Label Ranking through Multi-Label Classification

Label Ranking durch Multi-Label Klassifikation
Bachelor-Thesis von Constantin Stipnieks aus Bad Soden am Taunus
Tag der Einreichung:

1. Gutachten: Johannes Fürnkranz
2. Gutachten: Eneldo Loza Mencía

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Knowledge Engineering Group

Label Ranking through Multi-Label Classification
Label Ranking durch Multi-Label Klassifikation

Vorgelegte Bachelor-Thesis von Constantin Stipnieks aus Bad Soden am Taunus

1. Gutachten: Johannes Fürnkranz
2. Gutachten: Eneldo Loza Mencía

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den August 27, 2018

(C. Stipnieks)

1 Abstract

Label ranking aims to learn a model that finds a ranking over a fixed set of labels L for a given input x. This is usually achieved through specialized algorithms that produce a total order over L. In that case the algorithm will often be forced to make uncertain decisions that lead to incorrect rankings. Another approach is to give the algorithm the freedom to abstain from making a decision on a particular ranking over two labels, resulting in partial orders as the output. This can lead to fewer wrong decisions at the cost of less complete rankings. We propose to rephrase the problem of label ranking to multilabel classification, by modeling every pairwise preference between two labels as a label in itself. The multilabel classification model then has no constraints on what kind of order to output, meaning that abstentions become possible. As part of this bachelor thesis, we test this approach and provide multiple experimental results.

Contents

List of Tables

List of Figures

2 Introduction

Preference learning and label ranking in particular are relatively young topics in artificial intelligence research. The prediction of a label ranker for a given instance x consists of a ranking over a finite set of labels L. Many problems in machine learning can be seen as specific label ranking tasks, e.g. conventional classification is a label ranking setting in which only the top ranking label is the output. Additionally, it has been shown that multilabel classification is also a scenario that can be generalized to a label ranking problem [3]. In this thesis we are showing that the inverse is also possible, i.e. label ranking can be rephrased to be a multilabel classification problem. To do this, we transform the label ranking dataset into multilabel data by modeling every pairwise preference between two labels as a label in itself. Various multilabel classifiers are then trained and tested on the data to allow for a broad evaluation of the approach.

The ranking that label rankers output is generally of total order, meaning that for every two labels, a decision on the pairwise preference between them is made. While this has the advantage that we extract the maximally possible information for each instance, it naturally leads to more incorrect decisions being made along the way. A more ideal scenario is one in which the ranker recognizes that the preference relation between two labels a and b is highly uncertain. Instead of ranking a over b or b over a, it can now abstain from making a decision. This means that the output may now also be a partial order, with the hope that said partial order contains fewer incorrect rankings. This approach has been proposed by Cheng et al. [1]. It has been shown that a trade-off between the *completeness*, i.e. the degree of abstentions being made, and the *correctness*, the share of correct decisions in the ranking, can be achieved with previously introduced methods [1] [5]. We will show that through our reformulation of the label ranking problem, abstentions may naturally arise and that this often results in improved correctness in comparison to various baselines.

The thesis will be structured as follows. In the next section, we will introduce previous work regarding partial rankings in label ranking. After that, we define conventional classification, multilabel classification and label ranking, along with the specific learning algorithms that we use in this thesis. In sections 7 and 8, we explain our proposed method in detail and cover the previously hinted at evaluation measures of *correctness* and *completeness*. Section 9 introduces the utilized datasets, before we conduct multiple experiments and evaluate the performance of our rankers. The thesis then ends with a conclusion.

3 Previous Work

Cheng et al. first introduced the idea of partial orders in label ranking in 2010 [1]. The principal idea described in their paper is that we can build a label ranker with the option for abstentions from any conventional label ranker. Formally, we try to find a partial order $\sqsupset$, where $l_i \sqsupset l_j$ indicates that label l_i precedes l_j. That is done through the technique of ensembling. Let L be the given label ranking algorithm. Instead of training only one model, we train k such models by re-sampling from the given dataset each time. Given an instance x, we then let all these models make a prediction on the associated ranking. For each pair of labels l_i l_j, we use the share of models that predicted $l_i \succ l_j$ as a degree of certainty $P(l_i, l_j)$ for $l_i \sqsupset l_j$. Clearly, if all models agree that l_i precedes l_j the degree is 1. In a maximally uncertain situation, the degree would be 0.5.

To derive a ranking from these degrees, a threshold $0 \leq \alpha \leq 1$ is used, such that $l_i \sqsupset l_j$ only holds when $P(l_i, l_j) > \alpha$. For uncertain decisions, where both $P(l_i, l_j)$ and $P(l_j, l_i)$ are smaller than the threshold, an abstention will subsequently be made. A lower threshold may lead to cyclic predictions, while a larger one produces more abstentions. In the paper a method is introduced that finds the minimal α, such that the ranking produced after the cut-off is still acyclic.

In 2012, Cheng et al. [5] introduced a simpler method to deduct acyclic partial rankings. It only works on label rankers that output probability distributions over the set of all possible rankings. Given the distribution $\mathbf{P}$, the probability for a pairwise preference $l_i \succ l_j$ is formalized as

$$P(l_i, l_j) = \sum_{\pi \in E(l_i, l_j)} \mathbf{P}(\pi) \tag{1}$$

where $E(l_i, l_j)$ denotes the set of all rankings in which l_i precedes l_j . By thresholding this equation, i.e. only including pairwise rankings where $P(l_i, l_j) > \alpha$, a partial order that is already acyclic and transitive can be derived. This is true for all $\alpha \in [1/2, 1]$ and by varying the threshold, a trade-off between the error rate and the number of abstentions can be achieved.

4 Conventional Classification

In the conventional classification scenario, we try to learn a mapping from an instance space $\mathcal{X}$ to a set of predefined classes, commonly represented by labels $\mathcal{L} := \{l_1, ..., l_m\}$. Instances are defined by a set number of attributes, meaning that each instance x can be thought of as a feature vector $(a_1, ..., a_d)$ containing d attributes. The model is then defined as a function

$$\mathcal{X} \to \mathcal{L}$$
$$x \mapsto l$$

that maps an instance to a single label. In the case where $\mathcal{L}$ only consists of two labels, we call the problem *binary classification*. When a non-binary decision has to be made, i.e. $\mathcal{L}$ contains three or more labels, the problem is generally referred to as *multiclass classification*. In either case, only one label is assigned to each instance.

Classification falls into the supervised learning domain, meaning that the model learns from given *training* data $\mathcal{T} := \{(x_i, l_i)\}, i = 1, 2, ..., n$, where each instance x_i is already associated with its corresponding class label l_i. Depending on the classifier, training processes can vary a lot.

To see how well the model works after it learned from the training data, it is subsequently evaluated on *test* data. This is done by taking the predictions of the model for an instance in the test set and comparing it with the previously known *ground truth* of the associated label. It is crucial that the training and test dataset are kept apart, as evaluating the model on the same data that it was trained on would produce significantly biased results. That is because it already learned to classify the instances in the training set. Since the model should ultimately be able to predict new instances, we also need to test it on unseen ones.

In this thesis, two binary classification algorithms will be employed as base classifiers of our utilized multilabel classifiers, namely random forest and logistic regression.

4.1 Random Forest

Random forest is a learning algorithm that utilizes multiple decision trees [9]. An arbitrary example of such a decision tree is given in figure 1.

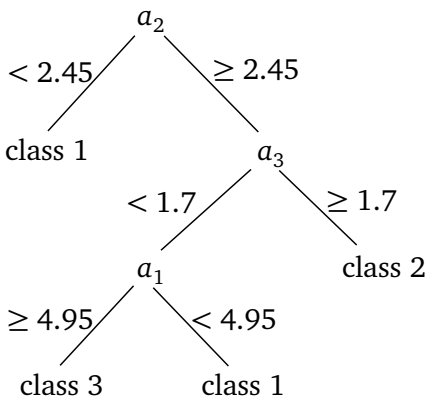


Figure 1: Example of a decision tree

Given a sample, the algorithm traverses the tree until it arrives at a leaf, which marks the predicted class. Each inner node represents a test on a certain attribute and the branches are the different outcomes. During the learning phase, the algorithm will learn which attributes have the highest impact on the class and those will appear higher up in the tree. Decision trees have the advantage of being a highly interpretable white-box. By examining the tree, we can retrace how it came to a certain decision and we can also understand which attributes affect its decisions the most. This is something that black-box models, like neural networks, do not allow as easily.

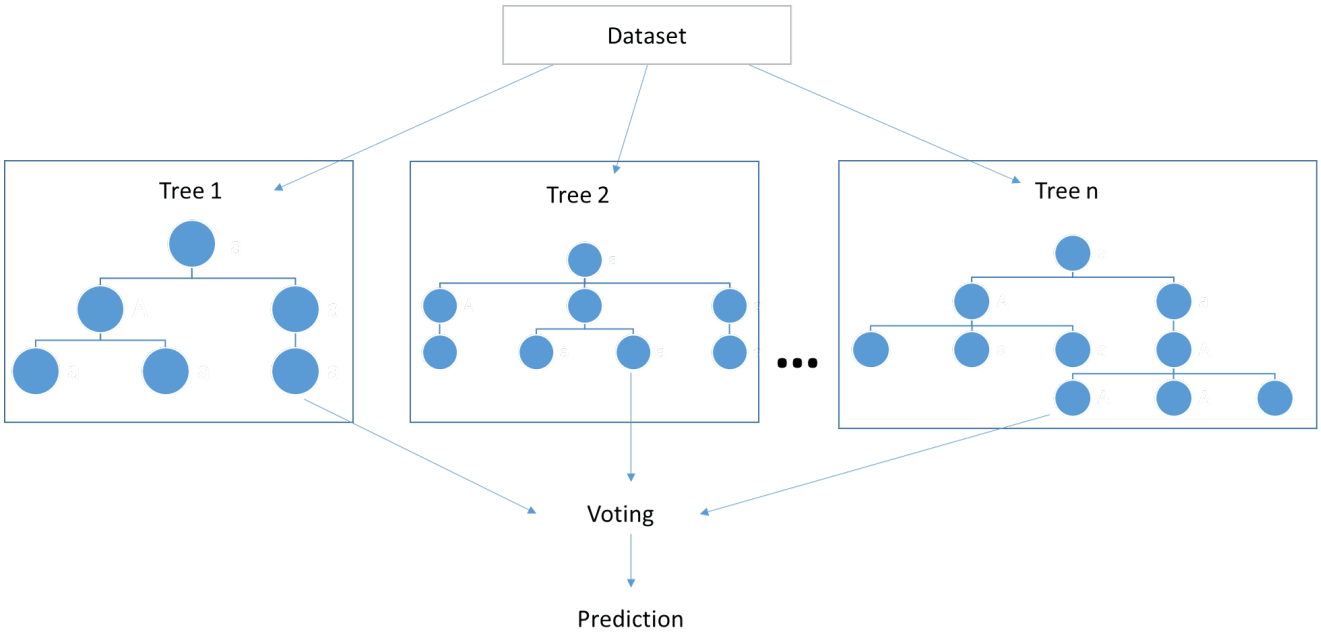


Figure 2: Random forest visualization

Source: https://analyticsdefined.com/introduction-random-forests/

However, a single decision tree has the disadvantage of generalizing poorly from the training data, i.e. overfitting. Random forest then tries to improve the generalization by training an ensemble of decision trees. Each tree is trained on a random subset of the features, meaning that they learn different attribute-class dependencies. After training the trees, predictions are made by taking the majority vote of the ensembles predictions. This can greatly reduce the degree of overfitting. A visualization of such a random forest is given in figure 2. Typically, the generalization converges to a limit, as the number of trees in the forest grows.

