Chapter 4

Label Ranking Forests

Cláudio Rebelo de Sá, Carlos Soares, Arno Knobbe, Paulo Cortez

in Expert Systems Journal, 2016

Abstract

The problem of Label Ranking is receiving increasing attention from several research communities. The algorithms that have been developed/adapted to treat rankings of a fixed set of labels as the target object, include several different types of decision trees (DT). One DT-based algorithm, which has been very successful in other tasks but which has not been adapted for label ranking is the Random Forests (RF) algorithm. RFs are an ensemble learning method that combines different trees obtained using different randomization techniques. In this work, we propose an ensemble of decision trees for Label Ranking, based on Random Forests, which we refer to as Label Ranking Forests (LRF). Two different algorithms that learn DT for label ranking are used to obtain the trees. We then compare and discuss the results of LRF with standalone decision tree approaches. The results indicate that the method is highly competitive.

4.1 Introduction

Label Ranking (LR) is an increasingly popular topic in the machine learning literature [116, 36, 27, 28, 123]. LR studies a problem of learning a mapping from instances to rankings over a finite number of predefined labels. It can be considered a natural generalization of the conventional classification problem, where the goal is to predict a single label instead of a ranking of all the labels [26].

Some application of Label Ranking approaches are [74]: Meta-learning [16], where we try to predict a ranking of a set of algorithms according to the best expected accuracy on a given dataset; Microarray analysis [74] to find patterns in genes from Yeast on five different micro-array experiments (spo, heat, dtt, cold and diau); Image categorization [58] of landscape pictures from several categories (beach, sunset, field, fall foliage, mountain, urban).

There are two main approaches to the problem of LR: methods that transform the ranking problem into multiple binary problems and methods that were developed or adapted to treat the rankings as target objects, without any transformation. An example of the former is the ranking by pairwise comparisons [74]. Examples of algorithms that were adapted to deal with rankings as the target objects include decision trees [120, 26], naive Bayes [6] and k-Nearest Neighbor [17, 26]. Some of the latter adaptations are based on statistical distribution of rankings (e.g., [24]) while others are based on ranking distance measures (e.g., [120, 36]).

Tree-based models have been used in classification [111], regression [20] and also label ranking [120, 26, 35] tasks. These methods are popular for a number of reasons, including how they can clearly express information about the problem, because their structure is relatively easy to interpret even for people without a background in learning algorithms.

In classification, combining the predictive power of an ensemble of trees often comes with significant accuracy improvements [19]. One of the earliest examples of ensemble methods is *bagging* (a contraction of bootstrap-aggregating) [18]. In bagging, an ensemble of trees is generated and each one is learned on a random selection of examples from the training set. A popular ensemble method is Random Forests [19] which combines different randomization techniques.

Considering the success of Random Forests in terms of improved accuracy for classification and regression problems [13], some approaches have been proposed to deal with different targets, such as *bipartite rankings* [30]. Label

Ranking Forests should also be seen as a potential robust approach for LR. Adapting RF to Label Ranking can be a straightforward process once you have adapted decision trees.

In this work, we propose an approach of ensemble learners which we refer to as Label Ranking Forests (LRF). The proposed method is a natural adaptation of Random Forests for LR, combining the task-independent RF algorithm with the traditional algorithm for top-down induction of decision trees adapted for label ranking. The available adaptations of decision tree algorithms for LR include Label Ranking Trees (LRT) [26], Ranking Trees [115] and Entropy-based Ranking Trees [35]. Considering that the set of trees, in most cases, predict distinct rankings, one should also take into account ranking aggregation methods.

This paper extends previous work [35], in which we proposed a new version of decision trees for LR, called the Entropy-based Ranking Trees and empirically compared them to existing approaches. The main contribution in this paper is the new Label Ranking Forests algorithm, which is an adaptation of the RF ensemble method, using Entropy-based Ranking Trees as the base level algorithm. The results indicate that LRF are competitive with state of the art methods and improve the accuracy of standalone decision trees. An additional contribution is an extension of the original experimental study on Entropy-based Ranking Trees, by analyzing model complexity.

