Dynamic Classifier Chain with Random Decision Trees [⋆]

Moritz Kulessa[1] and Eneldo Loza Mencía[1]

Knowledge Engineering Group, Technische Universtität Darmstadt, Germany
mkulessa@ke.tu-darmstadt.de, eneldo@ke.tu-darmstadt.de

Abstract. Classifiers chains (CC) is an effective approach in order to exploit label dependencies in multi-label data. However, it has the disadvantages that the chain is chosen at total random or relies on a pre-specified ordering of the labels which is expensive to compute. Moreover, the same ordering is used for every test instance, ignoring the fact that different orderings might be best suited for different test instances. We propose a new approach based on random decision trees (RDT) which can choose the label ordering for each prediction dynamically depending on the respective test instance. RDT are not adapted to a specific learning task, but in contrast allow to define a prediction objective on the fly during test time, thus offering a perfect test bed for directly comparing different prediction schemes. Indeed, we show that dynamically selecting the next label improves over using a static ordering of the labels under an otherwise unchanged RDT model and experimental environment.

Keywords: Multi-label classification, random decision trees, classifier chains

1 Introduction

Contrary to multi-class classification, where only one class label is expected to be associated to an example, multi-label classification (MLC) is the task of assigning a subset of all possible labels to an example. In this task, it is considered crucial to take the dependencies between labels into account. Classifier chains (CC) [18] and their extensions (cf. Section 2) have proven to be a simple but powerful method for exploiting label dependencies in MLC. Similarly to the binary relevance decomposition method these methods train a binary predictor for each of the labels. However, they are organized in a chain so that successive classifiers can make use of the predictions of the previous ones. This enables CC to capture dependencies between labels.

Nevertheless, this simple technique has several shortcomings, especially regarding the chain. Firstly, the ordering in which the labels are predicted in the chain has to be fixed beforehand. To find a sequence which best allows to consider dependencies between labels is a non-trivial task [12] and methods which try to explore different

[⋆] This is the authors' version of the work retrieved from http://www.ke.tu-darmstadt.de. The final publication is available at Springer at http://dx.doi.org/10.1007/978-3-319-67786-6_10 and appeared Larisa Soldatova, Joaquin Vanschoren, George Papadopoulos, Michelangelo Ceci (Eds.): Discovery Science, 21st International Conference, DS 2018, Limassol, Cyprus, October 29–31, 2018, Proceedings. LNAI 11198, pp. 33–50, 2018.

orderings are usually computationally much more expensive than the often taken option of just choosing a random ordering. Secondly, the assumption that there is one single prediction ordering which works best always for every possible single test instance might hold only in very restricted scenarios. Instead, our assumption in this work is that the ordering in which labels should be predicted in order to obtain the best performance highly depends on the specific context, namely the test instance at hand. The question of how to dynamically choose an appropriate ordering for individual instances instead of the entire datasets has been little researched so far. Silva et al. [20] made a first attempt by letting a nearest neighbor classifier decide which ordering to use for a given instance. However, the dynamic selection was restricted to a pre-determined set of static label orderings. The approach of Nam et al. [15] predicts the positive labels at the beginning of the chain, but the ordering in which these are predicted is pre-determined.

In this work, we propose to use random decision trees (RDT) for the purpose of constructing dynamic chains, since these trees have a series of convenient and appealing properties (Section 3).

Foremost, the construction of the model is independent of the specific leaning task. This has the advantage that the objective can easily be changed during prediction without the need for modifying the trees. Our dynamic classifier chains extension of RDT is strongly relying on this property. Instead of choosing the next label to predict from the pre-determined ordering, our proposed method predicts the label for which the RDT is most confident given the current context (Section 4). Our experiments on a series of datasets confirm that it is advantageous to predict the labels in such a dynamic way w.r.t. predictive performance (Section 5).

Moreover, we propose to take advantage of the flexibility of RDT to build a controlled experimental setup where not only the training hyper parameters can be fixed, but also the respective models (Section 4.1). This allows us to directly measure the impact of certain modifications, as well as to compare conceptually different approaches on a fair basis. For instance, we use this possibility in our experimental evaluation to analyze the specific utility of considering previous predictions, or to compare CC to our dynamic CC using the same actual ensemble of trees.

2 Multi-label Classification and Classifier Chains

Multi-label classification is the task of learning a mapping from instances $X \in \mathcal{X}$ to subsets $Y \subset \mathcal{Y}$ of a finite set of non-exclusive class labels $\mathcal{Y} = \{y_0, \ldots, y_n\}$. For convenience, Y is often represented as binary vector $Y = (y_0, \ldots, y_n)$ where y_i is 1 if the label is relevant (positive), otherwise 0 for irrelevant (negative) labels. An extensive overview over MLC is provided by Tsoumakas et al. [22].

The simplest method for solving MLC tasks is the binary relevance method (BR) where each label is handled as a single classification task for which a classifier is trained. Formally, we learn a function $h_i : \mathcal{X} \to \{0, 1\}$ for each y_i. According to this, each classification of a label is independent of the values of the other labels.

Another method is the label power-set method (LP) which reduces the problem of MLC to a single multi-class classification task by representing each possible combination of labels as one separate and exclusive class. This approach naturally considers to predict labels in dependence of the remaining labels, hence focusing on predicting correct label combinations. However, in addition to the obvious limitations due to the exponential growth of label combinations, LP does not allow to predict label combinations which have not been seen in the training data.

A more flexible approach of considering label dependencies was proposed by Read et al. [18] by using classifier chains. CC enhances the idea of BR and executes the binary classifiers in a chain which has the advantage that subsequent classifiers can use the information of the already predicted labels. More formally, each $h_i : \mathcal{X} \times \{0,1\}^{i-1} \rightarrow \{0,1\}$ uses the real labels $y_1, \ldots, y_{i-1}$ for training and the corresponding predictions $\hat{y}_1, \ldots, \hat{y}_{i-1}$ produced by previous classifiers in the chain during testing.