4.2 Logistic Regression

Logistic regression is a learning algorithm based on the sigmoid function, which takes any real numbered input and maps it between 0 and 1. The sigmoid function looks as follows:

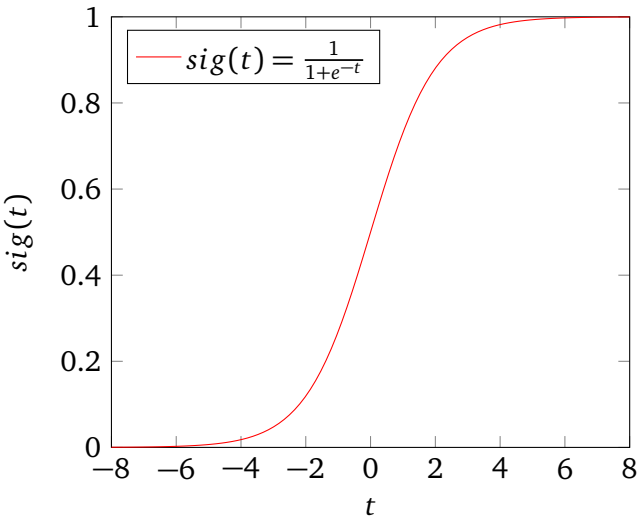


Figure 3: Sigmoid function

Since our instances are usually multi-dimensional, and not just a single number, logistic regression computes the dot product of the input instance with a weight vector β and uses that as the input to the sigmoid function. Formally, the logistic regression model then looks like

$$\frac{1}{1+e^{-(\beta_0+x_i^T\beta)}} \tag{2}$$

where β_0 is a bias and β has the same dimensionality as the instances. The models output is interpreted as the probability that an instance belongs to a certain class c_1 over c_0. Any input with a prediction above 0.5 will be labeled as class c_1 and anything below as c_0. There exist multiple ensemble based strategies to extend logistic regression to the multiclass scenario, i.e. when we have more than just two possible classes. Since we only have to use the binary model as part of our multilabel classifiers though, we will not discuss them in more detail.

Given training data $\mathcal{T}$, the model is commonly trained through maximum likelihood estimation. The weights β_0 and β are found under which the observations in the training data become most probable.

5 Multilabel Classification

Multilabel classification distinguishes itself from binary and multiclass classification by allowing for an arbitrary amount of labels in $\mathcal{L}$ to be assigned to instances. In many application scenarios this makes perfect sense, for example when trying to assign moods to music, more than just one mood may apply. Frequently, the assigned labels are also called the *relevant* labels, while the others are called *irrelevant*. Throughout this thesis we will often follow that habit.

Under the hood, this representation of an arbitrary amount of labels is most commonly realized by a vector $y = (l_1, ..., l_m)$ where $l_i = 1$ if l_i is assigned to x and $l_i = 0$ otherwise. Accordingly, our multilabel classifier can be thought of as a function that maps from $\mathcal{X}$ to $\{0,1\}^m$. Same as before, the model is induced by learning from training data $\mathcal{T} := \{(x_i, y_i)\}, i = 1, 2, ..., n$, where each instance x_i is already associated with its label vector y_i. Subsequently, it is evaluated on test data.

As part of this thesis we use three common and effective multilabel classifiers, namely binary relevance (BR), classifier chain (CC) [2] and calibrated label ranking (CLR) [3] and they will briefly be described in the following subsections. All of them are transformation based. Transformation based algorithms restructure the more complex multilabel dataset into binary or multiclass datasets, such that conventional classifiers can be used on them. During this thesis, we will refer to these underlying classifiers as the *base classifiers*.

5.1 Binary Relevance

Binary relevance learns a seperate binary classifier c_i for each label $l_i \in \mathcal{L}$. To do this, the original multilabel dataset is split into m datasets, one for each binary classifier. The instances associated with the labels are kept exactly as they are, only the labels for each instance are split and redistributed over the m datasets. Figure 4 shows this process for a dataset consisting of four instances.

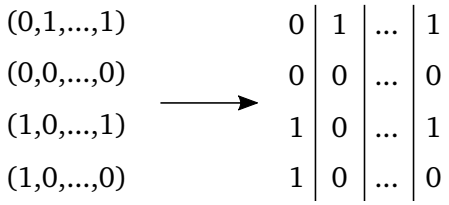


Figure 4: The labels are redistributed over m datasets

The base classifiers are then trained on their respective datasets. For a new prediction, given an instance x, each classifier c_i predicts the relevance of l_i for x. Their outputs can easily be combined to y by setting all l_i to 1 where l_i was predicted as relevant.

5.2 Classifier Chain

While binary relevance is quite effective, its biggest downside is that any dependencies between the labels are completely lost during the transformation. This may lead to situations where labels get predicted, that never occurred together in any sample. However, the base classifiers will not be able to learn such information if none of the other labels are in their training data.

The classifier chain approach tries to alleviate this problem by building on binary relevance. Same as before, we train m binary classifiers, each predicting one label in $\mathcal{L}$. However, the value of previous labels is additionally appended to the feature vector x. The instances for classifier c_i then look like $(x, l_1, ..., l_{i-1})$. Now, the value of previous labels flows into the prediction making of label l_i. To label a new instance x, the binary classifiers are chained behind each other, with the prediction of each one being added to the feature vector of the next one. It is important to note, that still, not all dependencies can be considered with this approach. If a decisive label appears later in the label chain, it can not have an effect on the previous labels. Different orders of chains can thus have an impact on the performance.

6 Label Ranking

Label Ranking describes the problem of finding a ranking over the labels in $\mathcal{L}$, given instances from $\mathcal{X}$. In Label Ranking, other than in the previous classification problems, more than just a partitioning of labels into a relevant and irrelevant set is made, instead, they are ordered by their rank. That ranking can be represented by a permutation $\mathcal{P}_x$ on $\{1, 2, ..., m\}$, where $\mathcal{P}_x(i)$ denotes the position of label l_i in the ranking. The output of the label ranker then looks like

$$l_{\mathcal{P}_x^{-1}(1)} \succ_x l_{\mathcal{P}_x^{-1}(2)} \succ_x ... \succ_x l_{\mathcal{P}_x^{-1}(m)} \tag{3}$$

where $\succ_x$ is the preference relation amongst labels for instance x. $l_i \succ_x l_j$ means that l_i is preferred to l_j, given x. $\mathcal{P}_x^{-1}(n)$ then is the index of the label that was assigned position n. For many applications, this output contains more useful information than, for instance, multilabel classification would. As an example, imagine our multilabel classifier outputs a set of 'relevant' (i.e. preferable) beverages, given a meal. If none of the relevant beverages were available, the output would be of no use. That is because all labels, i.e. beverages, in the irrelevant set are equal and thus incomparable to each other (the same, of course, is true for the relevant set). However, with a label ranker that ranks all the beverages according to preference, we would still have useful information for the lower ranking ones.

Label rankers map from $\mathcal{X}$ to the set of all permutations (total orders) over $\mathcal{L}$. As training data $\mathcal{T}$, a set of instances together with pairwise comparisons $l_i \succ_{gt} l_j$ is given, where $\succ_{gt}$ is the preference relation of the ground truth. A good label ranker should be able to also learn from partial ranking information. This is important, because in real life application scenarios, one will not always be able to bring all labels under a given instance into an order. Instead, one might just be able to rank a couple of them confidently and for the others, no statement can be made. Not being able to include such partial rankings in the training phase would only restrict the information that our label ranker can learn from. A popular measure to assess the performance of label rankers is Kendall's tau coefficient.

$$\tau = \frac{c - d}{m(m-1)/2} \tag{4}$$

c denotes the number of *concordant* label pairs and d the number of *discordant* pairs. A label pair $l_i l_j$ (with $i \neq j$) is concordant when its preference order in the output matches that of the ground truth, i.e.

$$(l_i \succ_x l_j \wedge l_i \succ_{gt} l_j) \vee (l_j \succ_x l_i \wedge l_j \succ_{gt} l_i) \tag{5}$$

If the order does not match, i.e.

$$(l_i \succ_x l_j \wedge l_j \succ_{gt} l_i) \vee (l_j \succ_x l_i \wedge l_i \succ_{gt} l_j), \tag{6}$$

the label pair is in the discordant set. Since m is the number of labels, $m(m-1)$ is the number of label pairs, excluding pairs of the same label. Half of those are 'flipped' duplicates, however, so dividing it by two results in the number of unique pairings. It follows that $c + d = m(m-1)/2$ and, thus, Kendall's tau coefficient normalizes the share of discordant label pairs to $[-1, 1]$. When the output ranking is exactly the same as the ground truth, it assumes the value 1, and in the extreme case where the two rankings are the inverse of each other, the value is -1. Kendall's tau only works when the ground truth and the output are of total order, as it assumes that every label pair is either concordant or discordant. In section 8.1 we show a modification of Kendall's tau to partial rankings, that will also be used in our evaluation process.

There exist multiple label ranking algorithms and we will now introduce a popular one that will find a role in our calibrated label ranking implementation.

6.1 Ranking by Pairwise Comparison

RPC [6] is a transformation based label ranker. It divides the label ranking problem into several subproblems that will be solved by one binary classifier each. Specifically, a model $\mathcal{M}_{ij}$ is learned for every label pair $(l_i, l_j) \in \mathcal{L}, 1 \leq i < j \leq m$. The model should output 1 if $l_i \succ_x l_j$ and otherwise 0. For that purpose, each model is trained on those instances that include preference information about the label pair (l_i, l_j) in their ground truth. All instances without that information are ignored. The associated label with the training instance is 1 or 0, depending on the pairwise order of the two labels. We can then train a conventional binary classifier for each unique label pair.

Predictions are made by letting each classifier predict their respective pairwise ranking. The challenging question then becomes how to combine all these pairwise rankings into a single one. The authors arrive at the following method. For each label a score is defined by the sum of votes

$$S(l_i) = \sum_{l_j \neq l_i} \mathcal{R}_x(l_i, l_j) \tag{7}$$

$\mathcal{R}_x(l_i, l_j)$ equals 1 if $l_i \succ_x l_j$ was predicted by $\mathcal{M}_{ij}$ and 0 otherwise. Intuitively, this equation then measures the number of times that l_i was predicted in front of another label. All the labels are then ranked according to their scores in decreasing order, i.e. such that the highest ranking label will also have the highest score. The resulting output will trivially be acyclic and transitive.

