{y}) \in D\}|}{N}$$

Let us further introduce another useful measure: the *P*roportion of label sets with the *M*aximum frequency (PMax). Assuming $\textsc{count}(\mathbf{y}, D)$ is the frequency that label combination $\mathbf{y}$ is found in the dataset $D$, we define:

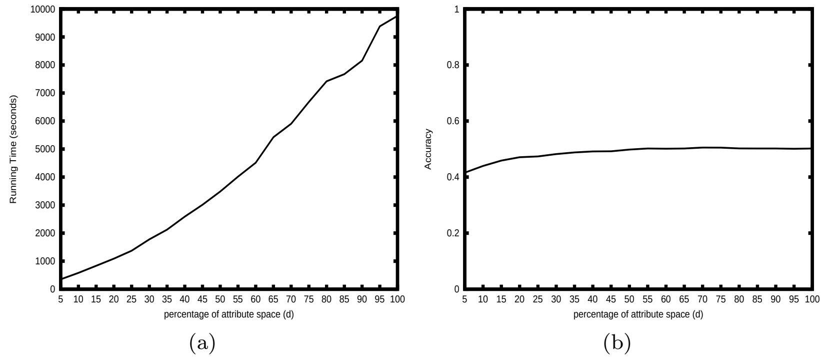

Fig. 3: ECC: subsets of attribute space and effect on running time (a) and accuracy (b) for 10 ensemble iterations

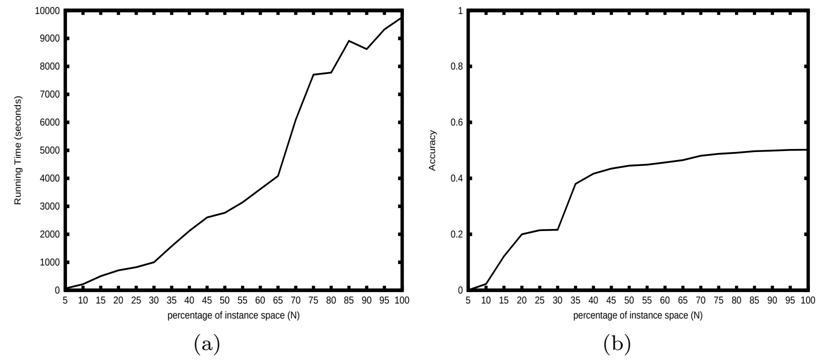

Fig. 4: ECC: subsets of instance space and effect on running time (a) and accuracy (b) for 10 ensemble iterations

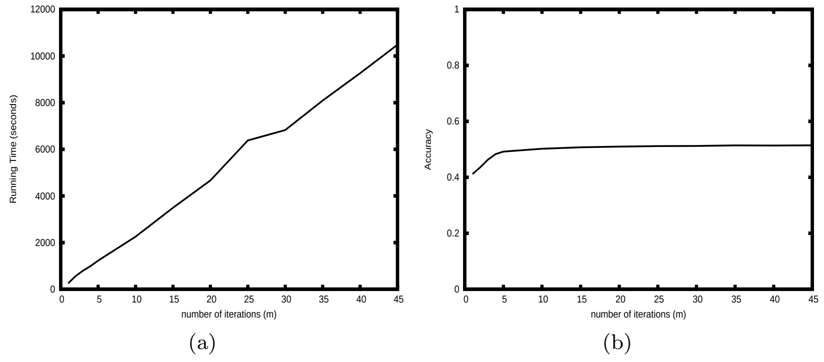

Fig. 5: ECC: number of ensemble iterations and the effect on running time (a) and accuracy (b) for full example and attribute spaces

Table 2: A collection of multi-label datasets and associated statistics; $n$ indicates numeric attributes; $b$ indicates binary attributes.

| | $N$ | $L$ | $d$ | LCard | PUniq | PMax | Type |
|---|---|---|---|---|---|---|---|
| Music | 593 | 6 | $72n$ | 1.87 | 0.046 | 0.137 | media |
| Scene | 2407 | 6 | $294n$ | 1.07 | 0.006 | 0.168 | media |
| Yeast | 2417 | 14 | $103n$ | 4.24 | 0.082 | 0.098 | biology |
| Genbase | 661 | 27 | $1185b$ | 1.25 | 0.048 | 0.257 | biology |
| Medical | 978 | 45 | $1449b$ | 1.25 | 0.096 | 0.158 | text |
| Slashdot | 3782 | 22 | $1079b$ | 1.18 | 0.041 | 0.139 | text |
| Enron | 1702 | 53 | $1001b$ | 3.38 | 0.442 | 0.096 | text |
| LangLog | 1460 | 75 | $1004b$ | 1.18 | 0.208 | 0.142 | text |
| Reuters | 6000 | 103 | $500n$ | 1.46 | 0.147 | 0.064 | text |
| OHSUMED | 13929 | 23 | $1002b$ | 1.66 | 0.082 | 0.084 | text |
| TMC2007 | 28596 | 22 | $500b$ | 2.16 | 0.047 | 0.087 | text |
| IMDB | 120919 | 28 | $1001b$ | 2.00 | 0.037 | 0.109 | text |
| Bibtex | 7395 | 159 | $1836b$ | 2.40 | 0.386 | 0.064 | text |
| MediaMill | 43907 | 101 | $120n$ | 4.38 | 0.149 | 0.054 | media |
| Delicious | 16105 | 983 | $500b$ | 19.02 | 0.981 | 0.001 | text |

$$\text{PMax} = \max_{\mathbf{y}} \frac{\text{count}(\mathbf{y}, D)}{N}$$

This represents the proportion of examples associated with the most frequently occurring label sets and, in combination with PUniq, gives an indication of regularity and uniformity of the labelling scheme, where high values indicate a skewed or irregular distribution.

We strove to include a considerable variety and scale of multi-label datasets. In total we use 15 datasets, with dimensions ranging from 6 to 983 labels, and up to over 120,000 examples. The datasets are roughly ordered by complexity ($N \times L \times d$) and divided between regular and large sizes. Note that *Delicious* could be considered a tag assignment problem outside the scope of multi-label classification. We include it here to help demonstrate the scalability of the algorithms we present and compare. We also introduce three new real-world multi-label text collections: *Slashdot*, which we collected from `http://slashdot.org`, *Lang*uage*Log* from `http://languagelog.ldc.upenn.edu/nll/`, and *IMDB* from `http://imdb.org` (data obtained from `http://www.imdb.com/interfaces#plain`). All datasets and further information about them can be found online[1].

## 8.2 Methods

We have selected well-known modern high-performing methods from the literature (which were discussed in Section 6) for our benchmark comparison. For easy reference, Table 3 lists all the algorithms used in the experiments, their corresponding abbreviation, some default parameters, and citation.