Further analysis revealed that the ordering of the classifiers has an effect on the predictive performance [18, 20]. Usually this ordering is chosen randomly or different random orderings have to be evaluated to find a good chain order. A straight-forward solution is to use ensembles of classifier chains. Nevertheless it turned out that these ensembles are often unnecessarily large for which reason Li and Zhou [11] proposed a method to composite the ensemble. By doing so a subset of CC is selected while keeping or improving the predictive performance of the ensemble.

However, creating and maintaining an ensemble of CC is not always feasible [8]. Another way to handle the label ordering problem is to determine a good chain sequence in advance. For this purpose methods such as genetic algorithms [8], Bayesian networks [21] or double Monte Carlo optimization technique [17] have been used. On the other hand, the classification sequence can be determined during the classification process by finding similar instances in the training set and using the label ordering which works well on these instances [20]. However, this method is not appealing in terms of time complexity since a new CC model has to be build on the fly.

A further improvement of CC could be achieved with probabilistic classifier chains (PCC) [2]. While the training process of both methods is the same, PCC modifies the classification procedure by considering the joint probability of each possible label assignment. According to this, Bayes optimal predictions can be created which makes PCC superior to CC [10]. However, this process has a much higher time complexity and is only feasible for datasets with not more than 15 labels [2]. To tackle this problem beam search [10] or A* search [13] can be used to perform the inferences which speeds up the process. In [14] an overview of inference methods for PCC is given. Nevertheless, PCC also relies on a predefined chain ordering for which reason ensembles of PCC have been introduced [2].

The research on CC and PCC contributed to the understanding and formalization of label dependencies in MLC. For instance, Dembczyński et al. [3] found that these methods are able to exploit so called unconditional dependencies which exist globally on the whole dataset, but also conditional dependencies which only appear locally in the instance space. Moreover, they also discovered that certain multi-label evaluation measures can be orthogonal to each other and optimizing them requires different approaches. For instance, methods such as LP and CC are tailored towards finding the correct label combination, which corresponds to the mode of the joint label distribution, whereas for correctly predicting each label individually (measured by the Hamming loss) it might be sufficient to use approaches such as BR.

3 Random Decision Trees for Multi-label Classification

Introduced by Fan et al. [6], the approach of RDT is an ensemble of randomly created decision trees. More precisely, the tests at the inner nodes are chosen completely at random. This is the major difference compared to classical decision tree algorithms [26], but also to the well known algorithm family of Random Forest [1], where only the subset of features which each tree learner can use is randomly drawn. In contrast, RDT do not optimize any objective function during training, yet they are able to achieve competitive and robust performance [25]. Moreover, by increasing the number of trees in the ensemble the estimation risk can be decreased [25] while we never tend to overfit [4]. Computational complexity is another major advantage because the random selection takes no time compared to computing information gain or similar heuristics [26].

In addition to these guarantees, the characteristics of RDT offers a wide range of possibilities since the random construction is independent of the learning task. For instance, Zhang et al. [26] make use of this property for large scale MLC problems since the computational costs do not depend on the number of labels in their formulation. Zhang et al. [25] propose to abstract RDT with hash functions which is claimed to handle MLC problems in an even more efficient way. RDT were also successfully applied to multi-label stream data and for handling concept drifts with only small modifications to the original algorithm [9]. Depending on the particular needs, RDT can flexibly be constructed before the arrival of the training data [9, 26, 6] or by taking advantage of it [7, 25].

In the following, we describe the general construction and prediction process of RDT as well as the adaptations to be taken for the MLC setting. In particular, we propose an extension to the weighting of the trees in the ensemble based on their individual confidences computed by the Gini-index (Section 3).

Training The construction of the trees for RDT is done recursively, as for most decision tree learners, with the aforementioned difference that the features in the inner nodes are chosen randomly. Hence, starting from the root node, inner nodes are constructed recursively by distributing the training instances according to the test as long as the stopping criterion of maximum depth or minimum number of instances is not fulfilled. Discrete features are chosen without replacement in contrast to continuous features, for which additionally a randomly picked instance determines the threshold [5]. In the case that no further test can be created a leaf will be constructed in which information about the assigned instances will be collected. In MLC, for instance, we might track the number of instances N_k^θ in leaf k of tree θ in relation to the number of positive values $n_k^\theta(i)$ for label y_i. However, any other information could be collected depending on the learning task at hand.

Prediction During prediction, an instance is forwarded from the root to a leaf node passing the respective tests in the inner nodes. In case of missing features, the function $U = q(\theta, X)$ returns the set $U = \{k | k \in [1, T] \subset \mathbb{N}\}$ of the leaves indices in tree θ to which the instance has been assigned to. Following Fan et al. [5], the posterior probability that the specific label y_i is true given an instance X and a tree θ, or an ensemble of trees Θ, respectively, can be formalized as

$$P(y_j = 1|X,\theta) = \frac{\sum_{k\in q(\theta,X)} n_k^\theta(i)}{\sum_{k\in q(\theta,X)} N_k^\theta},\ P(y_i = 1|X,\Theta) = \frac{1}{|\Theta|}\sum_{\theta\in\Theta} P(y_i = 1|X,\theta) \quad (1)$$

An obvious option in order to obtain multi-label predictions from the estimations in Eq. 1 is to use a threshold of 50% so that $\hat{y}_i = I\left[P(y_j = 1|X,\Theta) \geq 0.5\right]$ with $\mathbb{I}[x] = 1$ if x is true and 0 otherwise, which we refer to as the *probability threshold method* (or shortly *probability method*). However, as Quevedo et al. [16] observed, a threshold of 50% is not always ideal. Note that the tests in the tree are not specifically chosen to obtain a high purity of the distributions in the leaves, and in fact many leaves might contribute only with estimates close to the prior distribution, pulling down the average estimates. Thus, Zhang et al. [26] proposed to estimate the average number of relevant labels

$$r(X,\theta) = \frac{\sum_{k\in q(\theta,X)} \sum_{j=1}^{n} n_k^\theta(j)}{\sum_{k\in q(\theta,X)} N_k^\theta}, \quad R(X,\Theta) = \frac{1}{|\Theta|}\sum_{\theta\in\Theta} r(X,\theta) \quad (2)$$

where $R(X,\Theta)$ is rounded in order to get an integer. This value is then used to cut the ranking of labels induced by the distribution of the marginals $P(y_i|X,\Theta)$. We refer to this method as the *label threshold method* or *label method*.