One downside of this algorithm is that the number of required classifiers increases quadratic, since we need $m(m-1)/2$ different ones to model each label pair. For large numbers of labels, this can result in slow execution times and large demands on memory.

6.2 Calibrated Label Ranking

As previously mentioned, a couple of classification problems can be solved in the label ranking setting, emphasizing the attractiveness of this problem. By simply reducing the assigned ranking to the top ranking label, we receive a binary/multiclass classifier. Furthermore, it has been shown that a multilabel classifier can also be obtained through label ranking [3]. Interestingly, the resulting multilabel classifier is one of the three that will be tested for our proposed method. We will briefly explain it in this subsection.

Intuitively, to induce a multilabel classifier from a label ranker, instead of just reducing the ranking to the top ranking label, we need to reduce it to the several top ranking ones. All those top ranking labels would then be in the *relevant* set for instance x and the lower ranking ones in the *irrelevant* set. The difficulty lies in the fact, that we do not know where that cut-off within the ranking lies.

In calibrated label ranking, that issue is solved by introducing a so called calibration label l_0 that extends the ranking by splitting the labels into a bipartition. For an output

$$l_{\mathcal{P}_x^{-1}(1)} \succ_x \ldots \succ_x l_{\mathcal{P}_x^{-1}(i)} \succ_x l_0 \succ_x l_{\mathcal{P}_x^{-1}(i+1)} \succ_x \ldots \succ_x l_{\mathcal{P}_x^{-1}(m)} \tag{8}$$

all labels ranked higher than l_0 are put into the relevant set, in this case $\{l_{\mathcal{P}_x^{-1}(1)}, \ldots, l_{\mathcal{P}_x^{-1}(i)}\}$, and the labels ranked lower into the irrelevant set, i.e. $\{l_{\mathcal{P}_x^{-1}(i+1)}, \ldots, l_{\mathcal{P}_x^{-1}(m)}\}$. As we have mentioned in the previous section, a label ranker needs at least partial ranking information for training. However, we are trying to apply label ranking to multilabel classification and thus we only have our label vector y at hand, that provides a bipartition of labels. We can deduce a couple of rank pairings from y by observing that for all relevant labels $l \succ l_0$ should hold, as should $l_0 \succ l'$ for irrelevant labels. Additionally, all relevant labels should be ranked before the irrelevant ones. Given that information as the ground truth, a label ranking model can be trained. It is only important that the label ranker accepts partial information for training. In our implementation, RPC is used as the underlying learner. We receive a label ranker whose output can be transformed into a bipartition of labels. Thus the model can be seen as a multilabel classifier.

7 Label Ranking through Mutlilabel Classification

In this section we will present our proposed method to solve the label ranking problem in the multilabel classification domain. This undertaking can be seen as the inverse approach of the previous subsection, where we derived a multilabel classifier from a label ranker. Our method is based on the principal idea, that we transform label ranking datasets into multilabel datasets, so that a multilabel classifier can then be trained on the resulting data. That transformation is done by modeling each pairwise preference between two labels as a label in itself. To extend on that point, consider a label ranking

$$l_2 \succ_{gt} l_1 \succ_{gt} l_3$$

over labels $\mathcal{L} := \{l_1, l_2, l_3\}$ is given as the ground truth for instance x. We restructure the ranking as a multilabel vector y, in which each label models the pairwise preference relation between two labels. For the given example, the resulting label vector would look as follows:

$$\begin{array}{cccccc} l_1 \succ l_2, & l_2 \succ l_1, & l_1 \succ l_3, & l_3 \succ l_1, & l_2 \succ l_3, & l_3 \succ l_2 \\ 0, & 1, & 1, & 0, & 1, & 0 \end{array}$$

The top row shows the meaning of each label and is merely there for understandability. In the bottom row are the actual values of the label entries. As we can see, the first label models $l_1 \succ l_2$ while the second one models $l_2 \succ l_1$ and so on. The fact that the first one is not set while the second one is, means that $l_2 \succ l_1$ is given as ground truth. It follows that the ranking implicitly defined by the above vector is the same as that of the ground truth ranking. In fact, it is straightforward to see that any ranking can be represented this way, under the requirement that every pairwise relation is captured by the multilabel vector. To do this, a multitude of schemes to model the vector can be derived and applied. It is just important to stay consistent within the dataset. In the appendix, we explain how exactly we modeled our label vector and list the applied algorithm. The resulting multilabel vector will have $m(m-1)$ labels and that number arrives from the simple observation that for each of the m labels, there are $m-1$ pairings with different labels. A multilabel classifier can then be trained on the resulting dataset. Likewise, any multilabel vector y that the model outputs for an unseen instance can now be interpreted as a label ranking.

If no statement regarding the preference between two labels l_i and l_j can be made, both the $l_i \succ l_j$ - and the $l_j \succ l_i$ label should be set to zero. We call this scenario an *abstention*. Abstentions can easily be incorporated into the training data, making our proposed method universally applicable, since we can train on partial rankings.

But, importantly, the classifiers prediction for a given instance may also contain abstentions, as the classifier is free to set the labels however he wishes. Abstentions in the output of label rankers could reduce the share of wrong predictions being made, while obviously also reducing the completeness of the ranking. One of the interesting observations being made in later experiments is if a multilabel classifier will also resort to abstentions, without any explicit mechanisms that force him.

Another output that our model could possibly make is one in which for two labels l_i and l_j, *both* the $l_i \succ l_j$ and $l_j \succ l_i$ labels are set, i.e.

$$\begin{array}{cc} l_i \succ l_j, & l_j \succ l_i \\ 1, & 1 \end{array}$$

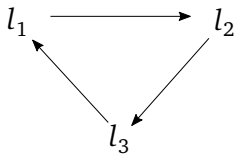
We derive two possible ways to handle this prediction.

1. Firstly, we can view this as uncertainty of the preference between the two labels. If both $l_i \succ l_j$ and $l_j \succ l_i$ are predicted, it seems that both alternatives seemed likely to the model. Therefore no clear statement regarding the preference between the two labels can be made and we treat this as an abstention.

2. Secondly, since the model seemed sure enough about both alternatives, we can view this as *two* predictions being made, one being true and one false. This results in a cyclic prediction. In subsection 8.2 on the evaluation measure *completeness*, we will reason why we chose the first approach instead of this one.

We will discuss cyclic rankings in a bit more detail now. Formally, a ranking is cyclic if there exists a closed chain in the ranking defined by $\succ$. For better visualization purposes we will depict a ranking $l_i \succ l_j$ as a directed graph

A simple cyclic ranking between $\{l_1, l_2, l_3\}$ could then look as follows:



Any ranking that contains a closed chain is considered cyclic. Our proposed method is free in the ranking that it predicts, so these cycles can occur. This could be regarded as an inconsistency. If l_3 is preferred to l_2 and l_2 to l_1, then l_3 should also be preferred to l_1. Many authors therefore demand the ranking to be acyclic. But since acyclic preferences may very well occur in realistic scenarios, we chose to keep this possibility in our model. As an example, it is possible that A wins more often versus B and B wins more versus C. However, C wins more often versus A. That is an acyclic relation that is not 'inconsistent' by any means.

Furthermore, the rankings produced by the multilabel classifier may not be transitive, as no mechanism enforces it. In section 11.5 we will enforce transitivity and see how the results compare to those where transitivity was not enforced.

8 Evaluation Measures

We have already hinted at the evaluation measures that we will use to assess the performance of our label rankers, namely *correctness* and *completeness*. In section 6 on label ranking, we introduced the conventional Kendall's tau measure.

$$\tau = \frac{c - d}{m(m-1)/2} \tag{9}$$

The reason we use different measures, is that Kendall's tau only works if complete rankings (total orders) are both the output and the ground truth. It normalizes the share of discordant label pairs to [-1, 1], but that only works if $c + d = m(m-1)/2$. Since $m(m-1)/2$ is the number label pairs, this means that any pair of labels has to either be discordant or concordant. If our output ranking is partial, any label pair that the ranker abstained from ordering is neither discordant nor concordant. Because of this we need to adapt Kendall's tau and we will also introduce a measure to evaluate the completeness, i.e. number of abstentions, in the ranking. We follow [1] for this.

To account for abstentions in the evaluation measure, let us first denote a way to formalize them. We write an abstention on labels l_i and l_j as $l_i \perp l_j$. Likewise, if no information on the preference between the two labels is given in the ground truth we write $l_i \perp_{gt} l_j$. Now, if either $l_i \perp l_j$ (label ranker made an abstention) or $l_i \perp_{gt} l_j$ (there was no need to compare l_i and l_j) is true, the label pair is neither concordant (5) nor discordant (6).

8.1 Correctness

Given this notion of concordance, discordance and abstentions, we modify Kendall's tau to

$$\frac{c-d}{c+d} \tag{10}$$

and call this measure *correctness*. Clearly, when the output ranking is the same as the ground truth, this measure still assumes 1. If the ranking is the inverse, the measure is also -1, so we can interpret it similarly to Kendall's tau. The important difference lies in the fact that abstentions are tolerated by the measure. Even if the ranker does not rank half the label pairs, it can still achieve a perfect correctness score by correctly ranking the remaining ones.

8.2 Completeness

To counterbalance this, we introduce the completeness measure to evaluate the number of abstentions in the output, relative to the ground truth. It is defined as

$$\frac{c+d}{|\succ_{gt}|} \tag{11}$$

$|\succ_{gt}|$ is the number of pairwise rankings in the ground truth. This measure then specifies the share of these label pairs, for which the label ranker made a prediction. For this, predictions that were made despite not being necessary (because $l_i \perp_{gt} l_j$) are ignored. That is because they can be neither concordant or discordant, as no ground truth on their ranking is given. Even if most of his predictions are wrong, i.e. discordant, a label ranker may still achieve a perfect completeness of 1.

In section 7 we asked ourselves the question if a scenario where both the $l_i \succ l_j$ - and $l_j \succ l_i$ - labels are set, should be interpreted as two predictions being made. If the ground truth contains a true ranking of the two labels, it follows that one of these predictions will be concordant while the other is discordant. That opens the door for a case where $c + d > |\succ_{gt}|$ and a value greater then the already ideal 1 could be obtained. That is why we chose to not interpret such a prediction this way and instead treat it as an abstention.