4.2 Label Ranking

In this section, we start by formalizing the problem of label ranking (Section 4.2.1) and then we discuss the adaptation of the decision trees algorithm for label ranking (Section 4.2.2) and one such adaptation, Entropy Ranking Trees (Section 4.2.3).

4.2.1 Formalization

The Label Ranking (LR) task is similar to classification. In classification, given an instance x from the instance space $\mathbb{X}$, the goal is to predict the label (or class) λ to which x belongs, from a predefined set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$. In LR, the goal is to predict the ranking of the labels in $\mathcal{L}$ that is associated with x [74]. A ranking can be represented as a total order over $\mathcal{L}$ defined on the permutation space Ω. A total order can be seen as a permutation π of the set $\{1,\ldots,k\}$, such that $\pi(a)$ is the position of λ_a in π.

As in classification, we do not assume the existence of a deterministic $\mathbb{X} \to \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω [26]. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $\mathcal{P}(\cdot|x)$ such that, for every $\pi \in \Omega$, $\mathcal{P}(\pi|x)$ is the probability that π is the ranking associated with x. The goal in LR is to learn the mapping $\mathbb{X} \to \Omega$. The training data is a set of instances $D = \{\langle x_i, \pi_i \rangle\}, i = 1,\ldots,n$, where x_i is a vector containing the values $x_i^j, j = 1,\ldots,m$ of m independent variables describing instance i and π_i is the corresponding target ranking.

Given an instance x_i with label ranking π_i, and the ranking $\hat{\pi}_i$ predicted by an LR model, we can evaluate the accuracy of the prediction with loss functions on Ω. Some of these measures are based in the number of discordant label pairs:

$$\mathcal{D}(\pi,\hat{\pi}) = \#\{(a,b)|\pi(a) > \pi(b) \wedge \hat{\pi}(a) < \hat{\pi}(b)\}$$

If normalized to the interval $[-1,1]$, this function is equivalent to Kendall's τ coefficient, which is a correlation measure where $\mathcal{D}(\pi,\pi) = 1$ and $\mathcal{D}(\pi,\pi^{-1}) = -1$, where π^{-1} denotes the inverse order of π (e.g. $\pi = (1,2,3,4)$ and $\pi^{-1} = (4,3,2,1)$).

The accuracy of a model can be estimated by averaging this coefficient over a set of examples. Other correlation measures, such as Spearman's rank correlation coefficient [118], have also been used [17]. Although we assume total orders, it may be the case that two labels are tied in the same rank (i.e. $\pi_i(a) = \pi_i(b), a \neq b$). In this case, a variation of Kendall's τ, the $tau - b$ [5] can be used.

4.2.2 Ranking Trees

Tree-based models have been used in classification [111], regression [20], and label ranking [120, 26, 35] tasks. These methods are popular for a number of reasons, including how they can clearly express information about the problem, because their structure is relatively easy to interpret even for people without a background in learning algorithms. It is also possible to obtain information about the importance of the various attributes for the prediction depending on how close to the root they are used.

The Top-Down Induction of Decision Trees (TDIDT) algorithm is commonly used for induction of decision trees [102]. It is a recursive partitioning algorithm that iteratively splits data into smaller subsets which are increasingly more homogeneous in terms of the target variable (Algorithm 3). A split is a test on one of the attributes that divides the dataset into two disjoint subsets. For instance, given a numerical attribute x^2, a split could be $x^2 \geq 5$. Given a splitting criterion that represents the gain in purity obtained with a split, the algorithm chooses the split that optimizes its value in each iteration. In its simplest form, the TDIDT algorithm only stops when the nodes are pure, i.e., when the value of the target attribute is the same for all examples in the node. This usually causes the algorithm to overfit, i.e., to generate models that capture the noise in the data, as well as the regularities that are of general usefulness. One approach to address this problem is to introduce a stopping criterion in the algorithm that tests whether the best split is significantly improving the quality of the model. If not, the algorithm stops and returns a leaf node. The algorithm is executed recursively for the subsets of the data obtained based on the best split until the stopping criterion is met. A leaf node is represented by a value of the target attribute generated by a rule that solves potential conflicts in the set of training examples that are in the node. That value is the prediction that will be made for new examples that fall into that node. In classification, the prediction rule is usually the most frequent class among the training examples.

