Classifier Chains for Multi-label Classification

Jesse Read, Bernhard Pfahringer, Geoff Holmes, Eibe Frank

Department of Computer Science
The University of Waikato
Hamilton, New Zealand
{jmr30,bernhard,geoff,eibe}@cs.waikato.ac.nz

Abstract. The widely known binary relevance method for multi-label classification, which considers each label as an independent binary problem, has been sidelined in the literature due to the perceived inadequacy of its label-independence assumption. Instead, most current methods invest considerable complexity to model interdependencies between labels. This paper shows that binary relevance-based methods have much to offer, especially in terms of scalability to large datasets. We exemplify this with a novel chaining method that can model label correlations while maintaining acceptable computational complexity. Empirical evaluation over a broad range of multi-label datasets with a variety of evaluation metrics demonstrates the competitiveness of our chaining method against related and state-of-the-art methods, both in terms of predictive performance and time complexity.

1 Introduction

The traditional data mining task of *single-label* classification, also known as *multi-class* classification, associates an instance x with a single label l from a previously known finite set of labels L. A single-label dataset D is composed of n examples $(x_1, l_1), (x_2, l_2), \cdots, (x_n, l_n)$. The *multi-label* classification task associates a *subset* of labels $S \subseteq L$ with each instance. A multi-label dataset D is therefore composed of n examples $(x_1, S_1), (x_2, S_2), \cdots, (x_n, S_n)$. The multi-label problem is receiving increased attention and is relevant to many domains such as text classification [10, 2], and genomics [19, 16].

A common approach to multi-label classification is by way of *problem transformation*, whereby a multi-label problem is transformed into one or more single-label problems. In this fashion, a single-label classifier can be employed to make single-label classifications, and these are then transformed back into multi-label representations. Prior problem transformation approaches have employed algorithms such as Support Vector Machines [2], Naive Bayes [5] and k Nearest Neighbor methods [19].

The alternative to problem transformation is to modify an existing single-label algorithm directly for the purpose of multi-label classification. Some well known approaches involve decision trees [16] and AdaBoost [10]. Algorithm adaption may be as simple as using a problem transformation method internally, or

collecting prediction confidences and using a threshold to predict the multi-labels associated with prediction confidences that lie above the threshold. Both of these approaches can be generalised to other single-label classifiers.

By abstracting away from a specific classifier, *external* problem transformation allows greater flexibility. Any single-label classifier can be used to suit requirements. Depending on the problem context, some classifiers may demonstrate better performance than others. Moreover, external problem transformation methods can also be implemented specifically to a particular algorithm or easily integrated with meta or ensemble frameworks.

There are several families of problem transformation methods that can be found in the multi-label literature. These methods arise from one or more fundamental problem transformation approaches that either form the core of more complex frameworks or are used as modifications to other algorithms. Here we review two fundamental methods.

The most well known problem transformation method is the *binary relevance* method (`BM`) [13, 2, 19]. `BM` transforms any multi-label problem into one binary problem for each label. Hence this method trains $|L|$ binary classifiers $C_1, \cdots, C_{|L|}$. Each classifier C_j is responsible for predicting the 0/1 association for each corresponding label $l_j \in L$.

`BM` is mentioned throughout the literature but consistently sidelined on the grounds of its assumption of label independence. That is to say, during its transformation process, `BM` ignores label correlations that exist in the training data. The argument is that, due to this information loss, `BM`'s predicted label sets are likely to contain either too many or too few labels, or labels that would never co-occur in practice.

We argue that `BM`-based methods have a lot to offer. The chaining method we present in this paper shows that the above issues can be overcome and are outweighed by the advantages of this method and any methods based closely upon it.

Another fundamental problem transformation method is the *label combination method*, or *label power-set method*, (`CM`), which has been the focus of several recent works [15, 8]. The basis of this method is to combine entire label sets into atomic (single) labels to form a single-label problem for which the set of possible single labels represents all distinct label subsets in the original multi-label representation. Each (x, S) is transformed into (x, l) where l is the atomic label representing a distinct label subset. In this way, `CM`-based methods directly take into account label correlations. A disadvantage of these methods, however, is their worst-case time complexity.

The consensus view in the literature is that it is crucial to take into account label correlations during the classification process [3, 2, 15, 8, 11, 18, 4]. However as the size of multi-label datasets grows, most methods struggle with the exponential growth in the number of possible correlations. Consequently, these methods are able to be more accurate on small datasets, but are not as applicable to larger datasets. This necessarily restricts their usefulness as many multi-label contexts involve large numbers of examples and labels.

The paper is structured as follows. We outline the advantages of BM-based methods and present our classifier chains method CC, which overcomes disadvantages of the basic binary method. We then introduce an ensemble framework for classifier chains called ECC. Finally, we demonstrate the performance of our methods under empirical evaluation on a wide range of datasets with various evaluation measures.