9 Datasets

We evaluate our method on the sixteen label ranking datasets that are part of the Uni Paderborn repository [1]. In particular, they have also been used in a lot of label ranking publications and thus make a direct comparison to other algorithms easy. The data is semi-synthetic, meaning it was modified to fit into the label ranking domain. Originally they were multi-class and regression datasets from the UCI machine learning repository and the Statlog collection. They were turned into label ranking datasets through the following methods:

- Type A: A naive Bayes classifier is trained on the multi-class dataset. After that, for each instance in the dataset, the classifier makes a prediction. The labels are then ordered according to their predicted probabilities to produce a ranking. Ties are solved by ranking the label with the lower index first. [6]

- Type B: For regression data, a number of numerical attributes are removed from the feature vector and each one is considered a label. These attributes are then standardized and ordered by their size, producing a ranking. Since the original attributes are correlated, the remaining ones will contain information about the ranking. [7]

Dataset	Type	instances	attributes	labels
authorship	A	841	70	4
bodyfat	B	252	7	7
calhousing	B	20640	4	4
Cpu-small	B	8192	6	5
elevators	B	16599	9	9
fried	B	40768	9	5
glass	A	214	9	6
housing	B	506	6	6
iris	A	150	4	3
pendigits	A	10992	16	10
segment	A	2310	18	7
stock	B	950	5	5
vehicle	A	846	18	4
vowel	A	528	10	11
wine	A	178	13	3
wisconsin	B	194	16	16

Table 1: The 16 label ranking datasets used in the following experiments

This was done due to a lack of available label ranking data and the characteristics of the datasets are shown in table 1. The type indicates how the data has been generated.

All the rankings produced in the datasets are total. Keep in mind that after the transformation explained in section 7, the number of labels increases to $m(m-1)$. For the wisconsin dataset, that means we will have to work with 240 labels.

10 Setup

We transformed the datasets from the label ranking to the multilabel domain with a python script. Our label ranking through multilabel classification method was implemented and evaluated in the Java library mulan [4]. We use the mulan implementation of binary relevance, classifier chain and calibrated label ranking. The base classifiers for these three multilabel learners are imported through weka[2]. Specifically, we try logistic regression and random forest as base classifiers. The evaluation results are obtained through a 10-fold cross validation, using the previously introduced evaluation measures. All programs were executed on a ThinkPad E470 with an Intel Core i3 (2x 2.4 GHz), 8GiB of RAM and integrated graphics.

11 Experiments

11.1 Comparison with RPC and LR-RF

As a first evaluation, we will compare our method with conventional label rankers that only output complete rankings (no abstentions). We will try to see how many abstentions our method makes and if they can improve the correctness in comparison to the baselines. We chose ranking by pairwise comparison (RPC) [6] and random forest for label ranking (LR-RF) [8] as our baselines. RPC is implemented in weka-lr [3], a label ranking standalone extension of weka. Logistic regression is used as the base classifier for RPC and we execute the algorithm with a 10-fold cross validation on the described datasets. We use

[1] https://cs.uni-paderborn.de/?id=63912
[2] https://www.cs.waikato.ac.nz/ml/weka/
[3] https://cs.uni-paderborn.de/?id=63906

Kendall's tau for evaluation purposes. As for LR-RF, we can directly use the results given in the related paper. They were also obtained in terms of Kendall's tau through a 10-fold cross validation.

Our correctness measure is comparable to Kendall's tau, as it is simply a generalization of Kendall's tau to partial rankings.

10-fold cross validation is used as a means to solve an inherent machine learning problem: We want to train our models on as much data as possible but we also want to test it on as much *different* data as possible, to get a good impression of how well the model works for new samples. That means we need to partition our dataset into a part for training and testing and we would like both parts to be maximally large. 10-fold cross validation works by randomly partitioning the dataset into 10 equal sized parts. There are 10 repetitions and in each one, a different part is used for testing, while the remaining 9 parts are used for training the model. The 10 different evaluation results are then averaged to produce a single estimation. The advantage of this is that every sample in the data was used once for testing.

Firstly, we use logistic regression as the base classifier for our proposed method. The results are shown in table 2.

			BR		CC		CLR	
	RPC	RF	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	0.908	0.913	0.9277	0.9746	0.8951	0.9994	**0.9472**	0.9764
bodyfat	**0.282**	0.185	0.2715	1	0.2161	0.9996	0.278	0.9864
calhousing	0.244	**0.367**	0.2495	1	0.2208	1	0.2501	0.9975
Cpu-small	0.449	**0.515**	0.471	1	0.4319	1	0.4783	0.9911
elevators	0.749	0.756	**0.7795**	1	0.7563	0.9999	0.5775	0.1038
fried	**0.999**	0.926	**0.999**	0.9999	0.9983	1	**0.999**	0.9996
glass	0.885	0.888	**0.8922**	0.9984	0.8728	0.996	0.8832	0.3086
housing	0.67	**0.792**	0.6594	1	0.6419	1	0.7022	0.9842
iris	0.889	**0.966**	0.8422	0.9956	0.8444	0.9978	0.8933	0.9822
pendigits	0.932	0.939	0.9294	0.9999	0.9401	0.9999	**0.9535**	0.9934
segment	0.934	**0.961**	0.9341	0.9991	0.9311	0.9991	0.9591	0.9924
stock	0.779	**0.922**	0.7888	1	0.7796	0.9999	0.8277	0.9869
vehicle	0.85	0.86	0.8542	0.9998	0.8535	0.9984	**0.8623**	0.988
vowel	0.652	**0.967**	0.664	1	0.5858	0.9968	0.7542	0.9244
wine	0.903	**0.953**	0.9031	1	0.9331	0.9963	0.9199	0.9963
wisconsin	**0.633**	0.478	0.6139	0.9998	0.409	0.9922	0.6245	0.9945

Table 2: Comparison with RPC/LR-RF; Logistic regression used as base; Corr and Compl used as measures

The highest achieved correctness for each dataset is highlighted in bold. We can see that for five datasets our method outperforms the two baselines w.r.t. correctness and there is one tie. Abstentions are also made on many datasets and calibrated label ranking abstains most often. It should be noted that it also runs the slowest.

That should not come as a surprise, as we have already noted in section 6.1 that the number of base classifiers used in RPC, which our CLR implementation is based on, grows squarely with the number of labels. For the other two algorithms, BR and CC, the number just grows linearly, since we train one separate classifier for each label. That means a significantly larger amount of calculations has to be made in calibrated label ranking and the algorithm runs slower. Aside from that, classifier chain seems a bit weaker than the other two algorithms as the correctness in most datasets is lower than for the other two, even when Binary Relevance made fewer abstentions.

The degree of abstentions seems very reasonable as well, being consistently in the high 90s. The only huge outliers come from CLR on the elevators and glass datasets, where the completeness drops to 10% and 30% respectively.

Next, let us evaluate the same experiment with random forest as the base classifier. We leave the number of decision trees in the forest at its default value, 10. The results are depicted in table 3.

	RPC	RF	BR		CC		CLR	
			Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	0.908	0.913	0.9539	0.9661	0.9252	0.9968	**0.956**	0.9616
bodyfat	**0.282**	0.185	0.216	0.7649	0.1314	0.997	0.2681	0.7656
calhousing	0.244	0.367	0.5196	0.8482	0.4229	1	**0.5637**	0.8043
Cpu-small	0.449	0.515	0.5605	0.8468	0.4754	1	**0.5959**	0.8366
elevators	0.749	0.756	**0.8359**	0.9454	0.7821	1	0.7388	0.0917
fried	**0.999**	0.926	0.9946	0.9923	0.9708	1	0.9951	0.9882
glass	0.885	0.888	**0.9293**	0.9708	0.8968	0.9991	0.9086	0.2942
housing	0.67	0.792	0.8546	0.9558	0.7913	1	**0.8683**	0.9446
iris	0.889	**0.966**	0.9511	0.9956	0.9556	1	0.9556	0.9889
pendigits	0.932	0.939	0.9832	0.9891	0.9664	0.9996	**0.9841**	0.987
segment	0.934	0.961	0.9828	0.9913	0.972	0.9995	**0.9843**	0.9886
stock	0.779	0.922	0.9323	0.9837	0.9211	1	**0.9376**	0.9782
vehicle	0.85	0.86	0.9086	0.9531	0.8718	0.9998	**0.9157**	0.9395
vowel	0.652	**0.967**	0.9301	0.9635	0.8834	0.998	0.9298	0.9026
wine	0.903	0.953	**0.9625**	0.9667	0.9403	0.9926	0.9609	0.9572
wisconsin	**0.633**	0.478	0.5884	0.8307	0.4705	0.9558	0.5983	0.8758

Table 3: Comparison with RPC/LR-RF; Random forest used as base; Corr and Compl used as measures

We can see that with random forest as the base, our method beats the baselines in 11 datasets, a significant improvement compared to logistic regression. Interestingly, that improvement is linked with a simultaneous decrease in completeness. That is in indicator that abstentions are indeed made for uncertain decisions. The exclusion of these decisions where there is a good chance for failure would logically result in a reduced share of discordant label pairs. Still, classifier chain performs the worst out of the three w.r.t. correctness, but it is also the only classifier that has seen an improvement in completeness compared to before. Meanwhile completeness has significantly dropped in a couple of datasets for both CLR and BR. For instance, the great correctness measures for calhousing and cpu-small come with a hefty trade-off in completeness. At the same time, CC did not make any abstentions on these two datasets, but also has significantly more mistakes in the rankings. Again, CLR is the most abstention friendly classifier and CC seems to be the most stable when it comes to achieving a ranking as complete as possible.

This makes sense when you recall how CC works (see section 5.2), as it predicts all the labels in a chain, one after another. The value of the previous labels is then included in the input of the classifier that predicts the next label. That enables the model to learn some of the label dependencies in the training phase. Since the datasets we use in this experiment are complete (contain only total orders), one of the $l_i \succ_{gt} l_j$ - and $l_j \succ_{gt} l_i$ - labels will always be set to 1 while the other is set to 0. That dependency is something that CC will learn and thus it will not make a lot of abstentions.

Another observation is that on every dataset where RPC (which uses logistic regression as the base classifier) outperformed random forest for label ranking, our multilabel classifiers with the logistic base achieved a better correctness on average than when they used the random forest base. Likewise, each time LR-RF outperformed RPC, it was the other way around. This of course is not too surprising. When a logistic regression/random forest based label ranker performed better for a certain dataset, it is a logical assumption that that base would also serve the multilabel classifier better.

11.1.1 Correctness and Completeness Multiplied

In the previous subsection we displayed the results for correctness and completeness separately. It depends on individual consideration, to decide if a better correctness at the cost of a worse completeness is better or not. Another way to evaluate the performance is to combine the two measures into a single one, simplifying the comparison. The most logical variant to do this would be to multiply the completeness with the correctness. That way, if the ranking is complete, the measure is exactly the same as Kendall's tau (which only works on complete rankings anyway). If the ranking is only 50% complete, the classifier can maximally achieve a total score of 50%, but only if the remaining rankings are all concordant.

It is important to note that this practically values abstentions as wrong predictions. A baseline classifier that ranks half of the label pairs wrong and receives a Kendall's tau coefficient of 0.5 would be considered equal to label ranker that only made concordant rankings but abstained from 50% of label pairs. That should be kept in mind, when looking at the results. Again, the best result for each dataset is highlighted in bold. Table 4 shows the results with logistic regression as the base classifier.