Weighting the trees As aforementioned, the randomness make for a large variety of distributions which are aggregated, many of them approaching the prior label distribution. Nevertheless, previous RDT approaches for MLC use equal weighting irrespectively. We propose to distinguish between the quality of the collected statistics and to reward trees with higher confidences in their estimates. The *Gini index* is often used for determining the purity of a distribution, which we use in inverted form as follows

$$w(X,\theta) = 1 - \frac{4}{n}\sum_{y_i\in\mathcal{Y}} P(y_i = 1|X,\theta)(1 - P(y_i = 1|X,\theta)) \quad (3)$$

in order to weight the estimates of the individual trees, resulting in the overall prediction

$$P(y_i = 1|X,\Theta) = \frac{1}{\sum_{\theta\in\Theta} w(X,\theta)}\sum_{\theta\in\Theta} P(y_i = 1|X,\theta)w(X,\theta) \quad (4)$$

Eq. 2 can be adapted accordingly.

We observed a better performance of using the inverted Gini index in preliminary experiments, so that we adopted it as the default setting for our proposed algorithm.

4 Dynamic Predictions with Random Decision Trees

As already stated, a key disadvantage of classical CC is that their predictive performance may be highly influenced by the pre-selected ordering of the labels. In this section, we propose an extension of RDT referred to as Dynamic Classifier Chains (DCC) where the sequence in which the values for the labels are predicted is chosen dynamically during the process of classification (Section 4.3). Moreover, RDT and their randomized construction provides a very convenient controlled environment for experimentation (Section 4.1).

4.1 Test Bed for Multi-label Classification

In Section 2 we reviewed some transformation methods for solving MLC tasks with different desirable properties, respectively. The description of RDT in Section 3 applies to binary classification problems as well. For instance, we could use RDT as base learner for a binary relevance decomposition, estimating $P(y_i|X, \Theta_i)$ instead of $P(y_i|X, \Theta)$. However, both ensembles Θ_i and Θ are drawn from the same tree distribution which is independently of any y_i. Hence, both estimations approach the same expected value as the number of constructed trees increases.

This key observation lead to the following advantages of RDT. Firstly, we can collapse BR, and other MLC transformation or decomposition methods [22] such as CC as we will see in the following, to a single RDT ensemble without loss in predictive accuracy, therefore saving memory and computational costs. Secondly, and more importantly, RDT can provide a controlled environment where we can compare alternative decomposition methods, prediction methods and other extensions isolated from any side effects since the model can be fixed beforehand and be the same for every analyzed approach.

4.2 Static Chain Ordering

Similarly to BR, we can collapse a classifier chain to a single RDT in the following way: Instead of augmenting the input space $\mathcal{X}$ by only the previous labels, we add the whole label matrix so that $X' \in \mathcal{X} \times \mathcal{Y}$. The prediction of base classifier h_i for label y_i (more specifically, $P(y_i = 1|X, \hat{y}_1, \ldots, \hat{y}_{i-1}, \Theta_i)$) is obtained by setting $\hat{y}_j$ to the previous predictions of h_j, $j < i$, as for CC, but leaving $\hat{y}_j$, $j \geq i$ as *missing*.[1] Remind that when encountering a missing value at inner nodes all branches are visited and aggregated (cf. Section 3). As RDT are completely randomized, we can —similarly to Section 4.1— expect on average the same predictions as for a RDT with one node less. In fact, we control in our experiments the percentage of activated label tests with a parameter σ, which allows us to analyze the effect of using previous predictions on an otherwise unchanged model.

Figure 1 visualizes the prediction process for a label on a single tree: Let us assume that the label to be predicted is y_i, which comes before y_j. In this case neither y_i nor y_j are known, i.e. all three colored branches are followed and the respective leaves are used in order to produce a prediction for y_i. For label y_j the previous label y_i would be known, so that we would skip either the left or right branch, obtaining a label distribution at the leaves which is different and more refined than the previous one. Indeed, we can observe that the number of leaves on which the prediction relies is monotonically decreasing during the classification process. Therefore, the set of leaves

[1] We assume, w.l.o.g., that $y_1, y_2, \ldots$ is the ordering of the predicted labels.

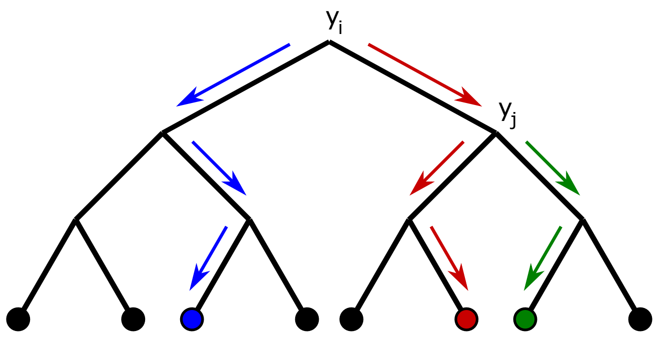


Fig. 1: Example for the refinement of a prediction for a particular instance and decision tree. y_i and y_j indicate tests on labels at the respective inner nodes.

to which the instance is assigned in the first iteration will always be a superset of the leaves of the following iterations. This leads to a refinement of the prediction through the iterations.