Algorithm 3 TDIDT algorithm

```
Input: Dataset $D$
BestSplit = Test of the attributes that optimizes the SPLITTING CRITERION
if STOPPING CRITERION == TRUE then
    Determine leaf prediction based on the target values in $D$
    Return a leaf node with the corresponding LEAF PREDICTION
else
    LeftSubtree = TDIDT($D_{\neg BestSplit}$)
    RightSubtree = TDIDT($D_{BestSplit}$)
end if
```

The adaptation of this algorithm for label ranking involves an appropriate choice of the splitting criterion, stopping criterion and the prediction rule (Algorithm 3).

Splitting Criterion The splitting criterion is a measure that quantifies the quality of a given partition of the data. It is usually applied to all the

Table 4.1: Illustration of the splitting criterion

Attribute	Condition=true		Condition=false	
	values	rank corr.	values	rank corr.
x^1	a	0.3	$\{b, c\}$	-0.2
	b	0.2	$\{a, c\}$	0.1
	c	0.5	$\{a, b\}$	0.2
x^2	< 5	-0.1	≥ 5	0.1

possible splits of the data that can be made with tests on the values of individual attributes.

In Ranking Trees (RT) the goal is to obtain leaf nodes that contain examples with target rankings as similar between themselves as possible. To assess the similarity between the rankings of a set of training examples, the mean correlation between them is calculated using Kendall, Spearman or any other ranking correlation coefficient. The quality of the split is given by the weighted mean correlation of the values obtained for the subsets, where the weight is given by the number of examples in each subset.

For simplicity, if we ignore the weights, the splitting criterion of ranking trees is illustrated both for nominal and numerical attributes in Table 4.1. The nominal attribute x^1 has three values (a, b and c). Therefore, three binary splits are possible. For the numerical attribute x^2, a split can be made in between every pair of consecutive values. In this case, the best split is $x^1 = c$, with a mean correlation of 0.5, in comparison to a mean correlation of 0.2 for the remaining, i.e., the training examples for which $x^1 = \{a, b\}$.

Stopping Criterion The stopping criterion is used to determine if it is worthwhile to make a split or if there is a significant risk of overfitting [102]. A split should only be made if the similarity between examples in the subsets increases substantially. Let $\mathcal{S}_{parent}$ be the similarity between the examples in the parent node and $\mathcal{S}_{split}$ the weighted mean similarity in the subsets obtained with the best split. The stopping criterion is defined as follows [115]:

$$(1 + \mathcal{S}_{parent}) \geq \gamma(1 + \mathcal{S}_{split}) \tag{4.1}$$

Note that the relevance of the increase in similarity is controlled by the γ parameter. A $\gamma \geq 1$ does not ensure increased purity of child nodes. On the other hand, small γ values require splits with very large increase in purity, which means that the algorithm will stop the recursion early.

Table 4.2: Illustration of the prediction rule

	λ_1	λ_2	λ_3	λ_4
π_1	1	3	2	4
π_2	2	1	4	3
$\overline{\pi}$	1.5	2	3	3.5
$\hat{\pi}$	1	2	3	4

Prediction Rule The prediction rule is a method to generate a prediction from the (possibly conflicting) target values of the training examples in a leaf node. In LR, the aggregation of rankings is not so straightforward as in other tasks (e.g. classification or regression) and is known as the *ranking aggregation problem* [126]. It is a classical problem in social choice literature [31] but also in information retrieval tasks [51]. A *consensus ranking* minimizes the distance to all rankings [84]. A simple approach, which we adopted in this work, is to compute the average ranking [17] of the predictions. It is calculated by averaging the rank for each label λ_j, $\overline{\pi}(j) = \sum_i \pi_i(j)/n$. The predicted ranking $\hat{\pi}$ is the ranking $\overline{\pi}$ of the labels λ_j obtained based on the average ranks $\overline{\pi}(j)$. Table 4.2 illustrates the prediction rule used in this work.