The parameter values given are those recommended by the authors of the relevant publications. In the case of `RAkEL`, we use two recommendations: `RAkEL`(1) which models

[1] See `http://meka.sourceforge.net/#datasets` and `http://mlkd.csd.auth.gr/multilabel.html#Datasets`

Table 3: Algorithms used in the experiments, and associated citations.

| Key | Algorithm | Parameters | Cit. |
|---|---|---|---|
| BR | Binary Relevance | | [33] |
| CC | Classifier Chains | | [27] |
| SMBR | Subset Mapped BR | | [28] |
| MBR | Meta-BR | | [14] |
| EBR | Ensembles of Binary Relevance | $m = 50/10$ | [27] |
| ECC | Ensembles of Classifier Chains | $m = 50/10$ | [27] |
| EPS | Ensembles of Pruned Sets | $m = 50/10, p = 1$—$5, n = 1$ | [25] |
| IBLR | Instance Based Logistic Regression | $k = 10$ | [4] |
| CLR | Calibrated Label Ranking | | [13] |
| RAkEL | RAndom K labEL subsets | (1): $m = 10, k = \frac{L}{2}$ (2): $m = 2L, k = 3$ | [34] |

relatively larger label sets, and RAkEL(2) which runs relatively more iterations. For other ensemble methods we use $m = 50$ iterations for regular datasets and $m = 10$ for large datasets. EPS selects randomly from the given range for $p$ at each iteration.

## 8.3 Evaluation measures

In any multi-label experiment, it is essential to include multiple and contrasting measures of evaluation because of the additional degrees of freedom that the multi-label setting introduces, as discussed recently in [6]. Thus a more complete picture of the capabilities of an algorithm is provided. This helps identifying algorithms which perform well across a range of evaluation measures.

There is an important division between *label-based* evaluation, which is carried out on a per-label basis, and *label set-based* evaluation, which evaluates label sets. We include several of both types of measures in our experimental evaluation.

When any predicted set of labels $\hat{\mathbf{y}}$ must match the true set of labels $\mathbf{y}$ *exactly* (label-set based evaluation), this is known as the *exact match* measure, or *0/1 loss* as a loss measure, as we use it:

$$0/1 \text{ LOSS} = 1 - \frac{1}{N} \sum_{i=1}^{N} 1_{\mathbf{y}^i = \hat{\mathbf{y}}^i}$$

When each label assignment is a separate binary evaluation (label-based evaluation), we have *Hamming loss*:

$$\text{HAMMING LOSS} = 1 - \frac{1}{NL} \sum_{i=1}^{N} \sum_{j=1}^{L} 1_{y^i_j = y^i_j}$$

0/1 LOSS can be very harsh, since any label set not predicted perfectly is given a zero score, whereas HAMMING LOSS tends to be very lenient due to the typical sparsity of multi-labelling, and ignores the multi-label problem as a whole. This illustrates the importance of multiple and contrasting measures.

The authors of [14] introduced a multi-label *accuracy* measure; a label set-based measure; as follows, for a set of $N$ test examples:

$$\textsc{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{y}^i \wedge \hat{\mathbf{y}}^i|}{|\mathbf{y}^i \vee \hat{\mathbf{y}}^i|}$$

The *F-measure* is the harmonic mean between precision and recall, common to information retrieval. It can be calculated from true positives, true negatives, false positives, and false negatives, based on a bit vector of predictions ($\hat{\mathbf{q}}$) and corresponding actual values ($\mathbf{q}$), as $\mathrm{F1}(\hat{\mathbf{q}}, \mathbf{q})$. To contrast with ACCURACY (label set-based evaluation) we use the F-measure macro averaged over labels (i.e. label-based evaluation), for $N$ test examples as:

$$\textsc{F-measure} = \frac{1}{L}\sum_{j=1}^{L}\mathrm{F1}([\hat{y}_j^1, \ldots, \hat{y}_j^N], [y_j^1, \ldots, y_j^N])$$

An additional consideration is the (very common) case, where a multi-label classifier initially outputs a vector $\hat{\mathbf{w}} \in \mathbb{R}^L$ of confidence values for each label, for example the `ECC` method (see Section 5), and a threshold function is applied to these outputs to produce a multi-label prediction $\hat{\mathbf{y}} = f_t(\hat{\mathbf{w}})$ (which can be evaluated under the above four measures, for example). Threshold functions are discussed in Section 8.4 below. However, another possibility is to use the confidence output vector itself for evaluation.

In [27] we introduced *log loss*, distinct from other measures because it evaluates confidence outputs directly and punishes over-confident errors more harshly, thus rewarding conservative prediction. The error is graded by the confidence at which it is predicted: predicting false positives with low confidence induces logarithmically less penalty than predicting with high confidence. This measure is important to contrast against other measures where guessing correctly with relatively little confidence can be rewarded. Given confidence outputs of $i, \ldots, N$ test examples, we have:

$$\textsc{Log-Loss} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\min(\textsc{log-loss}(\hat{w}_j^i, y_j^i), \ln(N))$$

where:

$$\textsc{log-loss}(\hat{w}, y) = -(\ln(\hat{w})y + \ln(1-\hat{w})(1-y))$$

We have used a dataset-dependent maximum of $\ln(N)$ to limit the magnitude of the penalty. Such a limit, as explained in [28], serves to smooth the values and prevent a small subset of poorly predicted labels from greatly distorting the overall error. Note that, as a loss metric, the best possible score for log loss is 0.0. This is a label-based evaluation measure.

In the analysis of running time we measure train and test times in seconds.

### 8.4 Threshold selection

Given a vector $\hat{\mathbf{w}} \in \mathbb{R}^L$ of real-valued confidence outputs, a multi-label prediction $\hat{\mathbf{y}}$ can be obtained under a threshold function $f_t(\hat{\mathbf{w}})$ such that:

$$\hat{y}_j = \begin{cases} 1 & \text{if } \hat{w}_j \geq t \\ 0 & \text{if } \hat{w}_j < t \end{cases}$$

In the context of text categorisation, [37] uses a threshold vector, i.e. a threshold $t_j$ for each $\hat{w}_j$. This approach has also been reviewed by [10] in the general multi-label context and is decidedly better that simply using an arbitrary threshold like 0.5. Our experience is that calibrating a single threshold $t$ to use across the entire evaluation is just as effective and more efficient. We calibrate the threshold $t$ as follows:

$$t = \operatorname*{argmin}_t \left\| \textsc{LCard}(D) - \left( \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} 1_{\hat{w}_j \geq t} \right) \right\|$$

where $N$ refers to the number of examples in the *test* set, so that the average observed label cardinality is close to the average predicted label cardinality.

We have found this method of threshold calibration to be more effective than an arbitrary threshold like 0.5, and robust across confidence vectors of different domains. Also, although empty-set predictions are always possible, they are very unlikely under this method.

## 8.5 Setup and method

We evaluate all algorithms using our open-source WEKA-based [15] software[2], which also provides a wrapper around the MULAN software[3] that contains additional methods. We use `SMO` (WEKA's implementation of Support Vector Machines based on the SMO algorithm [22]) as the base-classifier for problem transformation methods, with default parameters, implying a linear machine. Note that for multi-class transformations (i.e. `LC`-based methods) this implementation of `SMO` will compare each possible class label in a pairwise fashion internally using pairwise classification.

All experiments were run on 64 bit machines, allowing up to 2 GB RAM and one week of CPU time.