The main contributions of this paper are:

- We present Classifier Chains (CC) and Ensembles of Classifier Chains (ECC)
- We introduce an evaluation metric and new datasets for multi-label classification
- We present an extensive experimental evaluation to demonstrate the effectiveness of using Classifier Chains.

2 In Defence of the Binary Method

Although BM's disadvantages are widely acknowledged, its advantages are rarely mentioned. BM is theoretically simple and intuitive. Its assumption of label independence makes it suited to contexts where new examples may not necessarily be relevant to any known labels or where label relationships may change over the test data; even the label set L may be altered dynamically – making BM ideal for active learning and data stream scenarios.

However the most important and widely relevant advantage of BM is its low computational complexity compared to other methods. Given a constant number of examples, BM scales linearly with the size of the known label set L. This set is defined in the dataset and generally limited in scope: generally $|L| < |X|$, where X is the feature space. If L is very large, or not defined prior to classification, the problem is better approached as a tag-assignment or hierarchical problem, which are beyond the scope of this paper.

CM-based methods, on the other hand, have an upper bound complexity of $\min(|D|, 2^{|L|})$, due to the exponentially expanding number of possible combinations with increasing $|L|$ (D is the training set). All methods which model all label correlations will suffer this complexity. Note that just modelling all pair-wise label correlations is $O(|L|^2)$.

Although BM involves $|L|$ single label problems, each problem only involves two classes. Depending on the dataset, CM may have to deal with thousands or tens of thousands of classes. Other methods of transformation resulting in a single problem will have to produce decisions involving at least $|L|$ classes, which may imply greater than linear complexity.

Because BM'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, thus only requiring $|D|$ instances in memory at any point (over $|L|$ processors, or $|L|$ iterations).

In the next section we present our new binary method, CC, which overcomes the label independence assumption of BM while maintaining acceptable computational complexity.

3 The Classifier Chain Model (CC)

The Classifier Chain model (CC) involves $|L|$ binary classifiers as in BM. Classifiers are linked along a chain where each classifier deals with the binary relevance problem associated with label $l_j \in L$. The feature space of each link in the chain is extended with the 0/1 label associations of all previous links. The training procedure is outlined in Figure 1. Recall the notation for a training example (x, S), where $S \subseteq L$ is represented by binary feature vector $(l_1, l_2, \cdots, l_{|L|}) \in \{0,1\}^{|L|}$, and x is an instance feature vector.

TRAINING($D = \{(x_1, S_1), \cdots, (x_n, S_n)\}$)
1 **for** $j \in 1 \cdots |L|$
2 **do** ▷ single-label transformation and training
3 $D' \leftarrow \{\}$
4 **for** $(x, S) \in D$
5 **do** $D' \leftarrow D' \cup ((x, l_1, \cdots, l_{j-1}), l_j)$
6 ▷ train C_j to predict binary relevance of l_j
7 $C_j : D' \rightarrow l_j \in \{0, 1\}$

Fig. 1: CC's training phase for dataset D and label set L.

Hence a chain $C_1, \cdots, C_{|L|}$ of binary classifiers is formed. Each classifier C_j in the chain is responsible for learning and predicting the binary association of label l_j given the feature space, augmented by all prior binary relevance predictions in the chain $l_1, \cdots, l_{j-1}$. The classification process begins at C_1 and propagates along the chain: C_1 determines $Pr(l_1|x)$ and every following classifier $C_2 \cdots C_{|L|}$ predicts $Pr(l_j|x_i, l_1, \ldots, l_{j-1})$. This classification process is outlined in Figure 2.

CLASSIFY(x)
1 $Y \leftarrow \{\}$
2 **for** $j \leftarrow 1$ **to** $|L|$
3 **do** $Y \leftarrow Y \cup (l_j \leftarrow C_j : (x, l_1, \cdots, l_{j-1}))$
4 **return** (x, Y) ▷ the classified example

Fig. 2: CC's prediction phase for a test instance x.

This chaining method passes label information between classifiers, allowing CC to take into account label correlations and thus overcoming the label independence problem of BM. However, CC still retains advantages of BM including

low memory and runtime complexity. Although an average of $|L|/2$ features is added to each instance, because $|L|$ is invariably limited in practice, this has negligible consequences on complexity, as demonstrated in Section 6.