4.2.3 Entropy Ranking Trees

Recently, we proposed an alternative approach to decision trees for ranking data, the Entropy-based Ranking Trees (ERT) [35]. ERT uses an adaptation of Information Gain (IG) [39] to assess the splitting points and the Minimum Description Length Principle Cut (MDLPC) [54] as the stopping criterion. To explain this method, we start by presenting the IG for rankings measure and then the adapted splitting and stopping criteria.

Decision trees for classification, such as ID3 [111], use Information Gain (IG) as a splitting criterion to determine the best split points. IG is a statistical property that measures the gain in entropy, between the prior and actual state [102]. In this case, we measure it in terms of the distribution of the target variable, before and after the split. In other words, considering a set S of size n_S, since entropy, H, is a measure of disorder, IG is basically how much uncertainty in S is eliminated after splitting on a numerical attribute x^a:

$$IG\left(x^a, T; S\right) = H\left(S\right) - \frac{|S_1|}{n_S} H\left(S_1\right) - \frac{|S_2|}{n_S} H\left(S_2\right)$$

where $|S_1|$ and $|S_2|$ are the number of instances on the left side (S_1) and the number of instances on the right side (S_2), respectively, of the cut point T in attribute x^a.

In cases where S is a set of rankings, we can use the entropy for rankings [39] which is defined as:

$$H_{ranking}(S) = \sum_{i=1}^{K} P(\pi_i, S)\, log\,(P(\pi_i, S))\, log\left(\overline{kt}(S)\right) \tag{4.2}$$

where $P(\pi_i, S)$ is the proportion of rankings equal to π_i in S, K is the number of distinct rankings in S and $\overline{kt}(S)$ is the average normalized Kendall τ [85] distance in the subset S:

$$\overline{kt}(S) = \frac{\sum_{i=1}^{K}\sum_{j=1}^{n}\frac{\tau(\pi_i,\pi_j)+1}{2}}{K \times n_S}.$$

As in Section 4.2.2, the leaves of the tree should not be forced to be pure. Instead, a stopping criterion should be used to avoid overfitting and be robust to noise in rankings. Given an entropy measure, the adaptation of the splitting and stopping criteria comes in a natural way. As shown in [39], the *MDLPC Criterion* can be used as a splitting criterion with the adapted version of entropy $H_{ranking}$. This entropy measure also works with partial orders, however, in this work, we only use total orders.

4.3 Random Forests

Random Forests (RF) [19] are an ensemble method originally proposed for classification and regression problems. It essentially consists of the generation of multiple decision trees obtained using different randomization techniques. The set of predictions made by each of these trees is then aggregated to obtain the prediction of the ensemble.

The RF algorithm is related to another popular ensemble method by the same author, Bagging [18], which stands for *bootstrap-aggregating*. This is an ensemble method that takes a predefined number s of samples (without replacement) from the training data to construct s models. Given a new example, s predictions are generated, which are then aggregated, usually with average or mode, to obtain a combined prediction.

RF can be regarded as an extension of bagging. Given a forest size s and a training dataset D, a set of bootstrap samples, $\{D'_1 \dots, D'_s\}$ is generated by sampling with repetition from D. A decision tree is learned from each $\{D'_1 \dots, D'_s\}$, which is grown in a slightly different way from the original. At each node, only a random subset of the m features can be used for splitting. In classification, the number of random features used in each split is usually $\sqrt{m}$ and in regression $\log_2 m$. This results in what is usually referred to as *random trees.* As in bagging, each of the s random trees makes predictions on the test data, which are then combined using a suitable aggregation method.

One of the reasons for the popularity of RF lays in the fact that they have few parameters to tune and can be applied to various tasks [117]. They require a simple implementation and even with small sample sizes it usually gives accurate results. Moreover, considering that it uses s independent learners, it can be parallelized.

One of the reasons that makes RF a popular approach is that it is possible to take advantage of the algorithm to assess variable importance [61].