We apply the Nemenyi test [7] to indicate statistical significance. This test is based on methods' average rankings across datasets, and is appropriate for finding significant differences among multiple classifiers. We compare the methods listed in Section 8.2 under the various evaluation measures from Section 8.3, using (where appropriate) the threshold function described in Section 8.4. On the standard-sized datasets we carry out $5 \times 2$ fold cross validation (CV) and use the average result to display and test for significance. On large dataset we perform 60/40 train/test splits. Note that, in some cases, methods did not finish under the specified time and memory limits, i.e. yielded a DNF, (which we indicate with a missing result). In these cases, since a missing result affects all rankings, we do the Nemenyi test considering both the methods which completed on all datasets and again considering all the datasets upon which on methods completed. On large datasets, there were too many instances of DNF to obtain informative significance results.

Thus, for each evaluation measure, we display the values achieved and the rank pertaining to those values for each dataset; and we display the average rank for each method across all datasets and the rank of that value; and statistical significance such that $\{$ `a b` $\} \succ \{$ `d e` $\}$ indicates that methods `a` and `b` are both significantly favourable over methods `d` and `e`.

[2] available at `http://meka.sourceforge.net`
[3] available at `http://mulan.sourceforge.net`

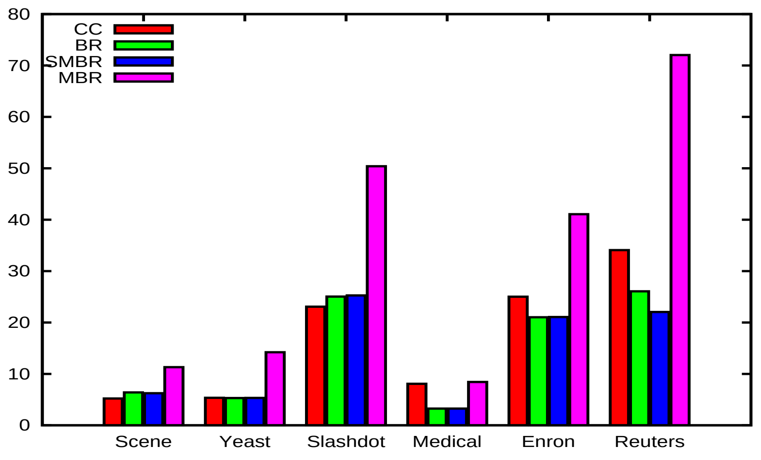

Fig. 6: `CC` vs. `BR` and related methods—average running times (seconds) over a selection of datasets (other datasets of similar dimensions show similar results).

## 8.6 Results

Initially, we compare standalone `CC` to `BR` and `BR`-related methods: a reproduction of the `MBR` method, and the `SMBR` method (refer to Section 6 and Table 3). Results of $5 \times 2$ CV in terms of ACCURACY and F-MEASURE are shown in Table 4. A selection of running times is graphed in Figure 6.

Secondly, we compare `CC` to `ECC` with both $m = 50$ and $m = 10$ iterations. Results of $5 \times 2$ CV in terms of ACCURACY and F-MEASURE are shown in Table 5.

As the main focus of our evaluation, we compare `ECC` to `EBR` and various state-of-the-art multi-label algorithms. Results of $5 \times 2$ CV for various evaluation measures are displayed in Table 6. Ensemble iterations (where relevant) are set to $m = 50$.

To consider scalability, we run the algorithms in a train-test scenario on large datasets. We use the `ECC` variation we presented in Section 7 to reduce redundancy in the learning space (75% and 50% random subsets of the example and attribute spaces, respectively). This strategy is also applied to `EBR`. For all ensembles, iterations are set to $m = 10$, and similarly we used `RAkEL`(2) because it scales better than `RAkEL`(1) and is thus more suited to large datasets. Results are displayed in Table 7.

To analyse running times we use an artificial dataset to control the size of the dataset dimensions. We create two datasets, one dataset to vary $L$ (with a constant $N = 5000$, $d = 500$ and $\text{LCARD}(D) \approx \sqrt{L}$ —a rough approximation suitable for measuring time complexity); and one dataset to vary $N$ (with a constant $L = 10$, $d = 20$, and $\text{LCARD}(D)$ as before). The process for generating such data is outlined in [26] (since we are only measuring running time, we do not detail the data composition here). In both experiments with artificial data, we consider methods without an ensemble, thus removing the variable $m$ of ensemble iterations. Running times are displayed in Figures 7 and 8. We emphasise that some lines in the plot end prematurely: some algorithms were unable to complete due to lack of time or memory.

Table 4: CC vs. BR and related methods; predictive performance (with ranks).

0/1 Loss$^{\downarrow}$:

| Dataset | BR | CC | MBR | SMBR |
|---|---|---|---|---|
| Music | 0.736 4 | 0.713 1 | 0.714 2 | 0.728 3 |
| Scene | 0.489 4 | 0.361 1 | 0.446 3 | 0.434 2 |
| Yeast | 0.856 4 | 0.788 1 | 0.854 3 | 0.836 2 |
| Genbase | 0.031 3 | 0.031 3 | 0.031 1 | 0.032 4 |
| Medical | 0.358 4 | 0.328 1 | 0.357 3 | 0.344 2 |
| Slashdot | 0.660 4 | 0.622 1 | 0.660 3 | 0.642 2 |
| Enron | 0.899 4 | 0.885 1 | 0.898 3 | 0.889 2 |
| LangLog | 0.784 4 | 0.781 2 | 0.784 4 | 0.781 1 |
| Reuters | 0.732 4 | 0.670 1 | 0.727 3 | 0.725 2 |
| avg. rank | 3.889 4 | 1.333 1 | 2.778 3 | 2.222 2 |

significance: CC $\succ$ { MBR BR } ; SMBR $\succ$ { BR } ;

Hamming Loss$^{\downarrow}$:

| Dataset | BR | CC | MBR | SMBR |
|---|---|---|---|---|
| Music | 0.201 2 | 0.223 4 | 0.198 1 | 0.204 3 |
| Scene | 0.110 3 | 0.107 2 | 0.106 1 | 0.130 4 |
| Yeast | 0.202 2 | 0.209 4 | 0.202 1 | 0.206 3 |
| Genbase | 0.001 3 | 0.001 3 | 0.001 3 | 0.002 4 |
| Medical | 0.011 3 | 0.011 1 | 0.011 3 | 0.013 4 |
| Slashdot | 0.049 1 | 0.051 3 | 0.050 2 | 0.057 4 |
| Enron | 0.060 3 | 0.060 4 | 0.060 2 | 0.060 1 |
| LangLog | 0.018 4 | 0.018 4 | 0.018 4 | 0.017 1 |
| Reuters | 0.011 2 | 0.011 3 | 0.011 2 | 0.016 4 |
| avg. rank | 2.556 2 | 3.111 3 | 2.111 1 | 3.111 3 |

significance: None

Accuracy:

| Dataset | BR | CC | MBR | SMBR |
|---|---|---|---|---|
| Music | 0.506 4 | 0.525 3 | 0.529 2 | 0.530 1 |
| Scene | 0.587 4 | 0.685 1 | 0.614 3 | 0.622 2 |
| Yeast | 0.496 3 | 0.527 1 | 0.497 2 | 0.491 4 |
| Genbase | 0.983 2 | 0.983 2 | 0.983 1 | 0.981 4 |
| Medical | 0.730 2 | 0.752 1 | 0.730 2 | 0.730 4 |
| Slashdot | 0.435 4 | 0.464 1 | 0.437 3 | 0.447 2 |
| Enron | 0.387 4 | 0.393 1 | 0.388 3 | 0.390 2 |
| LangLog | 0.106 2 | 0.108 1 | 0.106 2 | 0.106 4 |
| Reuters | 0.319 4 | 0.387 1 | 0.324 3 | 0.324 2 |
| avg. rank | 3.222 4 | 1.333 1 | 2.333 2 | 2.778 3 |

significance: CC $\succ$ { SMBR BR } ;