4.3 Dynamic Chain Ordering

In order to take advantage of the situation that predicting a label before or after another one might be easier depending on the instance at hand, we propose to let the RDT decide which label to predict next. Hence, instead of using the estimated distribution in order to decide whether the i-th label is positive or negative, we use it in order to set the label for which RDT is most confident in its prediction. Labels, which were already predicted, are ignored.

Predicting the next label in the sequence For convenience, we introduce the following definitions. Let C denote the label candidates which were not yet predicted, P^+ the set of labels which were predicted as relevant, and P^- the irrelevant labels, respectively. Accordingly, we start with $P^+ = P^- = \emptyset$, $C = \mathcal{Y}$ in the first iteration.

In each iteration, we first decide on the next label to be predicted. We select the label for which the RDT is most confident in the following way and remove it from C:

$$y_i = \arg\max_{y_j \in C} |0.5 - P(y_j = 1|X, P^+, P^-, \Theta)| \tag{5}$$

In preliminary experiments we found that this approach works consistently better than always choosing the label with the lowest or the highest probability, respectively.

With y_i chosen, we can use the *probability method* (cf. Section 3) to determine whether to add it to P^+ or P^-. We note that we rely on a threshold of 50% to classify a label as positive although this may be suboptimal regarding the skewed distribution of the label sizes (as already pointed out in Section 3). However, preliminary experiments with varying thresholds, e.g. by adapting them to the prior distribution, revealed that choosing the optimal thresholds is non-trivial. One particular reason is that the

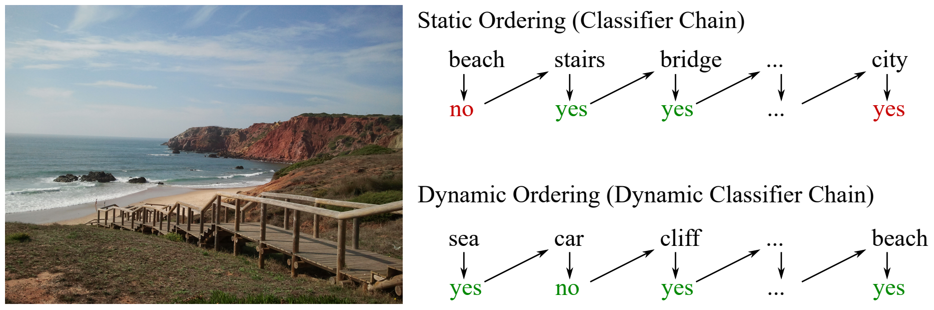

Static Ordering (Classifier Chain)
beach
no
stairs
yes
bridge
yes
...
...
city
yes
Dynamic Ordering (Dynamic Classifier Chain)
sea
yes
car
no
cliff
yes
...
...
beach
yes

Fig. 2: Example for different classifications using the static and the dynamic ordering for a picture associated to labels *beach, sea, cliff, bridge, stairs*. See text for explanations.

thresholding of a specific label is required at different stages of the prediction chain, in contrast to using a static ordering, introducing additional dependencies and dynamics of the right threshold. We leave further investigations for future work.

The process of predicting the value with the *label method* also needs further adaption due to the iterative prediction of the labels. The idea is to have predicted exactly $R(X, P^+, P^-, \Theta)$ labels positive after the prediction sequence is completed. Since the prediction changes during the classification process $R(X, P^+, P^-, \Theta)$ has to be recomputed in every iteration. First of all, we can only predict a label positive if the number of already predicted positive labels $|P^+|$ is smaller than $R(X, P^+, P^-, \Theta)$. Moreover, we have to predict a label as positive if we know that all the remaining labels in C need to be predicted positive to ensure that we obtain exactly $R(X, P^+, P^-, \Theta)$ positive labels.

$$y_i = \begin{cases} 1, & \text{if } P(y_i = 1 | X, P^+, P^-, \Theta) \geq 0.5 \text{ and } |P^+| < R(X, P^+, P^-, \Theta) \\ 1, & \text{if } n - |P^-| < R(X, P^+, P^-, \Theta) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Let us consider again the tree in Figure 1. The difference to the static chain approach is that the aggregated blue, red and green leaves would be used in order to determine whatever label y_k is most likely given the found distribution, instead of a specific label, in the previous example label y_i. Hence, the RDT could decide to predict y_j instead if they are more confident about it, or any other label with the highest confidence. We believe that this potentially leads to more reliable predictions, both in terms of individual labels as well as label combinations.

In particular Figure 2 displays an example how the quality of the predictions could differ between a static ordering and a dynamic ordering. Let us assume we want to identify objects in a scene. While the static ordering has to follow a pre-defined sequence for the classification, it has to classify the label *beach* first. Since the beach is not clearly visible on the picture, the label receives a negative value which already introduces an error for future predictions in the chain. Especially for our example, this has the consequence that the scene is classified as a city because stairs and bridges are more correlated with cities than with the seaside. In contrast, the dynamic chain classifies the most obvious targets first for which reason the label *sea* receives a positive classification in the first iteration. This increases the chance to classify the label cliff as positive which provides even more evidence for predicting the difficult label *beach* as positive. On other hand, we reduce the probability to incorrectly classify the scene as a city, since we exclude objects which are clearly not identifiable in the picture, such as a car, in early iterations.

Computational Costs The costs for building the trees and performing the dynamic predictions is conceptually equal to using a static ordering. They mainly depend on the size of the ensemble and the depth of the trees. However, the dynamic approach potentially allows to shorten the prediction process, namely when enough positive (or negative) labels have been already predicted, potentially removing the dependencies on the label size.

5 Evaluation

A key aspect in our experimental evaluation was, of course, to demonstrate that using dynamic, context-dependent predictions improves over using static orderings w.r.t. predictive performance (Section 5.3). A decisive role in this is played by the influence of the previous predictions on the current prediction, which is analyzed in Section 5.2.