4.3.1 Label Ranking Forests

Considering the success of Random Forests in terms of improved accuracy for classification and regression problems, some approaches have been proposed to deal with different targets, such as *bipartite rankings* [30]. Label Ranking Forests should also be seen as a potential robust approach for LR. Adapting RF to Label Ranking can be a straightforward process once you have adapted decision trees.

Thus, we propose a new ensemble LR algorithm, the Label Ranking Forests based on Random Forests. With this approach, we expect to increase the accuracy of Label Ranking tree methods.

In classification and regression, the aggregation of predictions is done in a simple way, mode and mean, respectively. However, as discussed in Section 4.2.2, the aggregation of rankings is not so straightforward. Like in Ranking Trees, we use the average ranking [17] to aggregate the predictions.

Given the similarity of the LR task to classification, the number of random subset features we use in each split is $\sqrt{m}$, the same value that is used in RF for classification.

When the algorithm is not able to find a good split on any of the $\sqrt{m}$ selected features for the root node, it looks for a split on all the m features instead. This prevents the random feature selection mechanism from generating empty trees.

4.4 Empirical Study

In this section we describe the empirical study to investigate the performance of LRF and the tree methods used at the base level. We start by describing the experimental setup (Section 4.4.1), then the results of the base-level algorithms (Section 4.4.2) and finally the results of the new algorithm (Section 4.4.3).

4.4.1 Experimental setup

The experiments are carried out on datasets from the KEBI Data Repository at the Philipps University of Marburg [26] that are typically used in LR research (Table 4.3). They are based on classification and regression datasets, obtained using two different transformation methods: A) the target ranking is a permutation of the classes of the original target attribute, derived from the probabilities generated by a Naive Bayes classifier; B) the target ranking is derived for each example from the order of the values of a set of numerical variables, which are then no longer used as independent variables. A few basic statistics of the datasets used in our experiments are presented in Table 4.3. Although these are somewhat artificial datasets, they are quite useful as benchmarks for LR algorithms.

A simple measure of the diversity of the target rankings is the *Unique Ranking's Proportion*, U_π. U_π is the proportion of distinct target rankings for a given dataset (Table 4.3). As a practical example, the *iris* dataset has 5 distinct rankings for 150 instances, which yields $U_\pi = \frac{5}{150} \approx 3\%$. This means that all the 150 rankings are duplicates of these 5.

The code for all the experiments presented in this paper has been written in R [113].[1]

The generalization performance of the LR methods was estimated using a methodology that has been used previously for this purpose [74]. The eval-

[1]The code is available at `https://github.com/rebelosa/labelrankingforests`.

Table 4.3: Summary of the KEBI datasets

Datasets	type	#examples	#labels	#attributes	U_π
autorship	A	841	4	70	2%
bodyfat	B	252	7	7	94%
calhousing	B	20,640	4	4	0.1%
cpu-small	B	8,192	5	6	1%
elevators	B	16,599	9	9	1%
fried	B	40,769	5	9	0.3%
glass	A	214	6	9	14%
housing	B	506	6	6	22%
iris	A	150	3	4	3%
pendigits	A	10,992	10	16	19%
segment	A	2310	7	18	6%
stock	B	950	5	5	5%
vehicle	A	846	4	18	2%
vowel	A	528	11	10	56%
wine	A	178	3	13	3%
wisconsin	B	194	16	16	100%

uation measure is Kendall's τ and the performance of the methods was estimated using ten-fold cross-validation.

4.4.2 Results with Label Ranking Trees

We evaluate the two variants of ranking trees described earlier: ranking trees (RT) and entropy ranking trees (ERT) (Sections 4.2.2 and 4.2.3). The RT algorithm has a parameter γ, that can affect the accuracy of the model. Based on previous results, we use $\gamma = 0.98$ for RT [35].

Table 4.4 presents the results obtained by the two decision tree approaches, RT and ERT, in comparison to the results for Label Ranking Trees (LRT), that are reproduced from the original paper [26]. We note that we have no information about the depth of the trees obtained with the latter and thus such information is omitted in Table 4.4. Even though LRT performs best in most of the cases presented, both RT and ERT are also competitive methods.