However in terms of computational complexity CC can be very close to BM, depending on the total number of labels and the complexity of the underlying learner. BM's complexity is $O(|L| \times f(|X|, |D|))$, where $f(|X|, |D|)$ is the complexity of the underlying learner. Using the same notation, CC's complexity is $O(|L| \times f(|X| + |L|, |D|))$, i.e. a penalty is incurred for having $|L|$ additional attributes. As demonstrated in the experiments in Section 6, in practice this penalty tends be small in many cases. For instance, assuming a linear base learner, CC's complexity simplifies to $O(|L| \times |X| \times |D| + |L| \times |L| \times |D|)$, where the first term dominates as long as $|L| < |X|$, which we expect. In this case the effective complexity of CC is then $O(|L| \times |X| \times |D|)$, which is identical to BM's complexity. CC's complexity will be worse than BM's whenever $|L| > |X|$ holds.

Also, although the chaining procedure implies that CC cannot be parallelized, it can be serialized and therefore still only requires a single binary problem in memory at any point in time – a clear advantage over other methods.

The order of the chain itself clearly has an effect on accuracy. Although there exist several possible heuristics for selecting a chain order for CC, we instead solve the issue by using an ensemble framework with a different random chain ordering for each iteration. In the next section, we present this framework.

4 Ensembles of Classifier Chains (ECC)

The classifier presented in this section is an Ensemble of Classifier Chains (ECC). Ensembles are well known for their effect of increasing overall accuracy and overcoming over-fitting, as well as allowing parallelism. They have successfully been used in various multi-label problems [10, 8, 15, 16].

Note that binary methods are occasionally referred to as ensemble methods because they involve 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*.

ECC trains m CC classifiers $C_1, C_2, \cdots, C_m$. Each C_k is trained with:

- a random chain ordering (of L); and
- a random subset of D.

Hence each C_k model is likely to be unique and able to give different multi-label predictions. These predictions are summed by label so that each label receives a number of votes. A threshold is used to select the most popular labels which form the final predicted multi-label set.

Each kth individual model (of m models) predicts vector $y_k = (l_1, \cdots, l_{|L|}) \in \{0, 1\}^{|L|}$. The sums are stored in a vector $W = (\lambda_1, \cdots, \lambda_{|L|}) \in \mathbb{R}^{|L|}$ such that $\lambda_j = \sum_{k=1}^{m} l_j \in y_k$. Hence each $\lambda_j \in W$ represents the sum of the votes for label $l_j \in L$. We then normalise W to W^{norm}, which represents a distribution of scores for each label in $[0, 1]$. A threshold is used to choose the final multi-label

set Y such that $l_j \in Y$ where $\lambda_j \geq t$ for threshold t. Hence the relevant labels in Y represent the final multi-label prediction.

This is a generic voting scheme and it is straightforward to apply an ensemble of any multi-label problem transformation method. We can therefore apply `BM` under this same scheme to create an Ensemble of the Binary Method (`EBM`). This is carried out identically to `ECC` (except that chain ordering has no effect on `BM`). As far as we are aware, this scheme has not been previously evaluated in the literature.

5 Related Work

Using labels in the feature space has been approached in the past by Godbole and Sarawagi [2]. In the main contribution of their work, the authors stacked `BM` classification outputs along with the full original feature space into a separate meta classifier, thereby creating a two-stage classification process. Similar to `CC`, this process is able to take into account label correlations, but their meta classifier implies an extra iteration of both training and test data as well as internal classifications on the training data to acquire the label outputs for this meta step. In contrast, `CC` only requires a single training iteration like `BM`, and uses labels directly from the training data without any internal classification. We evaluate and compare Godbole and Sarawagi's meta-stacking (`MS`) in our experimental evaluation.

A method reviewed in [9] maps confidence predictions of a single-label classifier to actual label subsets (observed in the training data) using Hamming distance. The subset with the shortest Hamming distance to the predictions is chosen as the predicted set. This procedure can also be applied to the binary outputs of `BM`. Hence we use this Subset Mapping method (`SM`) in our experimental evaluation.

The work in [3] is based upon the binary approach but their algorithm adds a second part for deriving a low-dimensional shared subspace in order to model label correlations. This is computationally expensive, despite an approximation algorithm, which is reflected in their experimental setup where they randomly select 1000 data points for training – relatively small sets. Similarly [11] uses a computationally complex hypergraph method to model label correlations and, despite a proposed approximate formulation, induces high computational complexity.

MLkNN [19] is a nearest-neighbor based method with similar time complexity to `BM`. MLkNN performs well against `BM`, but in [12], it did not perform as well as `RAKEL` (see below), which we use in our evaluation.

A boosting algorithm is introduced by [18] that aims to reduce complexity by reducing redundancy in the learning space and sharing models between labels. Binary models are trained on subsets of the instance and feature spaces, i.e. a random forest paradigm is used. This is a good example of how the complexity of the binary approach can be significantly reduced, and supports the conclusion of this paper that binary methods have been underrated.