F-measure:

| Dataset | BR | CC | MBR | SMBR |
|---|---|---|---|---|
| Music | 0.599 4 | 0.621 2 | 0.637 1 | 0.612 3 |
| Scene | 0.681 3 | 0.708 1 | 0.692 2 | 0.654 4 |
| Yeast | 0.326 4 | 0.354 1 | 0.330 2 | 0.326 3 |
| Genbase | 0.773 2 | 0.773 2 | 0.777 1 | 0.763 4 |
| Medical | 0.362 3 | 0.372 1 | 0.366 2 | 0.348 4 |
| Slashdot | 0.330 1 | 0.329 3 | 0.330 2 | 0.309 4 |
| Enron | 0.197 3 | 0.198 1 | 0.197 2 | 0.188 4 |
| LangLog | 0.046 2 | 0.047 1 | 0.046 2 | 0.044 4 |
| Reuters | 0.222 3 | 0.250 1 | 0.228 2 | 0.214 4 |
| avg. rank | 2.778 3 | 1.444 1 | 1.778 2 | 3.778 4 |

significance: CC $\succ$ { SMBR } ; MBR $\succ$ { SMBR } ;

$\downarrow$ = loss measure (lower is better).

Table 5: CC vs. ECC; predictive performance (with ranks).

ACCURACY:

| Dataset | CC | | $\texttt{ECC}_{m=50}$ | | $\texttt{ECC}_{m=10}$ | |
|---|---|---|---|---|---|---|
| Music | 0.525 | 3 | 0.564 | 1 | 0.560 | 2 |
| Scene | 0.685 | 3 | 0.706 | 1 | 0.695 | 2 |
| Yeast | 0.527 | 3 | 0.536 | 1 | 0.531 | 2 |
| Genbase | 0.983 | 1 | 0.978 | 3 | 0.981 | 2 |
| Medical | 0.752 | 3 | 0.772 | 1 | 0.763 | 2 |
| Slashdot | 0.464 | 3 | 0.495 | 1 | 0.485 | 2 |
| Enron | 0.393 | 3 | 0.452 | 1 | 0.446 | 2 |
| LangLog | 0.108 | 3 | 0.148 | 1 | 0.143 | 2 |
| Reuters | 0.387 | 3 | 0.460 | 1 | 0.450 | 2 |
| avg. rank | 2.778 | 3 | 1.222 | 1 | 2.000 | 2 |

significance: $\texttt{ECC}_{m=50} \succ \{ \texttt{CC} \}$ ;

F-MEASURE:

| Dataset | CC | | $\texttt{ECC}_{m=50}$ | | $\texttt{ECC}_{m=10}$ | |
|---|---|---|---|---|---|---|
| Music | 0.621 | 3 | 0.665 | 1 | 0.665 | 2 |
| Scene | 0.708 | 3 | 0.746 | 1 | 0.734 | 2 |
| Yeast | 0.354 | 3 | 0.378 | 2 | 0.381 | 1 |
| Genbase | 0.773 | 1 | 0.749 | 3 | 0.766 | 2 |
| Medical | 0.372 | 2 | 0.374 | 1 | 0.372 | 3 |
| Slashdot | 0.329 | 3 | 0.357 | 1 | 0.352 | 2 |
| Enron | 0.198 | 3 | 0.210 | 2 | 0.215 | 1 |
| LangLog | 0.047 | 3 | 0.061 | 1 | 0.056 | 2 |
| Reuters | 0.250 | 3 | 0.304 | 1 | 0.291 | 2 |
| avg. rank | 2.667 | 3 | 1.444 | 1 | 1.889 | 2 |

significance: $\texttt{ECC}_{m=50} \succ \{ \texttt{CC} \}$ ;

## 9 Discussion

We first discuss the value of classifier chains as opposed to related binary approaches. Secondly, we look at the performance of ECC in comparison with some competitive methods described in Section 6, both for standard-sized and large datasets. Finally we discuss running times.

### 9.1 The value of Classifier Chains

The results in Table 4 show the value of CC's chaining method compared with related classifiers. Overall CC improves convincingly over both the default BR method and related methods MBR and SMBR. We see that SMBR performed poorest under HAMMING LOSS and F-MEASURE. This is to be expected, since these are label-based evaluation measures, whereas SMBR predicts *label sets* like LC and therefore performs better under label-set based evaluation measures (0/1 LOSS and ACCURACY). The reverse is true for the performance of BR and MBR. It is not surprising that CC under-performed on HAMMING LOSS, nor is it particularly discouraging: a method obtains good performance under HAMMING LOSS by ignoring label dependencies, i.e. ignoring the multi-label dimension of the data.

The training times of BR and LC (Figure 6) support the theory presented in Section 3. BR is naturally the fastest. In practice the complexity added to BR by CC is not significant, except in cases where $L$ is particularly large relative to $N$ or $d$ (e.g. *Medical*). SMBR also involves only minimal increases in running time over BR. MBR's two-stage

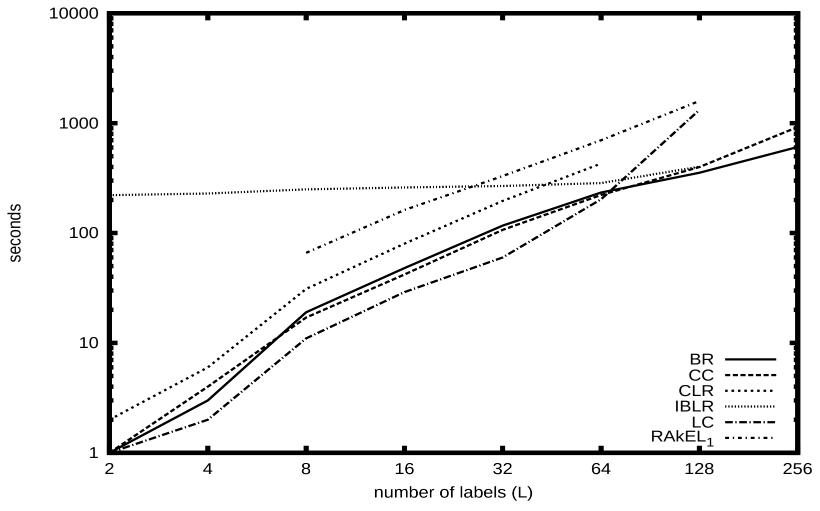

Fig. 7: Running times (seconds) on artificial datasets where $L = 2, 4, \ldots, 256$ (note the logarithmic scale).