	RPC	RF	BR Corr*Compl	CC Corr*Compl	CLR Corr*Compl
Authorship	0.908	0.913	0.9041	0.8946	**0.9248**
bodyfat	**0.282**	0.185	0.2715	0.2160	0.2742
calhousing	0.244	**0.367**	0.2495	0.2208	0.2495
Cpu-small	0.449	**0.515**	0.4710	0.4319	0.4740
elevators	0.749	0.756	**0.7795**	0.7562	0.0599
fried	**0.999**	0.926	0.9989	0.9983	0.9986
glass	0.885	0.888	**0.8908**	0.8693	0.2726
housing	0.67	**0.792**	0.6594	0.6419	0.6911
iris	0.889	**0.966**	0.8385	0.8425	0.8774
pendigits	0.932	0.939	0.9293	0.9400	**0.9472**
segment	0.934	**0.961**	0.9333	0.9303	0.9518
stock	0.779	**0.922**	0.7888	0.7795	0.8169
vehicle	0.85	**0.86**	0.8540	0.8521	0.8520
vowel	0.652	**0.967**	0.6640	0.5839	0.6972
wine	0.903	**0.953**	0.9031	0.9296	0.9165
wisconsin	**0.633**	0.478	0.6138	0.4058	0.6211

Table 4: Comparison with RPC/LR-RF; Logistic regression used as base; Corr*Compl used as measure

The overall impression is not too different. That stems from the fact, that the degree of completeness is quite high for logistic regression. For the vehicle dataset, the preferable algorithm has changed, but they are all very close, still.

We show the same results for random forest in table 5. Here the change is a bit more striking. The biggest difference can be noted in the evaluation of CLR. While it previously had the best correctness for 8 datasets, that number reduces to 3 after punishing abstentions. That is expected, as we have already noted how CLR makes the most abstentions, especially under random forest.

On the flipside, classifier chain is now the preferable algorithm for two datasets, when it previously was not for any at all. That of course comes from the fact that it made consistently few abstentions, especially compared to the other two. But still, the gain in correctness for BR and CLR over CC was so considerable on some datasets, that the measure remains higher (or at least competitive), even after taking the many abstentions into account. Examples for this are the bodyfat, wisconsin, calhousing and cpu-small datasets.

	RPC	RF	BR Corr*Compl	CC Corr*Compl	CLR Corr*Compl
Authorship	0.908	0.913	0.9216	**0.9222**	0.9193
bodyfat	**0.282**	0.185	0.1652	0.1310	0.2053
calhousing	0.244	0.367	0.4407	0.4229	**0.4534**
Cpu-small	0.449	**0.515**	0.4746	0.4754	0.4985
elevators	0.749	0.756	**0.7903**	0.7821	0.0677
fried	**0.999**	0.926	0.9869	0.9708	0.9834
glass	0.885	0.888	**0.9022**	0.8960	0.2673
housing	0.67	0.792	0.8168	0.7913	**0.8202**
iris	0.889	**0.966**	0.9469	0.9556	0.9450
pendigits	0.932	0.939	0.9725	0.9660	**0.9713**
segment	0.934	0.961	**0.9742**	0.9715	0.9731
stock	0.779	**0.922**	0.9171	0.9211	0.9172
vehicle	0.85	0.86	0.8660	**0.8716**	0.8603
vowel	0.652	**0.967**	0.8962	0.8816	0.8392
wine	0.903	**0.953**	0.9304	0.9333	0.9198
wisconsin	**0.633**	0.478	0.4888	0.4497	0.5240

Table 5: Comparison with RPC/LR-RF; Random forest used as base; Corr*Compl used as measure

11.2 Comparison with Cheng et al.

We have previously mentioned that other abstention based label rankers have already been introduced. The concept was first described by Cheng et al. [1] and we will compare our method with the one given in the paper. A description of their specific method is given in section 3. We use the same evaluation measures, simplifying a direct comparison. In the related paper, the evaluation was conducted on the same datasets, but only on those of type A. Presumably, the method for deriving label ranking data from regression datasets was derived at a later point. This means that our comparison spans fewer datasets.

Cheng et al. implement their approach with RPC as the label ranker, which in turn uses logistic regression as the base learner. They perform five repetitions of a 10-fold cross validation to obtain their results. For the following comparison we will use Corr*Compl as the measure. Their completeness is consistently above 98%, so their Corr*Compl value is mostly impacted by the correctness. Tables 6 and 7 show the results.

	Cheng Corr*Compl	BR Corr*Compl	CC Corr*Compl	CLR Corr*Compl
Authorship	**0.9306**	0.9041	0.8946	0.9248
glass	0.8831	**0.8908**	0.8693	0.2726
iris	**0.9018**	0.8385	0.8425	0.8774
pendigits	0.9321	0.9293	0.94	**0.9472**
segment	0.9361	0.9333	0.9303	**0.9518**
vehicle	0.8511	**0.854**	0.8521	0.852
vowel	0.6491	0.664	0.5839	**0.6972**
wine	0.9287	0.9031	**0.9296**	0.9165

Table 6: Comparison with Cheng; Logistic regression used as base; Corr*Compl used as measure

Our proposed method does quite well versus the baseline. On the given datasets, RF as the base performed better than LR and because of that, the number of winning datasets is also higher. If one were to only chose one classifier, given the data, CC with random forest would be the best, as it performs better than the baseline on all but one dataset. However, all the algorithms do very well.

	Cheng Corr*Compl	BR Corr*Compl	CC Corr*Compl	CLR Corr*Compl
Authorship	**0.9306**	0.9216	0.9222	0.9193
glass	0.8831	**0.9022**	0.896	0.2673
iris	0.9018	0.9469	**0.9556**	0.945
pendigits	0.9321	0.9725	0.966	**0.9713**
segment	0.9361	**0.9742**	0.9715	0.9731
vehicle	0.8511	0.866	**0.8716**	0.8603
vowel	0.6491	**0.8962**	0.8816	0.8392
wine	0.9287	0.9304	**0.9333**	0.9198

Table 7: Comparison with Cheng; Random forest used as base; Corr*Compl used as measure

For logistic regression, the algorithm that performed better than the baseline on the most datasets is CLR, although it only did better on four datasets, compared to the seven of CC under random forest. Overall, Cheng's method is only consistently better on one dataset.

What has to be taken into account, is that the utilized algorithm in the baseline implementation was RPC. We have already noted how our approach can outperform RPC (in terms of correctness) on many datasets and the three datasets in which RPC performed the best could not be a part of this evaluation. Since their method works with any conventional label ranker, it would also be interesting to see how it performs with newer, improved ones.

11.3 Parametrization

After having compared our approach to various baselines and seeing how the results look promising, we will see if we can improve the overall performance by means of changing the parameters of the utilized base learners.

11.3.1 Random Forest: No. of Trees

We will try to see how the number of trees affects the ranking outputted by the multilabel learners. The default value for the number of trees is 10 and we previously left that parameter unchanged. For the following evaluation we increased the number to 30 and then 60. First of all, the model runs slower and needs more memory the more trees it contains. This is especially true for the calibrated label ranking algorithm. As we have already mentioned, the number of base classifiers used in our CLR implementation is already significantly higher than for BR and CC. That is because it grows squarely with the number of labels, as opposed to linearly. By increasing the number of trees in each base classifier, the disparity in execution time and memory demand thus grows even larger.

Because of that, we could not train the CLR model on the elevators dataset for both 30 and 60 trees, due to an OutOfMemory error (using 8GiB of RAM). At 60 trees, that issue also applied to the pendigits and wisconsin datasets. That is why we stopped at 60 trees and did not increase the number even further. Figures 5 to 7 depict the change in correctness and completeness for all three classifiers. The exact change in the values is shown in tables 8 and 9, the change for 60 trees is relative to 30 trees.

We can see that the completeness steadily increases across all classifiers by increasing the number of trees. For classifier chain, the completeness is now perfect for all datasets except wisconsin. The lower completeness score for wisconsin may have something to do with the sheer amount of labels in that dataset (240 after mapping it to multilabel data).

For binary relevance and calibrated label ranking, the correctness decreases mostly. It does not stop decreasing, even when changing the number of trees to 60. However, the changes are small and no conclusion about any strong correlation can be given. It would be interesting to see if the correctness starts to improve once the completeness converged to its limit, but the required number of trees would strongly impact the algorithms performance.