The binary pairwise problem has also been employed for multi-label classification. This is the *one-vs-one* approach, as opposed to the *one-vs-rest* approach used by BM, therefore requiring $|L|^2$ classifiers as opposed to $|L|$. [7] accompanies each pairwise classifier with two probabilistic models to isolate the overlapping feature space. They cite a computational bottleneck for this method for large datasets. Another pairwise approach [4] works with large datasets when used with simple and efficient perceptrons, although this method is only able to provide a ranking and not actual multi-label classification sets. A related approach is taken in [1] which can conduct classification by using a virtual label to separate relevant and irrelevant labels. This method performed well against BM in terms of ranking evaluation, but only marginally in classification.

While label rankings can in most cases be turned into a multi-label classification, the reverse is not always true. Both BM and CC, for example, cannot naturally provide prediction confidences, and therefore are unable to provide a ranking. ECC can output a ranking directly from its voting scheme, although this ranking is only coincidental as a means to achieve a classification. Ranking methods and evaluation of ranking performance itself falls outside the scope of this paper.

Several ensemble approaches have been developed based on the common problem transformation methods introduced in Section 1, particularly CM due to its inherent ability to take into account label correlations.

A good example is the RAKEL system by Tsoumakas and Vlahavas [15]. For m iterations of the training data, RAKEL draws a random subset of size k from all labels L and trains a CM classifier using these labels. A simple voting process determines the final classification set. Using appropriate values of m and k, RAKEL was shown to be better than BM and CM.

HOMER [14] is a computationally efficient multi-label classification method specifically designed for large multi-label datasets. Its efficiency is due to hierarchically splitting up the label set L using a modified k-means algorithm, and solving each subproblem individually. We note that the authors use *Naive Bayes* as their base classifier to further reduce complexity. In our experiments we use the more computationally demanding *SVM*s, known for their predictive performance, particularly on text data.

In [8] we presented EPS: an ensemble method that uses pruning to reduce the computational complexity of CM, and an instance duplication method to reduce error rate as compared to CM and other methods. This method proved to be particularly competitive in terms of efficiency.

6 Experiments

We perform an experimental comparison based on several experimental setups designed to test a variety of methods in different contexts. Initially we carry out experiments to justify the value of CC by comparing to related methods and then later compare ECC to other state-of-the-art methods.

We consider our evaluation one of the most extensive in the multi-label literature. To the best of our knowledge, our collection of multi-label datasets represents the largest one so far in multi-label evaluation, and we have used four evaluation methods. First we introduce some evaluation measures, the datasets and relevant statistics, and following this, we review our experimental method and setup, and then present the results.

6.1 Evaluation Measures

It is essential to include several evaluation measures in multi-label evaluation. Given the extra label dimension, it is otherwise possible to optimise for certain evaluation measures. We use four different evaluation measures.

Multi-label classification requires a different measure of *accuracy* from standard single-label (multi-class) classification. Recall that for each ith classified instance, Y_i is the predicted set of labels, and S_i is the actual set. It is possible to measure accuracy by example (instance i is correct if $S_i = Y_i$), or by individual label (each $l \in Y_i$ is a separate evaluation), but in practice the former tends to be overly harsh and the latter overly lenient. Instead, we use accuracy as defined in [13]:

$$Accuracy = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|S_i \cap Y_i|}{|S_i \cup Y_i|}$$

Accuracy is micro-averaged across all examples. As a contrast we include *macro-averaged F-measure*, where the average is calculated per label and then averaged across all labels. The F-measure is the harmonic mean between precision and recall, common to information retrieval. If p_j and r_j are the precision and recall for all $l_j \in Y_i$ from $l_j \in S_i$, the macro-averaged F-measure is:

$$F1_{macro} = \frac{1}{|L|} \sum_{j=1}^{|L|} \frac{2 \times p_j \times r_j}{(p_j + r_j)}$$

We also evaluate using the average *area under the precision recall curve* ($AU(\overline{PRC})$). Instead of setting a fixed threshold, the threshold is varied on the confidence predictions W_i^{norm} (see Section 4) in steps $0.00, 0.02, \cdots, 1.00$, thus producing different precision and recall values for each label. The average across all labels is the average area under the precision recall curve. More information about this measure can be found in [16].

Finally we introduce the use of *log loss*, distinct from other measures because it punishes worse errors more harshly, rewarding conservative prediction. The error is graded by the confidence at which it was predicted: predicting false positives with low confidence induces logarithmically less penalty than predicting with high confidence. Therefore, again, we use the confidence predictions for each label $\lambda \in W_i^{norm}$ to compare to the actual value of each label $l_j \in S_i$:

$$LogLoss = \frac{1}{|D|}\sum_{i=1}^{|D|}\sum_{j=1}^{|L|} -max\Big(log(\lambda_j)l_j + log(1-\lambda_j)(1-l_j), log\frac{1}{|D|}\Big)$$

We have used a dataset-dependent maximum of $log(\frac{1}{|D|})$ to limit the magnitudes of penalty. Such a limit, as explained in [9], serves to smooth the values and prevent a small subset of poorly predicted labels from greatly distorting the overall error. Note that, as opposed to the other measures, the best possible score for the log loss is 0.0.