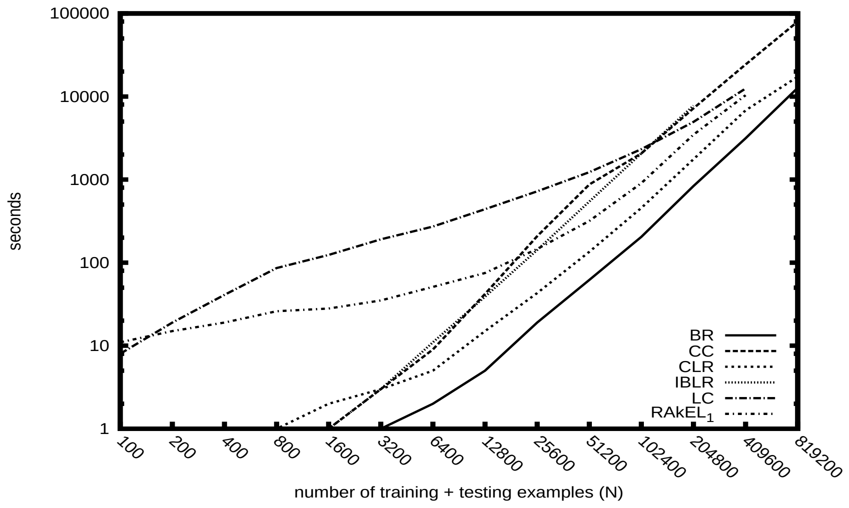

Fig. 8: Running times (seconds) on artificial datasets where $N = 100, 200, \ldots, 819200$ (note the logarithmic scale).

stacking process means that its training times are about twice those of BR on the datasets we consider.

### 9.2 The value of Ensembles of Classifier Chains

As we expect, using ensembles has a significant positive effect on predictive performance, since they significantly reduce the chance of a poor chain-ordering or error propagation negatively affecting overall predictive performance. In Table 5 we see that `ECC`, under both 10 and 50 iterations, performed better than `CC` in almost all cases (with the exception of *Genbase*), and often by a considerable margin, with more iterations clearly correlating to better performance.

### 9.3 `ECC` as compared with other methods

In comparison with other methods from the literature, `ECC` yielded superior predictive performance overall (see Table 6). It performed best on more evaluation measures than any other method, and obtains the highest rankings overall, and no method performed statistically significantly better than `ECC` in any case.

In multi-label classification, it can not be expected that a method performs best over all types of evaluation measures, except in cases where the method is tailored to each and run separately. Therefore it comes as no surprise that different methods performed best under different measures.

`EPS` and `RAkEL`(1) performed best under 0/1 LOSS. This is because these methods are the most "`LC`-like" methods, and `LC` is tailored towards performance under this measure. The case for ACCURACY is similar.

But the overall advantage of `ECC` is clear; it proved competitive not only under label set-based measures but also label-based measures like LOG LOSS and HAMMING LOSS.

The nearest-neighbor-based methods `IBLR` performed well on LOG LOSS relative to other measures: it votes relatively more conservatively than other methods and is rewarded for it. `CLR` performed well in several situations but its complexity prohibited its completion on anything but small-$L$ datasets.

The performance benefit of classifier chains is further demonstrated in the comparison between `ECC` and `EBR`: these methods only differ in that the latter does not use the classifier chains model across its classifiers. `ECC` did better overall, and particularly so in the context of, for example, the *Reuters* dataset, and—surprisingly—under the HAMMING LOSS evaluation measure.

On the other hand, `EBR` often performed surprisingly well against the other methods, for example on the *Enron* and *Slashdot* datasets. This is interesting, considering that this method is simply an ensemble of baseline `BR`. This result adds further support to the case for binary relevance methods.

### 9.4 Scaling up to large datasets

Table 7 shows that both binary methods (`EBR` and `ECC`) performed strongly on large datasets. Most surprising is that `EBR` often achieves even higher predictive performance than `ECC`—even on label-set based 0/1 LOSS—and, in some cases, even benchmark `BR` performs best. This indicates that, where large numbers of training examples are involved, efforts to model label correlations may become unnecessary or even detrimental. This is also suggested by the comparison to other methods that invest complexity into modelling label correlations (RAkEL, EPS) although there are cases where they perform better than EBR, namely RAkEL on *MediaMill* and EPS on *TMC07*. BR's occasional high performance relative to EBR indicates that the ensemble voting and thresholding process could be improved. We leave a thorough investigation for future work, but we speculate that, because train/test splits are used on large datasets, any time-ordered structure would remain in place (as opposed to multiple random cross validation splits as done on regular datasets), and the train and test sets can therefore exhibit a much greater difference in label cardinality. This difference creates a problem when choosing a threshold (which BR does not rely upon) so that the label cardinality of the predicted data approximates that of the training data.

Evaluation-measure specific performance is also apparent on large datasets: EPS performed well on 0/1 LOSS, BR on HAMMING LOSS, IBLR on LOG LOSS, and ECC particularly on F-MEASURE, but also on ACCURACY and 0/1 LOSS.

Overall, the combined best ranks of the binary relevance methods outnumber those of the other methods combined. This may be partially due to the fact that other methods were unable to complete on all datasets, but this is a point in itself: only binary relevance methods were efficient enough to complete on all datasets within time and memory bounds.

## 9.5 Time Complexity

In Figure 7 we see how different methods scale with respect to the number of labels ($L$). CC only diverges from BR when $L > 128$. Thus we see that under most multi-label contexts (see our dataset collection), the complexity of CC is very close to BR. LC is unable to complete when $L > 128$, and we see it scaling sharply at this point. LC-based RAkEL appears to scale similarly to BR, but runs out of memory for $L = 256$. As expected, the space complexity of the PW method (CLR) is very sensitive to $L$, and it is intractable when $L > 64$ (at $L = 128$ this method would need 16256 models). IBLR's running times are initially high, but almost constant with respect to $L$, although this changes abruptly and the method does not finish when $L = 256$ due to lack of memory.

In Figure 8 we see how different methods scale with respect to $N$. LC again does not complete, nor does RAkEL. The $k$NN-based IBLR also does not scale to large $N$ (greater than 204800), which is expected since $k$NN is $N$-sensitive where the number of training examples grows with $N$. Although we assume that $L$ is inherently limited in scope and will not grow without bounds, this does not apply to $N$, and, as $N$ becomes large, $k$NN methods will reach their natural limits of scalability. Only CC, BR, and the PW method CLR (which needs in this case only $L(L-1)/2 = 45$ models) complete when $N = 819200$. The time complexity of CC does not vary greatly from BR. We note that the worst case theoretical complexity of PS is equal to that of LC.

Overall, the binary relevance methods BR and CC are the best candidates for scaling to very large problems. CLR scales well with respect to $N$ but not with respect to $L$ and vice versa for $k$NN-based IBLR. Note that ECC is at most $m\times$CC in terms of complexity for $m$ iterations; this can be reduced substantially by subsampling the learning space, as we explained in Section 7.