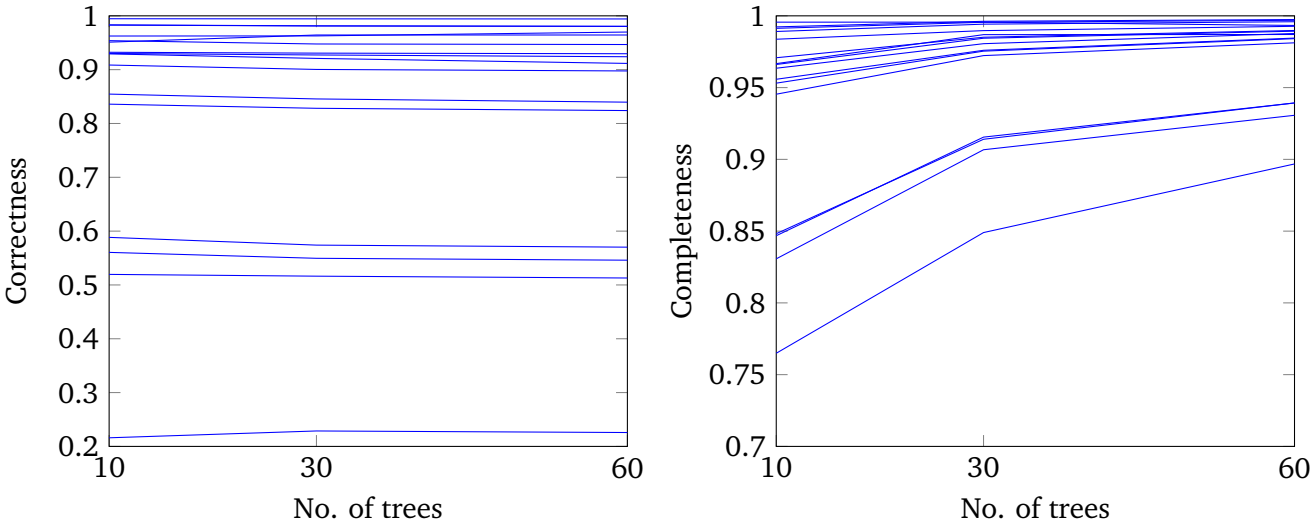


Figure 5: Effect of tree number for binary relevance

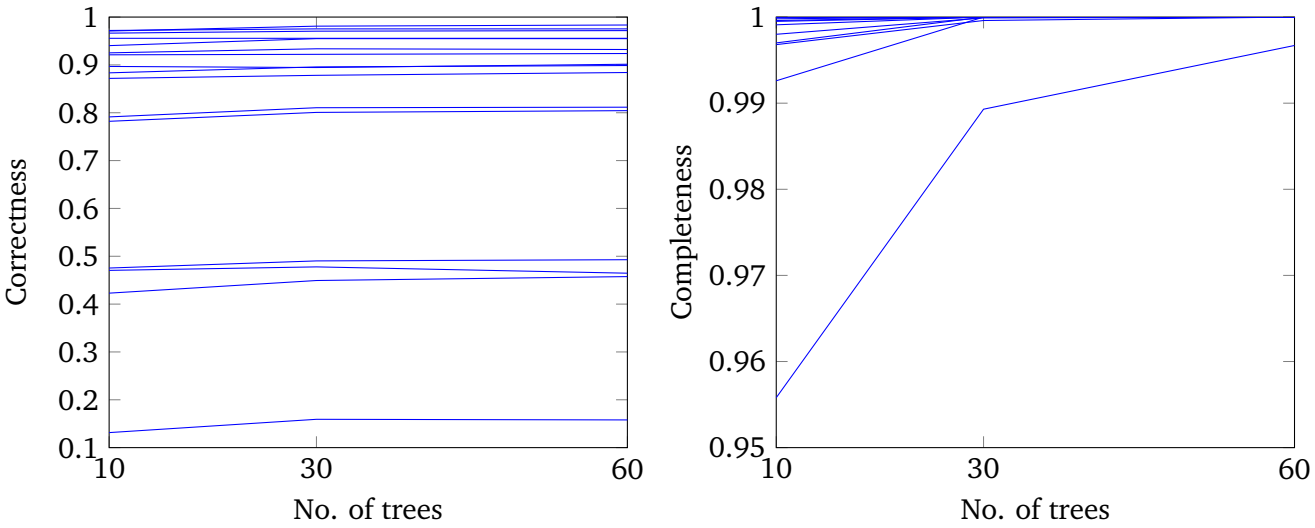


Figure 6: Effect of tree number for classifier chain

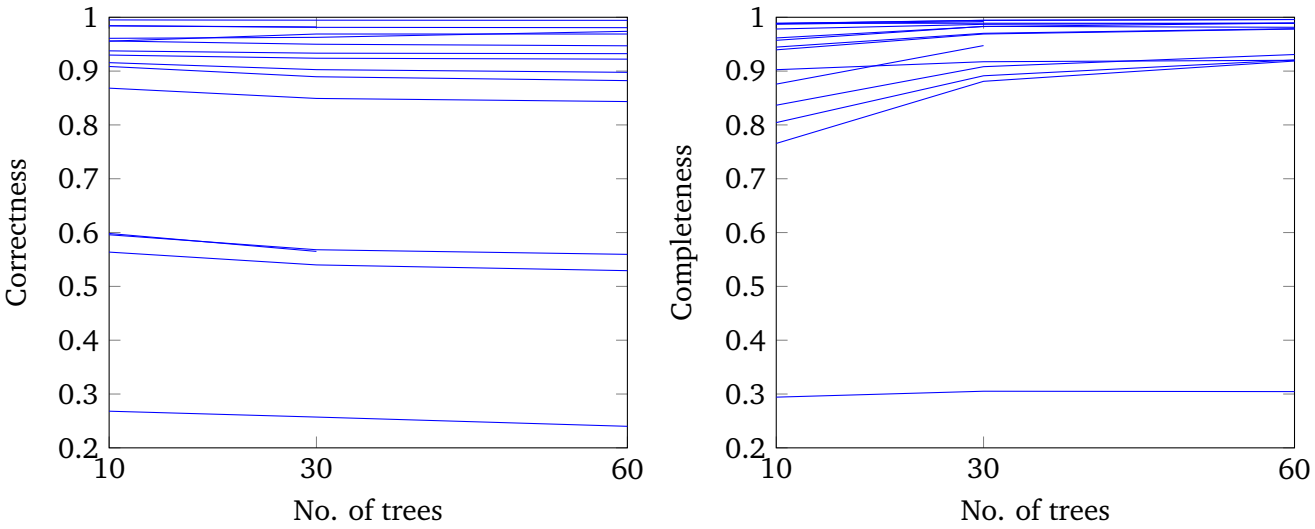


Figure 7: Effect of tree number for calibrated label ranking

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	-0.0062	+0.0183	+0.0085	+0.0028	-0.0061	+0.0212
bodyfat	+0.0127	+0.0841	+0.0278	+0.0030	-0.0109	+0.1154
calhousing	-0.0033	+0.0658	+0.0264	0	-0.0238	+0.0870
Cpu-small	-0.0110	+0.0687	+0.0149	0	-0.0278	+0.0716
elevators	-0.0077	+0.0269	+0.0186	0	———	———
fried	-0.0002	+0.0039	+0.0102	0	-0.0005	+0.0061
glass	-0.0084	+0.0143	-0.0023	+0.0009	-0.0191	+0.0110
housing	-0.0090	+0.0202	+0.0192	0	-0.0191	+0.0256
iris	+0.0133	0	0	0	+0.0133	0
pendigits	-0.0018	+0.0051	+0.0041	+0.0004	-0.0021	+0.0050
segment	-0.0016	+0.0042	+0.0033	+0.0005	-0.0030	+0.0058
stock	-0.0016	+0.0060	+0.0010	0	-0.0043	+0.0087
vehicle	-0.0082	+0.0223	+0.0064	+0.0002	-0.0130	+0.0294
vowel	-0.0028	+0.0171	+0.0124	+0.0019	-0.0059	+0.0148
wine	+0.0002	+0.0202	+0.0146	+0.0074	+0.0018	+0.0259
wisconsin	-0.0144	+0.0760	+0.0073	+0.0335	-0.0333	+0.0716

Table 8: Change in Completeness and Correctness from 10 to 30 trees

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	-0.0010	+0.0049	-0.0015	+0.0004	-0.0029	+0.0063
bodyfat	-0.0029	+0.0478	-0.0013	0	-0.0172	+0.0377
calhousing	-0.0034	+0.0253	+0.0082	0	-0.0107	+0.0294
Cpu-small	-0.0035	+0.0238	+0.0025	0	-0.0086	+0.0226
elevators	-0.0042	+0.0089	+0.0037	0	———	———
fried	-0.0002	+0.0012	+0.0024	0	-0.0003	+0.0016
glass	-0.0093	+0.0047	+0.0069	0	-0.0072	-0.0007
housing	-0.0060	+0.0085	+0.0011	0	-0.0058	+0.0086
iris	0	-0.0023	0	0	0	0
pendigits	-0.0009	+0.0017	+0.0012	0	———	———
segment	-0.0008	+0.0014	0	0	-0.0006	+0.0014
stock	-0.0011	+0.0030	+0.0017	0	-0.0009	+0.0023
vehicle	-0.0028	+0.0087	+0.0059	0	-0.0050	+0.0092
vowel	-0.0033	+0.0071	+0.0029	+0.0001	-0.0017	+0.0027
wine	+0.0072	+0.0001	0	0	+0.0112	-0.0017
wisconsin	-0.0038	+0.0240	-0.0132	+0.0074	———	———

Table 9: Change in Completeness and Correctness from 30 to 60 trees

CC is the only learner where the correctness generally improves. That might be because the completeness scores were already very high to begin with. The final conclusion is that we can steadily improve completeness by increasing the number of trees in the random forest base classifier.

11.3.2 Logistic Regression: Varying the Ridge

The logistic regression implementation in weka, which we use in our models, makes use of a technique called ridge regression. It is used as a means to improve the conditioning of ill-posed problems, i.e. problems where the numerical solution is unstable to compute. This is done by adding an additional term to the original problem, which can improve conditioning. That term is scaled by a factor which we can set in the logistic regression class. By default it is set to 0, meaning that the term is left out of the equation. We empirically tried a variety of different values for the ridge.

Overall, we observed that the effect of ridge regression on the evaluation outcome is highly dependent on the dataset. No universal recommendation for a best parameter can be given. Negative values for the regressor usually lead to significantly worse performance and should be avoided. Improvements could be most frequently noted for classifier chain in about half the datasets and with a parameter of $\sim$1. On a few datasets, however, no improvements whatsoever could be achieved, while for authorship and vowel, correctness could be improved by up to 5%.

In the end, one should explore the effect of different ridge parameters for each application scenario individually. In some cases it can improve the performance meaningfully.

11.4 Training on Incomplete Rankings

We have previously discussed that a good label ranker should be able to learn from partial information as well, instead of only from total rankings. That broadens the applicability of our learner, as for some problems it would be very challenging to produce a large enough dataset in which we have a total ranking for each instance. The utilized datasets for all previous evaluations were complete though, so our label ranking model had the luxury of being trained on total rankings only. As part of this thorough evaluation, we will change the amount of given information in the data and see how the model behaves on partial information.

We reduce the ranking by only including information regarding a certain amount of top ranking labels. Of course, the top ranking labels for a given sample are the ones that are most strongly linked to it. The label ranker will therefore get some share of the most valuable information in the training data, but not all of it. One possibility is that by making the label ranker train on the most prominent information only, it will make fewer but more accurate decisions.

To demonstrate that change in the data, we will reuse the example given in section 7:

$$l_2 \succ_{gt} l_1 \succ_{gt} l_3$$

Say we only want to include information regarding the top ranking label. We would then only model the pairwise rankings with l_2 in it. The other pairwise rankings, we abstain on. The resulting label vector looks as follows:

$$\begin{array}{cccccc} l_1 \succ l_2, & l_2 \succ l_1, & l_1 \succ l_3, & l_3 \succ l_1, & l_2 \succ l_3, & l_3 \succ l_2 \\ 0, & 1, & 0, & 0, & 1, & 0 \end{array}$$

Regarding the upcoming evaluation, it is important to remember that now a perfect completeness of 1 does not mean that the model only produced total rankings. Completeness is always relative to the rankings given in the ground truth, meaning that if the model outputs a prediction for each one given in the ground truth, the completeness for that sample is 1. The model is free to abstain from decisions that are also not made in the ground truth.

As a first experiment, we only included the pairwise rankings regarding the top ranking label in the datasets. Tables 10 and 11 show the change in completeness and correctness for logistic regression and random forest respectively. The measures dropped significantly across almost all datasets.