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

6.2 Datasets

Table 1 displays datasets from a variety of domains and their associated statistics. *L*abel *Card*inality ($LCard$) is a standard measure of "multi-labelled-ness" introduced in [13]. It is simply the average number of labels relevant to each instance. The *P*roportion of *Dist*inct label combinations ($PDist$) is simply the number of distinct label subsets relative to the total number of examples:

$$LCard(D) = \frac{\sum_{i=1}^{|D|}|S_i|}{|D|} \qquad PDist(D) = \frac{|\{S|\exists(x,S) \in D\}|}{|D|}$$

Table 1: A collection of multi-label datasets and associated statistics; n indicates numeric attributes. D_T is the training split used in some experiments.

	$\|D\|$	$\|L\|$	$\|X\|$	$LCard(D)$	$PDist(D)$	Type	D_T	$\|D_T\| \times \|L\| \times \|X\|$
Scene	2407	6	294n	1.07	0.006	media	1211	2.14E+06
Yeast	2417	14	103n	4.24	0.082	biology	1500	2.16E+06
Medical	978	45	1449	1.25	0.096	text	652	4.25E+07
Slashdot	3782	22	1079	1.18	0.041	text	1891	4.49E+07
Enron	1702	53	1001	3.38	0.442	text	1135	6.02E+07
Reuters	6000	103	500n	1.46	0.147	text	3000	1.55E+08
OHSUMED	13929	23	1002	1.66	0.082	text	6965	1.61E+08
TMC2007	28596	22	500	2.16	0.047	text	21519	2.37E+08
MediaMill	43907	101	120n	4.38	0.149	media	30993	3.76E+08
Bibtex	7395	159	1836	2.40	0.386	text	3698	1.08E+09
IMDB	95424	28	1001	1.92	0.036	text	47712	1.34E+09
Delicious	16105	983	500	19.02	0.981	text	12920	6.35E+09

We strived to include a considerable variety and scale of multi-label datasets. In total we use 12 datasets, with dimensions ranging from 6 to 983 labels, and from less than 1,000 examples to almost 100,000. The datasets are roughly ordered by complexity ($|D_T| \times |L| \times |X|$) and divided between regular and

large sizes. Included are two new real-world multi-label text collections: *Slashdot*, which we collected from `http://slashdot.org`, 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 at various sources[1].

6.3 Algorithms

For easy reference, Table 2 lists all the algorithms used in the experiments, their corresponding abbreviation, and any relevant citation (where citations are absent, the method has been introduced in this paper).

Table 2: Algorithms used in the experiments, and associated citations.

BM-based algorithms		Ensemble algorithms	
BM Binary Method	[13]	EBM Ensembles of Binary Method	
CM Chaining Method		ECC Ensembles of Classifier Chains	
SM Subset Mapping	[9]	EPS Ensembles of Pruned Sets	[8]
MS Meta Stacking	[2]	RAKEL RAndom K labEL subsets	[15]

6.4 Setup and Method

Our default experimental setup is as follows. We evaluate all algorithms under a WEKA-based [17] framework running under Java JDK 1.6 with the following settings. Support Vector Machines are used as the internal classifier using WEKA's SMO implementation with default parameters. Ensemble iterations are set to 10. Evaluation is done in the form of 5×2 fold cross validation on each dataset and the corrected paired t-test [6] determines significance under a value of 0.05. The exception to this is the experiments on large datasets where cross validation is too intensive for some methods, and a train/test split is used instead. These splits are shown in Table 1 where D_T is the training set (and therefore $(D \setminus D_T)$ the test set). Experiments are run on 64 bit machines, allowing up to 2 GB RAM *per ensemble iteration*.

The thresholds for *all* ensemble voting schemes, necessary for determining accuracy and the macro F-measure, are set as following, where D_T is the training set and a classifier H_t has made predictions for test set D_S under threshold t:

$$t = \underset{\{t \in 0.00, 0.001, \cdots, 1.00\}}{\arg\min} \left| LCard(D_T) - LCard(H_t(D_S)) \right| \quad (1)$$

This is the closest approximation of the label cardinality of the training set to the predictions made on the test set. This implies a close balance between precision

[1] `http://www.cs.waikato.ac.nz~/jmr30/#datasets` and `http://mlkd.csd.auth.gr/multilabel.html#Datasets`

and recall and therefore benefits accuracy and F-measure. It also avoids ad-hoc or arbitrary thresholds or intensive internal cross-validation.