Another aspect, we were particularly interested in, was to verify our ideas on the usage of RDT as controlled experimental environment for fair and specific comparisons.

Regarding our proposed dynamic approach, we will mainly distinguish between the two variants using the probability and the label threshold method for determining the value of the next label, respectively. We expected that other hyper parameters would behave quite different with respect to different datasets, both regarding the shapes (and densities) of the input and output spaces. In contrast to other hyper parameters like number of trees or minimum leaf sizes, we decided to consider this aspect separately.

5.1 Setup

For our experiments we have used eight different multi-label datasets from the Mulan repository [23]. An overview of these datasets is provided in Table 1. From the text datasets we have only included *Enron* and *Medical*, which have a relatively small vocabulary, since RDT are known to not perform well on sparse data without further adaptations which we did not want to put in the focus for this work.

A large variety of evaluation measures exist for MLC. We focus in this work on two of them, namely *subset accuracy* and *micro-averaged F1 measure*. Subset accuracy is a very restrictive evaluation metric since it only measures the percentage of instances for which all labels have been predicted correctly. Especially in the case of predicting a large amount of labels this measure often approaches zero without being able to distinguish. However, the objective of classifier chains is precisely to find exactly the correct label combination (cf. Section 2). Hence, we expect the impact of our proposed extensions to be best reflected in the subset accuracy.

Table 1: Dataset statistics: Total number of instances, of nominal and numeric attributes, of labels, average number of labels per instance and distinct label combinations.

name	**instances**	**nominal**	**numeric**	**labels**	**cardinality**	**distinct**
Flags	194	9	10	7	3.392	54
Emotions	593	0	72	6	1.869	27
Scene	2407	0	294	6	1.074	15
Yeast	2417	0	103	14	4.237	198
Birds	645	2	258	19	1.014	133
Medical	978	1449	0	45	1.245	94
Enron	1702	1001	0	53	3.378	753
CAL500	502	0	68	174	26.044	502

Micro-averaged F1 measure is less strict since it also considers partial matches and is therefore often used for providing a general comparison of the predictive quality. However, the measure is to a certain degree orthogonal to subset accuracy [3]. As Dembczyński et al. [3] indicate, it is sufficient to obtain good estimates for the individual labels in order to optimize univariate losses such as F-measure or Hamming loss. Nevertheless, our approach may still benefit from the dependencies captured by the chaining approach with respect to these measure, which is why we include micro-averaged F1 measure (micro F1) in our comparisons.

Given N test instances, corresponding true labels Y_j and predicted labels $\hat{Y}_j$, true positives $tp_j = Y_j \cap \hat{Y}_j$, false positives $fp_j = \hat{Y}_j \setminus Y_j$, false negatives $fn_j = Y_j \setminus \hat{Y}_j$ for the j-th test instance, we obtain the measures as follows:

$$\text{subset accuracy} = \frac{1}{N}\sum_{j=1}^{N} \mathbb{I}\left[Y_j = \hat{Y}_j\right] \qquad \text{micro F1} = \frac{\sum_{j=1}^{N} 2\,tp_j}{\sum_{j=1}^{N} 2\,tp_j + fp_j + fn_j} \tag{7}$$

Unless otherwise noted we have chosen to evaluate all combinations of parameter settings of ensembles with 300 decision trees, a maximum depth of 30, a minimum number of instances to create a test of $\{4, 6, 10\}$ and a percentage of label tests of $\{10\%, 20\%, 30\%\}$. Preliminary experiments with RDT revealed reasonable and stable performance for these parameter ranges also on other kind of problems. Furthermore, we compare the results based on the averages of a ten-fold cross validation performed on the whole dataset.

5.2 Independent Predictions vs. Exploiting Previous Predictions

In this experiment we evaluated how the prediction is influenced by the usage of the label tests, i.e., by the usage of the previous predictions in the dynamic chain. At this stage the flexibility of the RDT algorithm pays off since we can choose the ratio σ of activated tests on the labels without the need for adaptations of the model (cf. Section 4.2). Hence, $\sigma = 0$ corresponds to a binary relevance classifier using RDT (or the collapsed version, respectively). Incrementing σ allows to directly observe utility and the effectiveness of exploiting potential label dependencies.

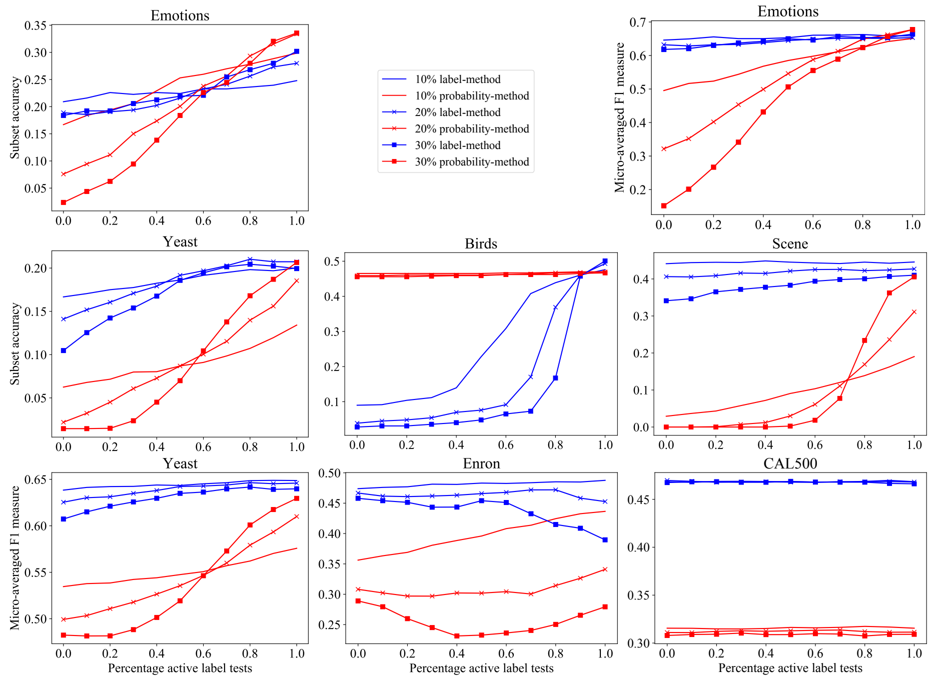


Fig. 3: Influence of label tests on DCC. The y-axis represents the value for the measure and the x-axis represents the percentage σ of activated label tests. The color indicates the prediction method and the style of the line represents the percentage of label tests per tree.