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	+0.0236	-0.0221	+0.0732	-0.0117	/	-0.9764
bodyfat	-0.2366	-0.8543	-0.6093	-0.5811	/	-0.9864
calhousing	-0.1784	-0.7008	-0.2816	-0.3536	/	-0.9975
Cpu-small	-0.3514	-0.7942	-0.5283	-0.4113	/	-0.9911
elevators	-0.4244	-0.5435	-0.3407	-0.2556	/	-0.1038
fried	-0.2502	-0.3002	-0.4084	-0.1539	/	-0.9996
glass	-0.3019	-0.2693	-0.4212	-0.2167	/	-0.3086
housing	-0.2035	-0.4233	-0.2786	-0.2551	/	-0.9842
iris	-0.0289	-0.2123	+0.1156	-0.0111	-0.8733	-0.9722
pendigits	-0.0163	-0.0711	-0.0639	-0.0551	/	-0.9934
segment	-0.0536	-0.1049	-0.0960	-0.0684	/	-0.9924
stock	-0.1204	-0.2163	-0.1059	-0.1223	/	-0.9869
vehicle	-0.0255	-0.1266	-0.0615	-0.0677	/	-0.9880
vowel	-0.2552	-0.5154	-0.1126	-0.2335	/	-0.9244
wine	+0.0126	-0.0699	-0.0109	-0.0241	-0.9032	-0.9769
wisconsin	-0.5773	-0.6192	-0.6711	-0.5932	/	-0.9945

Table 10: Change in Completeness and Correctness for logistic regression with only the top ranking label modeled

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	+0.0057	+0.0010	+0.0522	-0.0214	/	-0.9616
bodyfat	-0.3388	-0.5309	-0.5760	-0.6354	/	-0.7656
calhousing	-0.1872	-0.2123	-0.2187	-0.2653	/	-0.8043
Cpu-small	-0.4042	-0.3529	-0.4683	-0.3723	/	-0.8366
elevators	-0.3480	-0.2512	-0.3761	-0.2599	/	-0.0917
fried	-0.0230	-0.0135	-0.0154	-0.0169	/	-0.9882
glass	-0.2611	-0.1790	-0.2745	-0.1783	/	-0.2942
housing	-0.0817	-0.0830	-0.0639	-0.1118	/	-0.9446
iris	-0.0044	-0.0223	-0.0223	-0.0167	-0.9289	-0.9756
pendigits	-0.0346	-0.0254	-0.0291	-0.0341	/	-0.9870
segment	-0.0373	-0.0254	-0.0307	-0.0278	/	-0.9886
stock	-0.0965	-0.0611	-0.1190	-0.0745	/	-0.9782
vehicle	-0.0633	-0.0512	-0.0478	-0.0715	/	-0.9395
vowel	-0.2051	-0.1643	-0.1808	-0.1967	/	-0.9026
wine	-0.0069	-0.0056	+0.0146	-0.0290	-0.9498	-0.9461
wisconsin	-0.5379	-0.6196	-0.4455	-0.7846	/	-0.8758

Table 11: Change in Completeness and Correctness for random forest with only the top ranking label modeled

Most strikingly, calibrated label ranking falls apart and makes no predictions on all but two datasets. On the concerning datasets, no change in the correctness measure can be given, as there are no discordant or concordant label pairs and thus $(c-d)/(c+d)$ cannot be computed. That also means that on all datasets where no change in correctness is given, the computed completeness measure was 0. On the two datasets where CLR did not abstain from every decision (iris and wine), the share of decisions is still vanishingly small. On top, those two datasets have only three labels, the least amount among all datasets. That means the share of given information in the ground truth is the largest here.

As for BR and CC, the measures look good for authorship. A small tendency between logistic regression and random forest can be noted. Often, the drop for both measures is lower in random forest. For iris, pendigits, segment, vehicle, wine and fried the values are still very respectable, as they are close to the initial ones. Especially for the fried dataset it is striking how much worse the logistic regression based learners compare to random forest.

The share of abstentions differs quite a lot across the datasets. When we have 16 labels, only modeling the first one above the others and making the remaining label pairs incomparable produces a much more incomplete ranking than when we have 3 labels. For that reason, we will try to make the share of abstentions more equal by including every pairwise ranking containing a top 25% ranked label. For wisconsin, this means that now everything regarding the 4 top ranked labels is modeled and the rest is abstained on. The results of this experiment are shown in tables 12 and 13.

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	+0.0236	-0.0221	+0.0732	-0.0117	/	-0.9764
bodyfat	-0.0901	-0.7478	-0.1255	-0.3966	-0.1345	-0.7313
calhousing	-0.1784	-0.7008	-0.2816	-0.3536	/	-0.9975
Cpu-small	-0.3514	-0.7942	-0.5283	-0.4113	/	-0.9911
elevators	-0.2710	-0.4544	-0.0959	-0.1427	/	-0.1038
fried	-0.2502	-0.3002	-0.4084	-0.1539	/	-0.9996
glass	-0.0025	-0.1177	-0.0674	-0.0799	-0.1277	-0.0409
housing	+0.1012	-0.1958	+0.0967	-0.1265	-0.2693	-0.9053
iris	-0.0289	-0.2123	+0.1156	-0.0111	-0.8733	-0.9722
pendigits	+0.0577	-0.0722	+0.0294	-0.0332	/	-0.9934
segment	+0.0503	-0.0531	+0.0314	-0.0282	/	-0.9924
stock	-0.1204	-0.2163	-0.1059	-0.1223	/	-0.9869
vehicle	-0.0255	-0.1266	-0.0615	-0.0677	/	-0.9880
vowel	+0.2153	-0.3066	+0.2444	-0.1374	-0.5251	-0.9066
wine	+0.0126	-0.0699	-0.0109	-0.0241	-0.9032	-0.9769
wisconsin	+0.1690	-0.3999	+0.1108	-0.3200	-0.4652	-0.9881

Table 12: Change in Completeness and Correctness for logistic regression with only the top 25% ranking labels modeled

As you would expect, the drop in measures decreases, especially for the wisconsin dataset. In fact, BR and CC can improve the correctness compared to the total rankings on seven datasets each. Logistic regression based learners can significantly improve their correctness for vowel, although the measures are still lower than they are for random forest, which performed much better on the dataset to begin with. We still note huge drops in correctness for cpu-small and calhousing, but those are datasets that none of the label rankers performed particularly well on, even with total rankings.

While the correctness is improved on a couple of datasets, the completeness almost universally drops. That is not surprising, as the classifiers will learn to make more abstentions due to the incomplete data. Some of those abstentions will be made on labels where a predictions should have been made, reducing the completeness.

As for CLR, the measures also improve, now for seven datasets at least some predictions are made. The highest completeness values are achieved for bodyfat and glass, with approximately 30±5%. Glass of course is a dataset where the completeness of CLR was never good. But still, on the majority of datasets, CLR falls apart and makes no predictions at all, even when a good share of training information is given. While the learner looked promising in previous evaluations and could achieve the best correctness measure on many datasets, we would not recommend using it when partial information is used for training. This experiment strongly confirms our observation that CLR is the most abstention friendly classifier amongst the three.

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	+0.0057	+0.0010	+0.0522	-0.0214	/	-0.9616
bodyfat	-0.0905	-0.3564	-0.1660	-0.4485	-0.1406	-0.4427
calhousing	-0.1872	-0.2123	-0.2187	-0.2653	/	-0.8043
Cpu-small	-0.4042	-0.3529	-0.4683	-0.3723	/	-0.8366
elevators	-0.0729	-0.1217	-0.1266	-0.1453	/	-0.0917
fried	-0.0230	-0.0135	-0.0154	-0.0169	/	-0.9882
glass	+0.0041	-0.0382	-0.0261	-0.0680	-0.1424	-0.0243
housing	+0.0457	-0.0494	+0.0780	-0.0774	-0.3780	-0.8503
iris	-0.0044	-0.0223	-0.0223	-0.0167	-0.9289	-0.9756
pendigits	+0.0066	-0.0149	+0.0115	-0.0219	/	-0.9870
segment	+0.0054	-0.0136	+0.0099	-0.0164	/	-0.9886
stock	-0.0965	-0.0611	-0.1190	-0.0745	/	-0.9782
vehicle	-0.0633	-0.0512	-0.0478	-0.0715	/	-0.9395
vowel	+0.0408	-0.0524	+0.0605	-0.0706	-0.5963	-0.8792
wine	-0.0069	-0.0056	+0.0146	-0.0290	-0.9498	-0.9461
wisconsin	+0.0495	-0.2821	+0.0604	-0.3732	-0.4787	-0.8715

Table 13: Change in Completeness and Correctness for random forest with only the top 25% ranking labels modeled

Regarding the other two learners, the conclusion is mixed. While they can improve the share of correct decisions on some datasets, they fail to connect to previous performances on others. Nevertheless, all improvements are made at the cost of completeness. More complete training data will therefore be favorable in almost all cases.

11.5 Implementing Transitivity

So far, the learners were completely unrestricted in their outputs. That means that their rankings may both be cyclic and intransitive. Transitivity can be seen as a logical consequence of consistency. When $l_1 \succ l_3$ and $l_3 \succ l_2$ are modeled, then so should $l_1 \succ l_2$. It should be noted that a transitive ranking may still be cyclic. Transitivity can be enforced by computing the transitive closure of the ranking and using that as the output. Multiple algorithms exist to compute the transitive closure of a directed graph and we use Floyd-Warshall, which has a time complexity of $O(n^3)$ where n is the number of labels.

To integrate the algorithm into our setup, we first have to transform the ranking into an adjacency matrix, then run Floyd-Warshall on that matrix and map it back to a ranking. Computing the transitive closure can only *add* edges to the underlying graph, never remove them. Intuitively we would then expect the number of predictions to increase or at the very least stagnate. That should improve the Completeness of the ranking. We evaluate our approach with enforced transitivity and the results can be seen in tables 14 and 15.

As we can see, contrary to previous expectation, completeness dropped almost universally. The reason for this is that we regard pairwise rankings where both the $l_i \succ l_j$ - and $l_j \succ l_i$ - labels are set to 1 as abstentions. Computing the transitive closure can very much increase the number of these abstentions in the ranking and thus effectively decrease the completeness. For an illustrative example, consider computing the transitive closure of a cyclic ranking as shown in figure 8.

As we can see, while the non-transitive ranking was complete, three of the previous predictions are now considered abstentions, as they become bidirectional. This problem can only occur when the initial ranking was cyclic. Implicitly our experiment thus confirms that many of the rankings predicted by our classifiers are indeed cyclic. In fact, in every case where the completeness dropped, cycles in the ranking had to be present. Because of that, computing the transitive closure makes more sense when we already enforced the output to be acyclic.