All ensemble methods involve subsampling for the individual models. EBM,ECC, and EPS subsample the training set (we set 67% for each model), while RAKEL subsamples the label set according to its k parameter.

For RAKEL we always set parameter $k = \frac{|L|}{2}$ and for EPS we set $p = 1$ and n is set according to the $LCard(D_T)$ training set statistic. Both these algorithms allow a trade-off between predictive performance and training time costs and vice versa: using smaller k for RAKEL and higher p EPS will lead to reduced computational complexity for both algorithms. However, in these experiments we optimise for predictive performance. Results in the relevant papers [8] and [15] show that our choice of parameter values generally provides highest accuracy. One of the notable advantages of ECC is that it requires no additional parameters other than the generic ensemble parameters which we set as above.

6.5 Results

Initially we compare standalone CC to BM and BM-related methods: a reproduction of the MS method, and the SM method. CC is used as the base for determining statistical significance. Results for accuracy and macro-averaged F-measure are shown in Table 3 (the other evaluation methods are not appropriate because not all these methods can supply confidence predictions). Train times are graphed in Figure 3.

Secondly, we perform an experiment comparing ensemble implementations. We compare EBM and ECC to the state-of-the-art algorithms EPS and RAKEL. Statistical significance is taken against ECC. Results for all evaluation measures are displayed in Table 4 and train times are graphed in Figure 4.

Finally we compare ensembles separately on large datasets, for which we use train/test splits for evaluation. Results for predictive performance are displayed in Table 5; train and test times are displayed in Table 6. DNF indicates that the experiment Did Not Finish within one week under the available resources.

Table 3: Binary Methods - Predictive Performance.

	Accuracy				Macro F-measure			
Dataset	CC	BM	SM	MS	CC	BM	SM	MS
Scene	**67.3**	59.1 •	63.0	61.9 •	**0.696**	0.685	0.666	0.694
Yeast	**51.5**	49.6	50.4	49.8	**0.346**	0.326	0.327	0.331
Slashdot	**46.7**	43.4 •	44.7 •	43.6 •	0.327	**0.329**	0.298	0.328
Medical	**75.1**	73.0 •	73.1	73.1 •	**0.377**	0.364	0.321 •	0.370
Enron	39.5	38.6	**40.3**	38.8	**0.198**	0.197	0.144 •	**0.198**
Reuters	**39.6**	31.9 •	33.6 •	32.4 •	**0.245**	0.224 •	0.194 •	0.229 •

⊕, • statistically significant improvement or degradation vs. CC

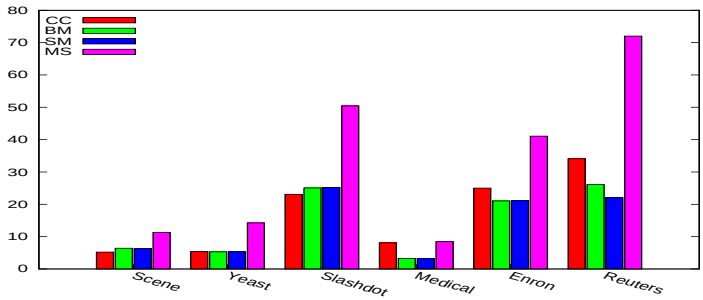

Fig. 3: Binary Methods - Train times (average seconds).

Table 4: Ensemble Methods.

	Accuracy				Log Loss			
Dataset	ECC	EBM	EPS	RAK.	ECC	EBM	EPS	RAK.
Scene	70.8	68.6 •	**73.7** ⊕	72.7 ⊕	**1.32**	1.97 •	1.41	1.78 •
Yeast	53.3	52.7 •	**54.9** ⊕	54.3	9.41	11.48 •	**9.16**	10.26 •
Slashdot	51.0	50.9	50.9	**51.4**	**3.45**	3.94 •	3.81 •	4.41 •
Medical	**77.6**	76.7	75.1 •	76.2	**1.87**	1.93	2.08 •	2.19
Enron	44.6	44.2	44.5	**45.9**	**10.84**	11.00	12.15 •	12.00 •
Reuters	44.7	36.0 •	**49.6** ⊕	45.3	7.52	8.33 •	**7.09** ⊕	8.23 •
	Macro F-measure				$AU(\overline{PRC})$			
Dataset	ECC	EBM	EPS	RAK.	ECC	EBM	EPS	RAK.
Scene	0.742	0.729 •	**0.763**	0.750	0.778	0.706 •	**0.780**	0.736 •
Yeast	0.362	0.364	**0.420** ⊕	0.413 ⊕	**0.645**	0.618 •	0.643	0.623 •
Slashdot	0.343	0.346	0.336	**0.353**	**0.514**	0.464 •	0.498	0.443 •
Medical	**0.386**	0.382	0.324 •	0.377	**0.789**	0.782	0.752 •	0.744 •
Enron	0.201	0.201	0.155 •	**0.206**	**0.488**	0.481 •	0.440 •	0.453 •
Reuters	**0.286**	0.264 •	0.264 •	0.282	0.347	0.311 •	**0.378** ⊕	0.330 •