Figure 4.1 shows how much smaller ERT trees are, in general. By generating smaller trees, ERT provides more interpretable models when compared with RT. An exception is the *calhousing* dataset, where ERT generates larger

Table 4.4: Results obtained for Ranking Trees on KEBI datasets (the mean accuracy is represented in terms of Kendall's tau, τ; the best mean accuracy values are in **bold**)

	RT		ERT		LRT
	mean accuracy	depth	mean accuracy	depth	mean accuracy
authorship	.883	8.0	**.889**	4.0	.882
bodyfat	.111	11.9	**.182**	2.7	.117
calhousing	.182	1.0	.291	11.6	**.324**
cpu-small	**.458**	17.2	.437	6.1	.447
elevators	.746	18.9	.757	7.9	**.760**
fried	.797	20.2	.774	13.2	**.890**
glass	.871	8.2	.854	3.0	**.883**
housing	.794	12.9	.704	3.4	**.797**
iris	**.963**	4.3	.853	2.0	.947
pendigits	.871	14.0	.838	5.9	**.935**
segment	.929	12.0	.901	5.0	**.949**
stock	**.897**	10.8	.859	5.0	.895
vehicle	.817	11.0	.787	4.1	**.827**
vowel	**.833**	12.5	.598	3.6	.794
wine	.905	4.0	**.906**	2.0	.882
wisconsin	.334	10.0	.337	2.3	**.343**
average	.712	11.1	.685	5.1	**.730**

trees. However, in this case, the increase in size is justified by a reasonable increase of accuracy (Table 4.4).

To compare different ranking methods, we use a combination of Friedman's test and Dunn's Multiple Comparison Procedure [104], which has been used before for this purpose [17]. First we run the Friedman's test to check whether the results are different or not, with the following hypotheses:

H_0 : The distributions of Kendall's τ are equal

H_1 : The distributions of Kendall's τ are not equal

Using the Friedman test (implemented in the *stats* package [113]) we obtained a p-value $< 1\%$, which shows strong evidence against H_0. This means that there is a high probability that the three methods have different performance.

Thus, we tested which of the three methods are different from one another

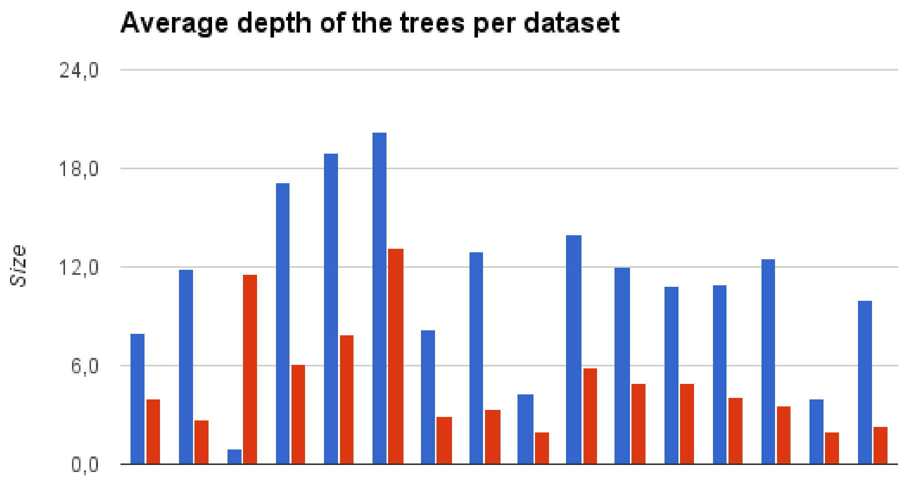

Figure 4.1: Comparison of the average depth of the trees obtained with RT (blue) and ERT (red) on KEBI datasets

with the Dunn's Multiple Comparison Procedure [104]. Using the R package *dunn.test* [45], we tested the following hypotheses for each pair of methods a and b:

H_0 The distributions of Kendall's τ for a and b are equal

H_1 The distributions of Kendall's τ for a and b are not equal

Table 4.5 indicates that there is no statistical evidence that the methods are different. The statistical tests confirm our observation that, although LRT generally obtains better results than RT and ERT, the latter approaches are competitive.