Figure 3 show the benefit for some selected cases w.r.t. subset accuracy but also for improving the univariate micro F1. For instance, we can observe on datasets *Emotions* and *Yeast* a major influence of the label tests on the performance for both prediction and evaluation methods. We can conclude that there is a strong dependency between the labels in the datasets of which we can take advantage. Similar but less pronounced effects can be seen for the remaining datasets except *Enron* and *CAL500*. *Enron* shows that the usage of (possibly wrong) previous predictions can also have a negative impact in some cases, or no impact as for *CAL500*. Moreover, both datasets are also an example for the observation that the best label prediction method is highly dataset dependent.

In general it can be seen that the values for the evaluation measures get better the more label tests are activated. Only on the dataset *CAL500* the label tests seem not to have any influence on the predictive performance. Moreover, on the dataset *Enron* it can be seen that the activation of the label tests have a negative impact on the micro-averaged F1 measure independently of the prediction method in the case that 30% of label tests have been used in the trees. Furthermore, on the dataset *Birds* only the label method benefits from the label tests whereas on the dataset *Scene* only the probability method benefits from the label tests. The values for the corresponding other prediction method stay the same along the activation of the tests.

Table 2: Comparison between dynamic and static chain method for subset accuracy. Bold entries indicate the best results.

	Flags	Emotions	Scene	Yeast	Birds	CAL500	Enron	Medical
DCC LM	**0.1959**	0.2799	**0.4271**	**0.2073**	**0.4930**	0.0000	0.0447	**0.1840**
CC LM	0.1623	0.1098	0.1714	0.0215	0.3673	0.0000	0.0116	0.0034
DCC PM	0.1856	**0.3339**	0.3112	0.1854	0.4698	0.0000	**0.0576**	0.0000
CC PM	0.1835	0.1482	0.0914	0.0407	0.4583	0.0000	0.0286	0.0000

Table 3: Comparison between dynamic and static chain method for micro F1.

	Flags	Emotions	Scene	Yeast	Birds	CAL500	Enron	Medical
DCC LM	**0.7506**	0.6535	**0.4682**	**0.6459**	**0.4484**	**0.4681**	**0.4527**	**0.2696**
CC LM	0.7301	0.4439	0.1935	0.4893	0.1288	0.2945	0.2669	0.0043
DCC PM	0.7448	**0.6774**	0.4344	0.6100	0.0708	0.3114	0.3410	0.0065
CC PM	0.7484	0.4490	0.1656	0.5204	0.0137	0.3093	0.3153	0.0000

5.3 Static vs. Dynamic Label Orderings

In this experiment we evaluated the advantage of the dynamic chain ordering in comparison to using a static chain ordering. Taking advantage of our controlled environment, we built for both approaches the same ensemble of trees, respectively, in this case with 20% of label tests. The only difference between the dynamic and the static setup is the ordering of the labels during the prediction process. We compare our proposed dynamic method to the averages over ten different randomly-drawn but fixed orderings used for the static CC approach in Tables 2 and 3.

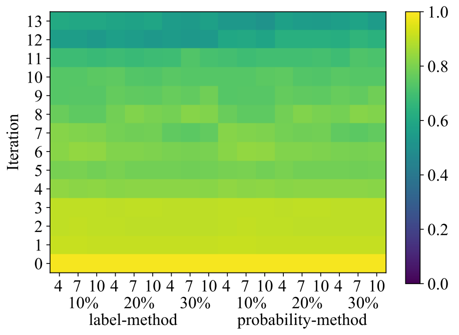
The first and foremost observation is that the dynamic chain ordering is clearly superior to the static chain ordering on all datasets. Moreover, the dynamic chain ordering improved the evaluation results of the label method (LM) on all datasets often to a great extent whereas the probability method (PM) does not take major advantage of the dynamic chain ordering on the datasets *Birds*, *CAL500* and *Medical*.

The results suggest that the improvement of DCC over CC relies on high confidence of the classifications in the first iterations. These classifications provide evidence to improve the classification of the difficult labels. This effect is analyzed in more detail in the following experiment.

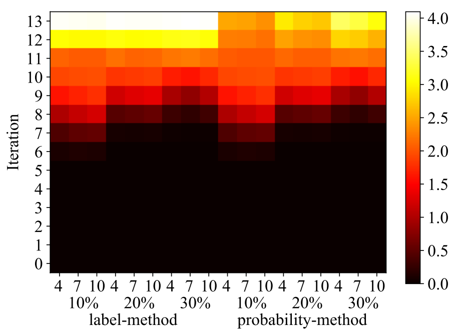
5.4 Analysis of the Dynamic Sequences

Our approach dynamically produces a different prediction sequence on the labels for each given test instance. We were interested in characterizing and analyzing these sequences, which were selected by the RDT as being most appropriate for producing accurate predictions.

Figure 4 visualizes our results exemplarily for *Yeast*. The heat map on the left shows the average accuracy (color) of predicting the i-th label in the dynamic sequence (y-



(a) Accuracy of the prediction



(b) Number of positive labels

Fig. 4: Heatmaps characterizing the predicted sequences on *Yeast*

axis) for the different configurations (x-axis), whereas the right map visualizes the number of labels (color) which were predicted as positive until a certain iteration.