⊕, • statistically significant improvement or degradation vs. ECC

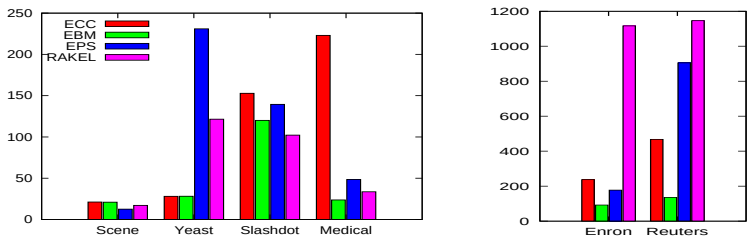

Fig. 4: Ensemble Methods - Train times (average seconds).

Table 5: Large Datasets - Ensembles - Predictive Performance.

	Accuracy				Log Loss			
Dataset	ECC	EBM	EPS	RAK.	ECC	EBM	EPS	RAK.
OHSUMED	41.1	41.39	**41.98**	41.55	5.36	**5.35**	6.17	6.65
TMC2007	**53.03**	52.74	52.30	52.85	5.70	6.02	**4.87**	5.82
Bibtex	**35.50**	35.10	34.13	DNF	11.95	**11.88**	12.73	DNF
MediaMill	**40.39**	39.80	38.20	31.40	**25.37**	27.05	26.52	29.97
IMDB	**24.88**	1.92	DNF	2.06	**13.34**	21.30	DNF	20.61
Delicious	**17.93**	16.93	9.41	DNF	**119.92**	128.30	133.34	DNF
	Macro F-measure				$AU(\overline{PRC})$			
Dataset	ECC	EBM	EPS	RAK.	ECC	EBM	EPS	RAK.
OHSUMED	0.378	0.379	0.376	**0.398**	0.495	0.499	**0.506**	0.501
TMC2007	0.551	0.548	**0.561**	0.557	**0.620**	**0.620**	0.614	**0.620**
Bibtex	**0.324**	0.313	0.257	DNF	**0.437**	0.433	0.423	DNF
MediaMill	**0.395**	0.366	0.338	0.309	**0.523**	0.518	0.482	0.413
IMDB	**0.221**	0.075	DNF	0.080	**0.329**	0.023	DNF	0.025
Delicious	0.154	0.133	0.038	DNF	**0.182**	0.158	0.095	DNF

Table 6: Large Datasets - Ensembles - Train and Test Times.

	Train Times (seconds)				Test Times (seconds)			
	ECC	EBM	EPS	RAK.	ECC	EBM	EPS	RAK.
OHSUMED	**5E3**	**5E3**	8E3	**5E3**	2E2	**7E1**	2E4	3E3
TMC2007	5E4	5E4	**3E3**	3E4	**9E1**	**9E1**	2E3	3E3
Bibtex	3E3	**2E3**	7E3	DNF	**2E3**	**2E3**	1E4	DNF
MediaMill	1E5	2E5	**6E4**	2E5	1E3	**7E2**	2E5	2E5
IMDB	4E5	**3E5**	DNF	3E5	2E3	**9E2**	DNF	1E4
Delicious	1E5	1E5	**3E2**	DNF	1E4	6E3	**1E2**	DNF

Shown in E notation where 5E3 $\approx$ 5000 seconds.

7 Discussion

7.1 The Value of Classifier Chains

The value of CC's chaining method can be seen in Table 3 where it is compared to related classifiers. CC improves convincingly over both the default BM method and related methods MS and SM 10 out of 12 times, and in many cases the difference is statistically significant. Hence the results justify using CC as a base method.

The training times of BM and CM (Figure 3) support the theory presented in Section 2. BM is naturally the fastest. This complexity is exceeded only marginally by CC. On smaller datasets, the effect is even negligible, and tiny variances in runtime conditions cause CC to run marginally faster in some cases - similarly to SM. SM also involves only minimal overhead over BM. Not surprisingly, because of MS's two-stage stacking process, its train times are about twice that of BM.

7.2 Ensemble Methods on Regular Datasets

In Table 4, we see that ECC competes well against the other ensemble methods. Although in some cases these methods demonstrate better performance than ECC, such gains are not paralleled under all evaluation measures. Under both log loss and $AU\overline{PRC}$, ECC improves over other methods in four out of six cases, and the improvement is for the most part statistically significant.