## 10 Conclusions

This paper presented a novel chaining method for multi-label classification. We derived this method from the binary relevance method, which we argued has many advantages over more sophisticated methods currently in use, including reduced computational complexity. By passing label correlation information along a chain of classifiers, our method counteracts the disadvantages of the binary relevance method and obtains high predictive performance compared to more complex methods while maintaining low computational complexity. We used an ensemble of classifier chains to eliminate the chance of poorly ordered chains, and thus further augment predictive performance. Since ensembles of binary relevance methods give rise to considerable redundancy in the learning space, we took advantage of this to create a more scalable ensemble classifier.

In an extensive empirical multi-label evaluation, we compared our classifier chains method against a variety of other methods. The chaining method proved superior to related methods and, in an ensemble, proved highly competitive with state-of-the-art multi-label methods. Unlike specialised algorithm-adaptations, ensembles of classifier chains can be seen as a general 'off-the-shelf' approach, needing no parameter configuration, and having robust performance across a range of data. As well as achieving high performance, they are highly scalable, and were able to complete on the largest datasets that we considered.

## References

1. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognition **37**(9), 1757–1771 (2004)
2. Breiman, L.: Bagging predictors. Machine Learning **24**(2), 123–140 (1996)
3. Cheng, W., Dembczyński, K., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML '10: 27th International Conference on Machine Learning. Omnipress, Haifa, Israel (2010)
4. Cheng, W., Hüllermeier, E.: Combining instance-based learning and logistic regression for multilabel classification. Machine Learning **76**(2-3), 211–225 (2009). DOI 10.1007/s10994-009-5127-5
5. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. Lecture Notes in Computer Science **2168** (2001)
6. Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence in multi-label classification. In: Workshop Proceedings of Learning from Multi-Label Data, pp. 5–12. Haifa, Israel (2010)
7. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7**, 1–30 (2006)
8. Dimou, A., Tsoumakas, G., Mezaris, V., Kompatsiaris, I., Vlahavas, I.: An empirical study of multi-label learning methods for video annotation. In: Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing. IEEE (2009)
9. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: In Advances in Neural Information Processing Systems 14, pp. 681–687. MIT Press (2001)
10. Fan, R.E., Lin, C.J.: A study on threshold selection for multi-label classification. Tech. rep., National Taiwan University (2007). URL `http://www.csie.ntu.edu.tw/~cjlin/papers/threshold.pdf`
11. Freund, Y., Schapire, R.E.: A short introduction to boosting. Japanese Society for Artificial Intelligence **14**(5), 771–780 (1999)
12. Fürnkranz, J.: Round robin classification. Machine Learning **2**, 721–747 (2002)
13. Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., Brinker, K.: Multilabel classification via calibrated label ranking. Machine Learning **73**(2), 133–153 (2008). DOI 10.1007/s10994-008-5064-8

14. Godbole, S., Sarawagi, S.: Discriminative methods for multi-labeled classification. In: PAKDD '04: Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 22–30. Springer (2004)
15. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Peter, R., Witten, I.H.: The weka data mining software: An update. SIGKDD Explorations **11**(1) (2009)
16. Hsu, D., Kakade, S.M., Langford, J., Zhang, T.: Multi-label prediction via compressed sensing. In: NIPS '09: Neural Information Processing Systems 2009 (2009)
17. Ji, S., Tang, L., Yu, S., Ye, J.: Extracting shared subspace for multi-label classification. In: KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pp. 381–389. ACM (2008). DOI 10.1145/1401890.1401939
18. Kiritchenko, S.: Hierarchical text categorization and its application to bioinformatics. Ph.D. thesis, Queen's University, Kingston, Canada (2005)
19. Loza Mencía, E., Fürnkranz, J.: Efficient pairwise multilabel classification for large-scale problems in the legal domain. In: ECML-PKDD '08: European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 50–65. Springer (2008). DOI 10.1007/978-3-540-87481-2_4
20. McCallum, A.K.: Multi-label text classification with a mixture model trained by EM. In: Association for the Advancement of Artificial Intelligence workshop on text learning (1999)
21. Petrovskiy, M.: Paired comparisons method for solving multi-label learning problem. In: HIS '06: Sixth International Conference on Hybrid Intelligent Systems. IEEE (2006). DOI 10.1109/HIS.2006.264925
22. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in Large Margin Classifiers, pp. 61–74. MIT Press (1999)
23. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**(1), 81–106 (1986)
24. Ráez, A.M., López, L.A.U., Steinberger, R.: Adaptive selection of base classifiers in one-against-all learning for large multi-labeled collections. In: EsTAL: 4th International Conference on Advances in Natural Language Processing, pp. 1–12 (2004)
25. Read, J., Pfahringer, B., Holmes, G.: Multi-label classification using ensembles of pruned sets. In: ICDM'08: Eighth IEEE International Conference on Data Mining, pp. 995–1000. IEEE (2008)
26. Read, J., Pfahringer, B., Holmes, G.: Generating synthetic multi-label data streams. In: MLD '09: 1st ECML/PKDD 2009 Workshop on Learning from Multi-Label Data (2009)
27. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. In: ECML '09: 20th European Conference on Machine Learning, pp. 254–269. Springer (2009)
28. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning **37**(3), 297–336 (1999)
29. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning **39**(2/3), 135–168 (2000)
30. Spyromitros, E., Tsoumakas, G., Vlahavas, I.: An empirical study of lazy multilabel classification algorithms. In: SETN '08: Fifth Hellenic conference on Artificial Intelligence, pp. 401–406. Springer, Berlin, Heidelberg (2008)
31. Sun, L., Ji, S., Ye, J.: Hypergraph spectral learning for multi-label classification. In: KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pp. 668–676. ACM (2008). DOI 10.1145/1401890.1401971
32. Tai, F., Lin, H.T.: Multi-label classification with principle label space transformation. In: Workshop Proceedings of Learning from Multi-Label Data. Haifa, Israel (2010)
33. Tsoumakas, G., Katakis, I.: Multi label classification: An overview. International Journal of Data Warehousing and Mining **3**(3), 1–13 (2007)
34. Tsoumakas, G., Vlahavas, I.P.: Random k-labelsets: An ensemble method for multilabel classification. In: ECML '07: 18th European Conference on Machine Learning, pp. 406–417. Springer (2007)
35. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning **2**(73), 185–214 (2008). DOI 10.1007/s10994-008-5077-3
36. Yan, R., Tesic, J., Smith, J.R.: Model-shared subspace boosting for multi-label classification. In: KDD '07: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, pp. 834–843. ACM (2007). DOI 10.1145/1281192.1281281
37. Yang, Y.: A study on thresholding strategies for text categorization. In: Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval, pp. 137–145. ACM Press (2001)

38. Zhang, M.L., Zhou, Z.H.: A k-nearest neighbor based algorithm for multi-label classification. In: GnC '05: IEEE International Conference on Granular Computing, 2005, pp. 718–721. IEEE (2005)
39. Zhang, M.L., Zhou, Z.H.: ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7), 2038–2048 (2007)

Table 6: Methods on standard-sized datasets.