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	+0.0040	-0.0091	+0.0064	-0.0172	+0.0006	-0.0029
bodyfat	+0.0233	-0.1127	0	-0.0006	+0.0228	-0.0980
calhousing	+0.0021	-0.0102	0	0	+0.0017	-0.0095
Cpu-small	+0.0087	-0.0270	-0.0001	-0.0001	+0.0109	-0.0301
elevators	-0.0097	-0.0656	0	0	+0.0064	+0.0365
fried	0	0	0	0	+0.0001	-0.0001
glass	+0.0035	-0.0522	-0.0048	-0.0415	+0.0041	+0.0738
housing	+0.0173	-0.0713	+0.0012	-0.0095	+0.0129	-0.0434
iris	-0.0133	-0.0578	-0.0088	-0.0534	-0.0200	-0.0155
pendigits	+0.0129	-0.0225	+0.0003	-0.0008	+0.0049	-0.0099
segment	+0.0085	-0.0128	+0.0006	-0.0065	+0.0043	-0.0078
stock	+0.0265	-0.0385	-0.0004	-0.0022	+0.0095	-0.0154
vehicle	+0.0071	-0.0146	0	-0.0004	+0.0051	-0.0208
vowel	+0.0856	-0.2321	+0.0080	-0.2741	+0.0683	-0.1389
wine	+0.0019	-0.0281	-0.0018	-0.0056	-0.0054	-0.0188
wisconsin	+0.1352	-0.2989	-0.0796	-0.8306	+0.1040	-0.2233

Table 14: Change in Completeness and Correctness for logistic regression with enforced transitivity

	BR		CC		CLR	
	Correctness	Completeness	Correctness	Completeness	Correctness	Completeness
Authorship	-0.0053	-0.0051	-0.0009	-0.0018	-0.0082	-0.0078
bodyfat	-0.0280	-0.4715	-0.0004	-0.0161	-0.0143	-0.3369
calhousing	-0.0190	-0.0997	0	0	-0.0416	-0.0877
Cpu-small	-0.0056	-0.1304	0	0	-0.0097	-0.1077
elevators	-0.0316	-0.1311	0	0	-0.0596	+0.0474
fried	+0.0002	-0.0002	0	0	+0.0001	-0.0008
glass	-0.0009	-0.0375	+0.0006	-0.0019	+0.0006	+0.0990
housing	+0.0054	-0.0508	+0.0007	-0.0048	+0.0001	-0.0369
iris	0	0	0	0	0	0
pendigits	+0.0021	-0.0051	+0.0006	-0.0009	+0.0012	-0.0033
segment	+0.0016	-0.0046	+0.0003	-0.0004	+0.0011	-0.0055
stock	+0.0025	-0.0036	+0.0007	-0.0016	+0.0031	-0.0040
vehicle	+0.0021	-0.0170	-0.0008	-0.0018	+0.0047	-0.0245
vowel	+0.0177	-0.1359	+0.0044	-0.0087	+0.0233	-0.0646
wine	-0.0075	-0.0115	-0.0020	-0.0170	-0.0059	-0.0076
wisconsin	-0.0304	-0.6311	+0.0582	-0.2372	+0.0900	-0.3511

Table 15: Change in Completeness and Correctness for random forest with enforced transitivity

Figure 8: Example: Computing transitive closure on a cyclic ranking

Nevertheless, except for some outliers, the completeness of CC dropped the least. This is an indication that the multilabel learner does not include too many cycles in its predictions. Again, this can be attributed to the label dependencies that CC learns. As cycles do not occur in the training data, the model will learn to not produce many cyclic rankings itself.

Correctness improves more often for logistic regression than for random forest. It would be interesting to see the change in correctness when only acyclic rankings are possible. In such a scenario, we would expect the correctness to improve when most of the pairwise rankings are already correct and to decrease when they are not.

A strong conclusion we can draw from this experiment is that CC generally outputs the least amount of cyclic rankings.

11.6 Examining the Abstentions on the Vowel Dataset

As a final test, we want to empirically see if the abstentions made by the learners follow an intuitive reason. To do this, we utilize the great interpretability of the labels in the vowel dataset. These labels represent the eleven steady state vowels used in the English language. The vowel label ranking dataset was originally multiclass data of the UCI Machine Learning Repository [4]. Recordings of various speakers pronouncing each vowel were made. Features in the recordings were used as the attributes of an instance. The vowels under question are shown in table 16. An ASCII approximation to the International Phonetic Association symbol of the vowel is given, as well as the respective word in which the vowel was recorded.

Vowel	Word
i	heed
I	hid
E	head
A	had
a	hard
Y	hud
O	hod
o	hoard
U	hood
u	who'd
e	heard

Table 16: Vowels used in the vowel dataset

For each label pair we measured the number of times that a classifier abstained on their pairwise ranking. The share of abstentions that each pair took part in is then compared. We were curious to

[4] https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Vowel+Recognition+-+Deterding+Data)

see if the models made most of their abstentions on a restricted amount of label pairs and if we can intuitively follow the models decision. For example, to a human, the vowels in 'head' and 'had' sound more similar than those in 'hid' and 'heard'. It should follow that the attributes in those instances with less distinguishable labels would also be more similar to each other. The model could than have more difficulties ranking these labels and be more inclined to abstain on them.

What we found is that for those models that made a sizable amount of abstentions (CLR under both base classifiers and BR under random forest), the abstentions are relatively uniformly distributed, i.e. most label pairs took part in about 2% of abstentions, with no significant spikes. This includes pairings that for a human should be relatively easy to tell apart.

For CC there were more significant spikes in the distribution, however, that could very well be due to the fact that the number of abstentions is just that much smaller. Under logistic regression, CC made 94 abstentions in total and 12.8% of them were made for the vowel pairs *Y e* and *o e* respectively. 8.5% for the pairs *E a*, *E U* and *U e* each.

Of course, artificial intelligence does not always follow human intuition and this experiment shows that. In the end, no significant difference in the likelihood for abstentions regarding the label pair can be noted.

12 Conclusion

We have shown that modeling a label ranking as a binary bipartition and solving the label ranking problem in the multilabel classification domain is an entirely feasible and promising approach. Through this way, we derive a label ranking model that can easily incorporate partial rankings into its training data and also produce abstentions in its output. These facts are independent of the utilized multilabel classifier.

Our method was often able to improve the correctness at the cost of completeness in comparison to RPC and LR-RF. It also compared favorably to a partial ranking method derived by Cheng et al. [1] that is implemented on the RPC model. The behavior as to how many abstentions are made and how large the share of correct decisions is varied noticeably between our three tested classifiers.

We observed how classifier chain is the most stable when it comes to producing a maximally complete ranking. We can trace that back to the algorithms ability to learn some of the label dependencies. On complete training data, where the label vectors are mostly an alternation of zeros and ones, the classifier will also learn to mimic such rankings.

On the contrary, we showed how calibrated label ranking is the most abstention prone learning algorithm of the tested ones. Not only did it produce the most abstentions on complete training data, the learner collapsed once the training data contained a fair share of abstentions itself, failing to produce a reasonably complete ranking. However, CLR also outperformed the other learners with respect to correctness on many datasets, especially under random forest. Because of that, if the training data is complete, the classifier is certainly worth a try. But if it is not, we would strongly advise against using it.

Binary relevance tends to make a few more abstentions than classifier chain, but generally also attains a better correctness. It is not as abstention friendly as CLR though, and can still produce rankings under incomplete training information.

In section 11.1 we learned how random forest as the base classifier outperformed logistic regression on most datasets. Additionally we could show how the completeness of the output can be improved by increasing the number of trees in the forest, at the cost of running time and memory demand. Comparable parametrization achievements could not be scored for logistic regression. By varying the ridge in the regressor, one could improve the performance depending on the dataset. As a first option, we would recommend random forest due to the better scores and adaptability. Of course our approach could be tested with different base classifiers as well.

A potential shortcoming of our implementation is that the produced rankings can be cyclic. In section 11.5 we learned how this hindered our implementation of transitivity by effectively reducing the com-

pleteness of the ranking. Either a different evaluation measure or the enforcement of acyclic rankings could be a remedy for this issue. Exploring these options would be a great opportunity for future work.

13 Appendix

We model the multilabel vector so that for a label pair l_i, l_j with $0 < i < j \leq m$, the $l_i \succ l_j$- and $l_j \succ l_i$-labels of the vector are situated next to each other, in that particular order. Furthermore, all pairwise preferences that include l_1 are modeled first (from left to right) and in ascending order. That means l_1, l_2 comes first, then l_1, l_3 and so on. After that we proceed with the remaining label pairs that include l_2 and so on. A slightly larger example that exemplifies the multilabel vector for a ranking over 4 labels is given:

$$l_2 \succ l_4 \succ l_1 \succ l_3$$

$l_1 \succ l_2$,	$l_2 \succ l_1$,	$l_1 \succ l_3$,	$l_3 \succ l_1$,	$l_1 \succ l_4$,	$l_4 \succ l_1$,	$l_2 \succ l_3$,	$l_3 \succ l_2$,	$l_2 \succ l_4$,	$l_4 \succ l_2$,	$l_3 \succ l_4$,	$l_4 \succ l_3$
0,	1,	1,	0,	0,	1,	1,	0,	1,	0,	0,	1

The algorithm that we use to model the vector looks as follows.

Algorithm 1 Label Ranking to Multilabel Vector

```
vectorLength = labelCount*(labelCount - 1)        ▷ labelCount is the original number of labels
multilabelVector = Array[vectorLength]
for each label in labels do                       ▷ labels start with index 1
    for each larger label in labels do
        index = 0                                 ▷ find the index for each label pair
        for i = 0 to label-1 do
            index += (labelCount-1-i)*2
        end for
        index += (larger label-label-1)*2
        if rankOf(label) < rankOf(larger label) then
            multilabelVector[index] = 1
        else if rankOf(label) > rankOf(larger label) then
            multilabelVector[index+1] = 1
        end if
    end for
end for
```

References

[1] Weiwei Cheng, Michaël Rademaker, Bernard De Baets, Eyke Hüllermeier,
Predicting Partial Orders: Ranking with Abstention,
In Proceedings of ECMLPKDD 2010: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pages 215-230, Springer, 2010

[2] Jesse Read, Bernhard Pfahringer, Geoff Holmes, Eibe Frank,
Classifier Chains for Multi-label Classification,
In Proceedings of ECMLPKDD 2009: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pages 254-269, Springer, 2009

[3] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, Klaus Brinker,
Multilabel classification via calibrated label ranking,
In Machine Learning, Volume 73, Issue 2, pages 133-153, Springer, 2008

[4] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, Ioannis Vlahavas,
Mulan: A Java Library for Multi-label Learning,
In Journal of Machine Learning Research, Volume 12, pages 2411-2414, July 2011

[5] Weiwei Cheng, Eyke Hüllermeier, Willem Waegeman, Volkmar Welker,
Label Ranking with Partial Abstention based on Thresholded Probabilistic Models,
In NIPS 2012: Neural Information Processing Systems 25, pages 2510-2518, 2012

[6] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, Klaus Brinker,
Label ranking by learning pairwise preferences,
In Artificial Intelligence, Volume 172, Issues 16-17, pages 1897-1916, November 2008

[7] Weiwei Cheng, Jens Hühn, Eyke Hüllermeier,
Decision Tree and Instance-Based Learning for Label Ranking,
In Proceedings of ICML 2009: 26th Annual International Conference on Machine Learning, pages 161-168, 2009

[8] Yangming Zhou, Guoping Qiu,
Random forest for label ranking,
In Expert Systems with Applications, Volume 112, pages 99-109, 2018

[9] Leo Breiman,
Random Forests,
In Machine Learning, Volume 45, Issue 1, pages 5-32, Springer, October 2001