We can observe on *Yeast* as well as on the remaining datasets and independent of the parameter configuration that the predictions of the first iterations are pretty accurate in comparison to the error-prone predictions at the end. One reason is of course that our label selection method chooses the labels where the RDT ensemble is most confident first. On the other hand, this can be the result of the error propagation since erroneously predicted labels influence the predictions of the following iterations. Furthermore, it can be observed that errors are mostly made on positive labels after predicting almost all the negative labels. This can be again explained by the confidences, which is higher for negative labels due to the sparsity of the label assignments.

5.5 Comparison with other Classifiers

In order to put the performance of RDT and the different prediction methods in a larger context we present in this section a comparison to a couple of other algorithms. We have evaluated the BR, the LP and the CC method with the J48 WEKA implementation of the C4.5 decision tree learner as the base classifier. This approach represents in our comparison the family of classical decision tree learners which address a learning task by choosing splits at the inner nodes which optimize a certain pre-determined criterion such as the information gain. For the J48 algorithm we have used the default parameter settings which are 0.25 for the confidence threshold for pruning and 2 for the minimum number of instances per leaf. For the CC method we have evaluated ten different random chain orderings and averaged the results. Moreover, we have evaluated the BR and the LP method using RDT. For these methods we have chosen to build ensembles with 300 decision trees, with a maximum depth of 30 and a minimum of four instances to create a test. The dynamic chain methods share the same settings and use 20% of label tests in their ensembles. Hence, the dynamic chain methods use 20% less tests on the features compared to the BR and the LP method because they are replaced by label tests. The ten-fold cross validation results can be seen in Tables 4 and 5.

Table 4: Results for the subset accuracy

	Flags	Emotions	Scene	Yeast	Birds	CAL500	Enron	Medical
DCC LM	0.1959	0.2799	0.4271	0.2073	0.4930	0.0000	0.0447	0.1840
DCC PM	0.1856	0.3339	0.3112	0.1854	0.4698	0.0000	0.0576	0.0000
RDT LP	0.1959	**0.3929**	0.4612	**0.2416**	**0.5008**	0.0000	**0.1363**	0.2566
RDT BR	0.1701	0.2479	0.1379	0.0852	0.4698	0.0000	0.0129	0.0000
J48 BR	0.1443	0.1686	0.3515	0.0633	0.4930	0.0000	0.0593	0.6708
J48 CC	0.2211	0.2169	0.4548	0.1327	0.4998	0.0000	0.0977	**0.6906**
J48 LP	**0.2474**	0.1939	**0.4902**	0.1419	0.4729	0.0000	0.0823	0.6585

Table 5: Results for the micro-averaged F1 measure

	Flags	Emotions	Scene	Yeast	Birds	CAL500	Enron	Medical
DCC LM	0.7506	0.6535	0.4682	**0.6459**	0.4484	**0.4681**	0.4527	0.2696
DCC PM	0.7448	0.6774	0.4344	0.6100	0.0708	0.3114	0.3410	0.0065
RDT LP	0.7310	**0.7168**	0.4953	0.6353	0.3390	0.3349	0.3652	0.3520
RDT BR	**0.7545**	0.5997	0.2433	0.5598	0.1063	0.3178	0.3842	0.0000
J48 BR	0.7416	0.5791	**0.5563**	0.5774	**0.4675**	0.3553	**0.5096**	0.8198
J48 CC	0.7248	0.5806	0.5420	0.5516	0.4576	0.3501	0.4996	**0.8225**
J48 LP	0.7045	0.5668	0.5374	0.5397	0.4259	0.3309	0.3818	0.7527

First of all, it can be seen that the label method outperforms the J48 methods on the datasets *CAL500*, *Emotions* and *Yeast* in terms of subset accuracy and micro-averaged F1 measure. On the other hand, the J48 methods were able to beat the dynamic chain methods on the datasets *Scene*, *Enron* and *Medical*. Especially the results on the dataset *Medical* are conspicuous, where RDT generally performs very poorly. As aforementioned, sparse input data is particularly challenging for RDT-like approaches.

Furthermore, it can be seen that the dynamic chain methods are superior to the RDT-BR method on almost all datasets, as anticipated by the results in Section 5.2.

Of particular interest is the comparison to the RDT-LP method. In terms of subset accuracy this method could outperform the dynamic chain methods on almost all datasets. Especially the results on the datasets *Emotions*, *Yeast* and *Enron* are much better than the results of the other methods. However, a closer examination reveals that the results for the micro-averaged F1 measure of the RDT-LP method are not always that good. The label method could achieve a much higher score for micro-averaged F1 measure on the datasets *Birds*, *CAL500* and *Enron*. Senge et al. [19] observed that LP can benefit from the restricted set of label combinations it can choose from, especially when the number of distinct combinations is relatively low, as it is the case for the used datasets. The other approaches, instead, have to make up valid combinations by concatenating single decisions. Whereas these single decisions might be better than for LP, as seen in terms of micro-averaged F1 measure, the complete combination might still be wrong especially if the cardinality is high.

6 Conclusions and Future Work

In this paper we have proposed a new approach on multi-label classification based on random decision trees and the idea of classifier chains. With our proposed algorithm we have been able to overcome the major problem of the label ordering by dynamically selecting the next label in the sequence depending on the context, namely the instance at hand and the previously predicted labels for it. In comparison to other approaches for CC, which try to pre-compute appropriate sequences, our approach comes at no additional cost, since the framework of RDT allows to perform the necessary inferences completely during prediction time.