0/1 LOSS$^{\downarrow}$:

| Dataset | CLR | EBR | ECC | EPS | IBLR | RAkEL(1) | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| Music | 0.740 7 | 0.729 6 | 0.695 4 | 0.676 1 | 0.704 5 | 0.690 3 | 0.689 2 |
| Scene | 0.391 4 | 0.477 7 | 0.367 2 | 0.316 1 | 0.371 3 | 0.400 5 | 0.403 6 |
| Yeast | 0.970 7 | 0.858 6 | 0.814 4 | 0.766 1 | 0.812 3 | 0.795 2 | 0.825 5 |
| Genbase | 0.116 7 | 0.048 3 | 0.048 4 | 0.106 6 | 0.097 5 | 0.036 1.5 | 0.036 1.5 |
| Medical | | 0.347 3 | 0.328 1 | 0.362 5 | 0.632 6 | 0.345 2 | 0.354 4 |
| Slashdot | | 0.652 4 | 0.620 3 | 0.577 1 | 0.887 6 | 0.614 2 | 0.668 5 |
| Enron | | 0.881 4 | 0.873 3 | 0.857 1 | 0.922 6 | 0.872 2 | 0.896 5 |
| LangLog | | 0.796 4 | 0.796 5 | 0.760 1 | 0.856 6 | 0.764 2 | 0.780 3 |
| Reuters | | 0.737 6 | 0.687 3 | 0.622 1 | 0.731 5 | 0.643 2 | 0.719 4 |
| avg. rank | 6.000 7 | 4.667 5 | 3.111 3 | 1.889 1 | 4.889 6 | 2.389 2 | 3.944 4 |

significance: ECC $\succ$ { CLR } ; EPS $\succ$ { IBLR EBR CLR } ; RAkEL(1) $\succ$ { CLR EBR } ;

HAMMING LOSS$^{\downarrow}$:

| Dataset | CLR | EBR | ECC | EPS | IBLR | RAkEL(1) | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| Music | 0.214 7 | 0.205 6 | 0.200 1 | 0.203 5 | 0.202 3 | 0.202 4 | 0.200 2 |
| Scene | 0.101 6 | 0.110 7 | 0.095 3 | 0.088 1 | 0.091 2 | 0.100 5 | 0.100 4 |
| Yeast | 0.226 7 | 0.210 6 | 0.209 5 | 0.207 2 | 0.205 1 | 0.209 4 | 0.207 3 |
| Genbase | 0.004 5 | 0.002 3.5 | 0.002 3.5 | 0.007 7 | 0.005 6 | 0.002 2 | 0.001 1 |
| Medical | | 0.011 2 | 0.010 1 | 0.013 5 | 0.025 6 | 0.011 4 | 0.011 3 |
| Slashdot | | 0.050 2 | 0.049 1 | 0.051 3 | 0.073 6 | 0.051 4 | 0.053 5 |
| Enron | | 0.056 2 | 0.056 1 | 0.059 4 | 0.067 6 | 0.057 3 | 0.061 5 |
| LangLog | | 0.026 4 | 0.026 5 | 0.025 3 | 0.031 6 | 0.021 2 | 0.019 1 |
| Reuters | | 0.016 5 | 0.014 3 | 0.014 4 | 0.017 6 | 0.012 2 | 0.011 1 |
| avg. rank | 6.250 7 | 4.167 5 | 2.611 1 | 3.778 4 | 4.667 6 | 3.333 3 | 2.778 2 |

significance: ECC $\succ$ { CLR } ; RAkEL(1) $\succ$ { CLR } ; RAkEL(2) $\succ$ { CLR } ;

ACCURACY:

| Dataset | CLR | EBR | ECC | EPS | IBLR | RAkEL(1) | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| Music | 0.557 5 | 0.547 7 | 0.564 3 | 0.564 4 | 0.551 6 | 0.567 2 | 0.569 1 |
| Scene | 0.695 4 | 0.624 7 | 0.706 3 | 0.739 1 | 0.706 2 | 0.685 5 | 0.684 6 |
| Yeast | 0.514 7 | 0.525 6 | 0.536 4 | 0.545 1 | 0.538 3 | 0.544 2 | 0.536 5 |
| Genbase | 0.142 7 | 0.978 3 | 0.978 4 | 0.945 6 | 0.950 5 | 0.982 1 | 0.982 2 |
| Medical | | 0.758 3 | 0.772 1 | 0.743 4 | 0.480 6 | 0.760 2 | 0.742 5 |
| Slashdot | | 0.469 4 | 0.495 2 | 0.514 1 | 0.241 6 | 0.494 3 | 0.451 5 |
| Enron | | 0.443 3 | 0.452 2 | 0.440 4 | 0.360 6 | 0.457 1 | 0.410 5 |
| LangLog | | 0.147 4 | 0.148 3 | 0.174 1 | 0.008 6 | 0.151 2 | 0.119 5 |
| Reuters | | 0.382 4 | 0.460 2 | 0.496 1 | 0.360 5 | 0.445 3 | 0.337 6 |
| avg. rank | 5.750 7 | 4.556 5 | 2.667 3 | 2.556 2 | 5.000 6 | 2.333 1 | 4.444 4 |

significance: EPS $\succ$ { IBLR } ; RAkEL(1) $\succ$ { IBLR } ;

LOG LOSS$^{\downarrow}$:

| Dataset | CLR | EBR | ECC | EPS | IBLR | RAkEL(1) | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| Music | 3.633 4 | 3.624 3 | 3.077 2 | 3.845 5 | 2.620 1 | 4.938 7 | 4.865 6 |
| Scene | 3.420 7 | 1.926 4 | 1.425 2 | 1.607 3 | 1.329 1 | 2.495 6 | 2.435 5 |
| Yeast | 8.772 2 | 12.560 5 | 9.280 3 | 10.097 4 | 6.196 1 | 14.866 6 | 15.685 7 |
| Genbase | 4.080 7 | 0.160 2 | 0.159 1 | 0.512 5 | 0.632 6 | 0.176 3 | 0.189 4 |
| Medical | | 1.792 2 | 1.722 1 | 2.099 3 | 4.802 6 | 2.284 4 | 2.698 5 |
| Slashdot | | 3.710 2 | 3.422 1 | 3.925 4 | 3.766 3 | 5.128 5 | 5.975 6 |
| Enron | | 9.610 2 | 9.252 1 | 11.116 3 | 11.395 4 | 13.927 5 | 15.462 6 |
| LangLog | | 6.397 3 | 6.394 2 | 6.182 1 | 14.063 6 | 7.271 4 | 8.091 5 |
| Reuters | | 6.971 3 | 5.551 1 | 6.543 2 | 7.763 5 | 7.588 4 | 8.971 6 |
| avg. rank | 5.000 6 | 2.889 2 | 1.556 1 | 3.333 3 | 3.667 4 | 4.889 5 | 5.556 7 |

significance: ECC $\succ$ { RAkEL(2) RAkEL(1) CLR } ; EBR $\succ$ { RAkEL(2) } ;