As expected, the timing results again weigh in favour of EBM and ECC, especially for small $|L|$. This is shown in Figure 4. An exception is the *Medical* dataset which has a very high $|L| : |D|$ ratio, spiking ECC's build time. EPS's reductions to complexity are considerable in some cases, but sporadic and not theoretically bounded as low as those of the BM-based methods, which are closely related to the dataset constants $|L|$ and $|D|$.

7.3 Ensemble Methods on Large Datasets

We presented ECC as an efficient method for multi-label classification, so the experiment on large datasets is important to justify this claim. Results are displayed in Table 5. Although the t-test could not be used on the train-test evaluation, we can still see overall superiority for ECC, even more so on regular datasets. ECC shows a clear majority of wins over the CM-based methods on all measures of predictive performance. EBM also performs particularly well in this experiment, indicating that, particularly on larger datasets, the disadvantages of BM-based methods are outweighed by the large number of examples they can train on.

Only the ECC method performs satisfactorily on the *IMDB* dataset under the experiment setup. The other methods suffer problems. EPS's pruning mechanism fails and the individual models of EBM and RAKEL predict too many empty sets. The different chain orderings prevent this effect in ECC.

Again the BM-based methods show an overall advantage in time costs (Table 6). These datasets are much larger than those typically approached in the literature. Although ECC and EBM are not always the fastest, they are the most

consistent, and are the only methods to complete on every dataset under the time and memory constraints of the experiment. Again, EPS's reductions to training time are in some cases effective, but not reliable. For example, on *IMDB*, where there are few outlying label combinations to prune (indicated by a low $PDist$ value), EPS's pruning mechanism is ineffective, resulting in DNF. On *Delicious*, the effect is the opposite: EPS prunes away far too much information, resulting in particularly poor predictive performance. This is arguably an improvement over RAKEL (DNF), but only about half as accurate as the binary methods.

7.4 Summary

CM-based methods and other methods that intensively model label correlations obviously have a place in multi-label classification, especially for datasets of relatively small dimensions. On larger datasets, however, not only does it become computationally challenging to model all label correlations, but there are no significant predictive advantages in doing so. These other methods work hard to model the label correlations in the training data but end up sacrificing individual label accuracy. ECC models correlations using an approach which is efficient and not prone to over-fitting, and for this reason performs strongly over a wide range of datasets and evaluation measures.

8 Conclusions

This paper presented a novel chaining method for multi-label classification. We based this method on the binary relevance method, which we argued has many advantages over more sophisticated current methods, especially in terms of time costs. By passing label correlation information along a chain of classifiers, our method counteracts the disadvantages of the binary method while maintaining acceptable computational complexity. An ensemble of classifier chains can be used to further augment predictive performance.

Using a variety of multi-label datasets and evaluation measures, we carried out empirical evaluations against a range of algorithms. Our classifier chains method proved superior to related methods, and in an ensemble scenario was able to improve on state-of-the-art methods, particularly on large datasets. Despite other methods using more complex processes to model label correlations, ensembles of classifier chains can achieve better predictive performance and are efficient enough to scale up to very large problems.

References

1. Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153, November 2008.
2. Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *PAKDD '04: 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30. Springer, 2004.

3. Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. Extracting shared subspace for multi-label classification. In *KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 381–389. ACM, 2008.
4. Eneldo Loza Mencía and Johannes Fürnkranz. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML-PKDD '08: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases)*, pages 50–65. Springer-Verlag, 2008.
5. Andrew K. McCallum. Multi-label text classification with a mixture model trained by EM. In *Association for the Advancement of Artificial Intelligence workshop on text learning*, 1999.
6. Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning*, 52(3):239–281, September 2003.
7. Mikhail Petrovskiy. Paired comparisons method for solving multi-label learning problem. In *HIS '06: Sixth International Conference on Hybrid Intelligent Systems*, page 42. IEEE, 2006.
8. Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. In *ICDM'08: Eighth IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2008.
9. Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.
10. Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
11. Liang Sun, Shuiwang Ji, and Jieping Ye. Hypergraph spectral learning for multi-label classification. In *KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 668–676. ACM, 2008.
12. K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *ISMIR '08: 9th International Conference on Music Information Retrieval*, 2008.
13. Grigorios Tsoumakas and Ioannis Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2007.
14. Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, 2008.
15. Grigorios Tsoumakas and Ioannis P. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *ECML '07: 18th European Conference on Machine Learning*, pages 406–417. Springer-Verlag, 2007.
16. Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 2(73):185–214, August 2008.
17. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.
18. Rong Yan, Jelena Tesic, and John R. Smith. Model-shared subspace boosting for multi-label classification. In *KDD '07: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 834–843. ACM, 2007.
19. Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. In *GnC '05: IEEE International Conference on Granular Computing*, pages 718–721. IEEE, 2005.