In several experiments the dynamic label ordering has been analyzed in depth and compared with other baseline methods. Even though we cannot achieve state-of-the-art results with RDT in some cases and domains, they have appealing properties that allow a fundamental analysis of the advantages and disadvantages of certain approaches. For instance, our experiments revealed the importance of the dynamic label ordering on different datasets, as well as the impact of using the previous predictions. These observations could be made with the guarantee that they were independent of any other factors like the usage of more sophisticated methods or more powerful models.

However, to improve the predictive capabilities of RDT still remains a goal for future work. For instance, the proposed Gini index considers the skew of the counts, but not the number of instances these counts are based on, which could be used as further indicator for the confidence. Efficiency could also be improved if we consider that labels are usually sparse in MLC problems. Therefore, it could be enough to focus on positive labels only, which would considerably reduce the length of the prediction sequence. In addition, as we have seen, RDT have still clear disadvantages on data which is sparse in the feature values, such as text. New types of tests in the inner nodes, which for instance consider disjunctions of several features, could solve this problem. Furthermore, we plan to transfer our ideas on dynamic chains to other kinds of algorithms as well. A first step will be to adapt predictive clustering trees [24]. The construction of these trees does also not necessarily depend on a specific target. However, its clustering may allow for more discriminative distributions at the leaves.

References

[1] Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)

[2] Dembczyński, K., Cheng, W., Hüllermeier, E.: Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. pp. 279–286 (2010)

[3] Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E.: On label dependence and loss minimization in multi-label classification. Machine Learning 88(1-2), 5–45 (2012)

[4] Fan, W.: On the Optimality of Probability Estimation by Random Decision Trees. In: Proceedings of the 19th National Conference on Artificial Intelligence. pp. 336–341 (2004)

[5] Fan, W., Greengrass, E., McCloskey, J., Yu, P.S., Drammey, K.: Effective Estimation of Posterior Probabilities: Explaining the Accuracy of Randomized Decision Tree Approaches. In: Proceedings of the 5th International Conference on Data Mining. pp. 154–161 (2005)

[6] Fan, W., Wang, H., Yu, P.S., Ma, S.: Is random model better? On its accuracy and efficiency. In: Proceedings of the 3rd IEEE International Conference on Data Mining. pp. 51–58 (2003)

[7] Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 63(1), 3–42 (April 2006)

[8] Goncalves, E.C., Plastino, A., Freitas, A.A.: A Genetic Algorithm for Optimizing the Label Ordering in Multi-label Classifier Chains. In: Proceedings of the IEEE 25th International Conference on Tools with Artificial Intelligence. pp. 469–476 (2013)

[9] Kong, X., Yu, P.S.: An Ensemble-based Approach to Fast Classification of Multi-label Data Streams. In: Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing. pp. 95–104 (October 2011)

[10] Kumar, A., Vembu, S., Menon, A.K., Elkan, C.: Beam search algorithms for multilabel learning. Machine Learning 92(1), 65–89 (2013)

[11] Li, N., Zhou, Z.: Selective Ensemble of Classifier Chains. In: Multiple Classifier Systems: 11th International Workshop on Multiple Classifier Systems, pp. 146–156 (2013)

[12] Malerba, D., Semeraro, G., Esposito, F.: A multistrategy approach to learning multiple dependent concepts. In: Machine Learning and Statistics: The Interface, chap. 4, pp. 87–106 (1997)

[13] Mena, D., Montañés, E., Quevedo, J.R., Coz, J.J.d.: Using A* for Inference in Probabilistic Classifier Chains. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 3707–3713 (2015)

[14] Mena, D., Montañés, E., Quevedo, J.R., Coz, J.J.d.: An Overview of Inference Methods in Probabilistic Classifier Chains for Multilabel Classification. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 6(6), 215–230 (2016)

[15] Nam, J., Loza Mencía, E., Kim, H.J., Fürnkranz, J.: Maximizing subset accuracy with recurrent neural networks in multi-label classification. In: Advances in Neural Information Processing Systems 30 (NIPS-17). pp. 5419–5429 (2017)

[16] Quevedo, J.R., Luaces, O., Bahamonde, A.: Multilabel Classifiers with a Probabilistic Thresholding Strategy. Pattern Recognition 45(2), 876–883 (2012)

[17] Read, J., Martino, L., Luengo, D.: Efficient Monte Carlo methods for multi-dimensional learning with classifier chains. Pattern Recognition 47(3), 1535 – 1546 (2014)

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

[19] Senge, R., Del Coz, J.J., Hüllermeier, E.: On the problem of error propagation in classifier chains for multi-label classification. In: Data Analysis, Machine Learning and Knowledge Discovery, pp. 163–170 (2014)

[20] Silva, P.N.d., Gonçalves, E.C., Plastino, A., Freitas, A.A.: Distinct Chains for Different Instances: An Effective Strategy for Multi-label Classifier Chains, pp. 453–468 (2014)

[21] Sucar, L.E., Bielza, C., Morales, E.F., Hernandez-Leal, P., Zaragoza, J.H., Larrañaga, P.: Multi-label Classification with Bayesian Network-based Chain Classifiers. Pattern Recognition Letters 41, 14–22 (2014)

[22] Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data. In: Data Mining and Knowledge Discovery Handbook, pp. 667–685 (2010)

[23] Tsoumakas, G., Spyromitros-Xioufis, E., Vilcek, J., Vlahavas, I.: MULAN: A Java Library for Multi-label Learning. Journal of Machine Learning Research 12, 2411–2414 (July 2011)

[24] Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185 (2008)

[25] Zhang, X., Fan, W., Du, N.: Random Decision Hashing for Massive Data Learning. In: Proceedings of the 4th International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications. pp. 65–80 (2015)

[26] Zhang, X., Yuan, Q., Zhao, S., Fan, W., Zheng, W., Wang, Z.: Multi-label Classification without the Multi-label Cost. In: Proceedings of the Society for Industrial and Applied Mathematics International Conference on Data Mining. pp. 778–789 (2010)