F-MEASURE:

| Dataset | CLR | EBR | ECC | EPS | IBLR | RAkEL(1) | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| Music | 0.668 1 | 0.649 7 | 0.665 2 | 0.661 3 | 0.656 6 | 0.660 5 | 0.660 4 |
| Scene | 0.736 6 | 0.705 7 | 0.746 3 | 0.764 1 | 0.755 2 | 0.738 4 | 0.737 5 |
| Yeast | 0.405 4 | 0.372 7 | 0.378 6 | 0.420 2 | 0.415 3 | 0.424 1 | 0.390 5 |
| Genbase | 0.121 7 | 0.747 4 | 0.749 3 | 0.614 6 | 0.656 5 | 0.763 1 | 0.761 2 |
| Medical | | 0.373 2 | 0.374 1 | 0.290 5 | 0.194 6 | 0.369 4 | 0.369 3 |
| Slashdot | | 0.352 2 | 0.357 1 | 0.326 5 | 0.155 6 | 0.350 3 | 0.344 4 |
| Enron | | 0.209 3 | 0.210 2 | 0.141 6 | 0.143 5 | 0.210 1 | 0.205 4 |
| LangLog | | 0.061 1 | 0.061 2 | 0.053 4 | 0.014 6 | 0.057 3 | 0.051 5 |
| Reuters | | 0.276 3 | 0.304 1 | 0.237 4 | 0.205 6 | 0.278 2 | 0.233 5 |
| avg. rank | 4.500 6 | 4.000 3 | 2.333 1 | 4.000 3 | 5.000 7 | 2.667 2 | 4.111 5 |

significance: ECC $\succ$ { IBLR } ; RAkEL(1) $\succ$ { IBLR } ;

$\downarrow$ = loss measure (lower is better).

Table 7: Methods on large datasets.

0/1 LOSS$^{\downarrow}$:

| Dataset | BR | CLR | EBR | ECC | EPS | IBLR | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| TMC2007 | 0.740 1 | 0.853 6 | 0.779 4 | 0.767 3 | 0.740 1 | 0.797 5 | 0.744 2 |
| Ohsumed | 0.828 4 | 0.914 6 | 0.781 1 | 0.784 2 | 0.797 3 | 0.937 7 | 0.830 5 |
| MediaMill | 0.928 3 | | 0.958 5 | 0.938 4 | | 0.922 2 | 0.920 1 |
| Bibtex | 0.846 3 | | 0.852 4 | 0.846 1 | 0.846 2 | 0.942 6 | 0.867 5 |
| IMDB | 0.999 4 | | 0.899 1 | 0.938 2 | | 0.956 3 | |
| Delicious | 0.998 2 | | 1.000 4 | 1.000 3 | 0.996 1 | | |
| avg. rank | 2.833 3 | 6.000 7 | 3.167 4 | 2.500 2 | 1.750 1 | 4.600 6 | 3.250 5 |

HAMMING LOSS$^{\downarrow}$:

| Dataset | BR | CLR | EBR | ECC | EPS | IBLR | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| TMC2007 | 0.064 1 | 0.078 7 | 0.069 4 | 0.068 3 | 0.069 5 | 0.078 6 | 0.068 2 |
| Ohsumed | 0.074 3 | 0.088 6 | 0.062 1 | 0.063 2 | 0.074 4 | 0.097 7 | 0.075 5 |
| MediaMill | 0.032 2 | | 0.035 3 | 0.041 5 | | 0.038 4 | 0.032 1 |
| Bibtex | 0.015 1 | | 0.017 3 | 0.016 2 | 0.022 5 | 0.023 6 | 0.017 4 |
| IMDB | 0.072 1 | | 0.082 2 | 0.095 3 | | 0.097 4 | |
| Delicious | 0.018 1 | | 0.025 2 | 0.026 4 | 0.025 3 | | |
| avg. rank | 1.500 1 | 6.500 7 | 2.500 2 | 3.167 4 | 4.250 5 | 5.400 6 | 3.000 3 |

ACCURACY:

| Dataset | BR | CLR | EBR | ECC | EPS | IBLR | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| TMC2007 | 0.541 3 | 0.506 6 | 0.510 5 | 0.517 4 | 0.549 1 | 0.479 7 | 0.549 2 |
| Ohsumed | 0.361 6 | 0.374 5 | 0.423 3 | 0.426 1 | 0.424 2 | 0.230 7 | 0.383 4 |
| MediaMill | 0.390 4 | | 0.410 2 | 0.380 5 | | 0.426 1 | 0.400 3 |
| Bibtex | 0.329 3 | | 0.350 1 | 0.348 2 | 0.306 5 | 0.161 6 | 0.328 4 |
| IMDB | 0.005 4 | | 0.211 3 | 0.221 2 | | 0.236 1 | |
| Delicious | 0.134 3 | | 0.181 1 | 0.178 2 | 0.057 4 | | |
| avg. rank | 3.833 5 | 5.500 7 | 2.500 1 | 2.667 2 | 3.000 3 | 4.400 6 | 3.250 4 |

LOG LOSS$^{\downarrow}$:

| Dataset | BR | CLR | EBR | ECC | EPS | IBLR | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| TMC2007 | 13.1 6 | 16.3 7 | 8.2 3 | 9.4 4 | 6.9 2 | 4.1 1 | 10.2 5 |
| Ohsumed | 14.6 4 | 17.4 5 | 29.0 6 | 30.8 7 | 6.8 2 | 5.0 1 | 8.3 3 |
| MediaMill | 31.5 5 | | 24.1 2 | 25.0 3 | | 10.1 1 | 30.4 4 |
| Bibtex | 19.3 4 | | 55.3 6 | 54.8 5 | 14.2 1 | 14.6 2 | 15.7 3 |
| IMDB | 21.9 4 | | 18.6 2 | 21.7 3 | | 6.0 1 | |
| Delicious | 158.9 4 | | 130.5 2 | 110.9 1 | 155.7 3 | | |
| avg. rank | 4.5 6 | 6.0 7 | 3.5 3 | 3.8 5 | 2.0 2 | 1.2 1 | 3.8 4 |

F-MEASURE:

| Dataset | BR | CLR | EBR | ECC | EPS | IBLR | RAkEL(2) |
|---|---|---|---|---|---|---|---|
| TMC2007 | 0.547 4 | 0.577 1 | 0.476 6 | 0.496 5 | 0.573 3 | 0.434 7 | 0.577 2 |
| Ohsumed | 0.370 5 | 0.407 3 | 0.416 1 | 0.414 2 | 0.366 6 | 0.127 7 | 0.392 4 |
| MediaMill | 0.036 5 | | 0.040 4 | 0.048 2 | | 0.179 1 | 0.042 3 |
| Bibtex | 0.319 4 | | 0.345 1 | 0.337 2 | 0.227 5 | 0.139 6 | 0.334 3 |
| IMDB | 0.011 4 | | 0.037 3 | 0.055 1 | | 0.049 2 | |
| Delicious | 0.105 3 | | 0.112 2 | 0.138 1 | 0.029 4 | | |
| avg. rank | 4.167 5 | 2.000 1 | 2.833 3 | 2.167 2 | 4.500 6 | 4.600 7 | 3.000 4 |

$\downarrow$ = loss measure (lower is better).