Table 4.5: Dunn's test results (p-values)

	RT	ERT	LRT
RT		0.22	0.37
ERT	0.22		0.13
LRT	0.37	0.13	

4.4.3 Results with Label Ranking Forests

We generated forests with 100 trees and aggregated the predicted rankings with the average ranking method [17]. Table 4.6 presents the results obtained by the Label Ranking Forests using RT and ERT, referred to as LRF-RT and LRF-ERT, respectively.

The average depth of the trees for LRF-RT is, for most cases, smaller than that of the tree obtained with the RT algorithm, while the accuracy is better. On average, for each 0.019 increase in accuracy there was a decrease of 1.8 in the average depth of the trees. One exception is the *elevators* dataset, with suffered a significant decrease in accuracy by using the LRF method.

The comparison between ERT and LRF-ERT leads to different observations. The average depth of the trees increases when using LRF. This can be explained by the fact that the measure of entropy for rankings used in ERT is very robust to noise in rankings [39]. Hence, it requires a larger amount of dissimilarity in a set of rankings to find a partition. As noted in Section 4.4.3 (Figure 4.1) the depth of the trees is much smaller with ERT than with RT.

In Figure 4.2, we can observe how much the accuracy increases/decreases with LRF when compared to the corresponding base-level trees alone. In the vast majority of datasets, there is some improvement in accuracy. The only exception is the *elevators* dataset, as mentioned above.

Using the same statistical tests as before (Section 4.4.2), we compare LRF-RT and LRF-ERT with the RT, ERT and LRT methods. With the Friedman's test we got a p-value $< 1\%$, which shows strong evidence against H_0. Then, now that we know that there are some differences between the 2 methods we will test which are different from one another with the Dunn's Multiple Comparison Procedure (Table 4.7). Since we got a p-value around 25%, between LRF-RT and the LRF-ERT, we cannot conclude that there is no statistical evidence that the methods are different.

On the pairwise comparisons of the methods Table 4.8, we measure how many times each method wins, in terms of accuracy. In this analysis, we conclude that Label Ranking Forests using RT give the best results, proving the effectiveness of the approach.

On the other hand, even though LRF-ERT shows some improvement in terms of accuracy relatively to ERT, it did not behave much better than RT or LRT (Table 4.8). Again, this might be caused by the fact that the measure

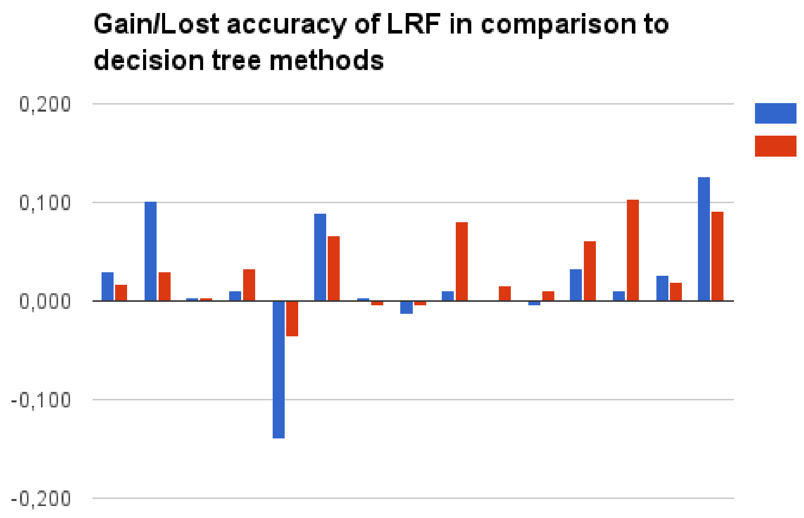

Figure 4.2: Accuracy gained/lost per dataset for using the ensemble method LRF, instead of standalone decision trees RT (blue) and ERT (red) on KEBI datasets

of entropy for rankings used in ERT is very robust to noise. For this reason, the depth of trees in LRF-ERT is, on average, 70% the depth of trees in LRF-RT. While this can be an advantage in terms of Label Ranking Trees, in Label Ranking Forests it is less relevant because it is hard to interpret 100 trees per dataset.

4.5 Conclusions

In this work, we propose an ensemble of decision tree methods for Label Ranking, called Label Ranking Forests (LRF). The method is tested with two different base-level methods Ranking Trees (RT) and Entropy-based Ranking Trees (ERT). We present an empirical evaluation using well known datasets in this field. We also extend the analysis from previous work for tree-based methods, RT and ERT, and compare with the state of the art *Label Ranking Trees* (LRT) approach.

The analysis on the decision trees shows that both RT and ERT are valid and competitive approaches. While RT usually gives better accuracy, on the other

Table 4.6: Results obtained for Label Ranking Forests on KEBI datasets, using two different label ranking trees, RT and ERT (the mean accuracy is represented in terms of Kendall's tau, τ; the best mean accuracy values are in **bold**)

	LRF-RT		LRF-ERT	
	mean accuracy	depth	mean accuracy	depth
authorship	**.912**	8.3	.906	7.7
bodyfat	**.212**	10.6	.211	5.3
calhousing	.185	1.0	**.294**	8.3
cpu-small	.469	13.9	**.471**	7.8
elevators	.605	10.0	**.721**	9.5
fried	**.887**	15.5	.841	14.5
glass	**.874**	6.0	.849	2.7
housing	**.780**	10.9	.699	3.7
iris	**.973**	4.9	.933	2.3
segment	**.930**	10.8	.917	5.2
stock	**.892**	9.9	.869	5.5
vehicle	**.850**	10.0	.849	9.4
vowel	**.844**	11.5	.701	4.9
wine	**.932**	4.3	.925	2.8
wisconsin	**.460**	8.8	.429	3.7
average	**.720**	9.1	.708	6.2

hand, ERT generates trees with much smaller depth (around 50% less, in comparison to RT). Our results were also compared with the published results for Label Ranking Trees (LRT) [26]. LRT has in general better accuracy than RT and ERT, however, statistical tests showed that none of the methods is significantly different. This means that both RT and ERT are competitive approaches, and, since they are distance-based methods, we can also say that this kind of approaches is worth pursuing.

The two ensemble approaches, LRF-RT and LRF-ERT, used the base ranking tree models RT and ERT, respectively. Similarly to the application of Random Forests to other tasks, there was a general increase in accuracy when compared to the corresponding base-level methods. The results confirm that both LRF-RT and LRF-ERT are highly competitive LR methods. LRF-RT, in particular, stands out as a clear winner in terms of accuracy.

As future work, we might improve the comparison with LRT method [26], by implementing it and testing it both as learning algorithm and as the base-level method for Label Ranking Forests. Also, LRF can potentially

Table 4.7: Dunn's test for all the methods (p-values)

	RT	ERT	LRT	LRF-RT	LRF-ERT
RT		0.23	0.34	0.31	0.44
ERT	0.23		0.13	0.11	0.28
LRT	0.34	0.13		0.46	0.29
LRF-RT	0.31	0.11	0.46		0.25
LRF-ERT	0.44	0.28	0.29	0.25	

Table 4.8: Pairwise comparisons of the methods in terms of win statistics.

	RT	ERT	LRT	LRF-RT	LRF-ERT	Total (Rank)
RT		9	6	3	7	25 (4)
ERT	6		3	2	4	15 (5)
LRT	9	12		7	9	37 (2)
LRF-RT	12	13	8		13	46 (1)
LRF-ERT	8	11	6	2		27 (3)

produce similar benefits as the Random Forest method, in terms of feature selection or input variable importance measurement, when applied to LR datasets. Finally, the experiments in this paper were carried out on a set of standard benchmark datasets, which represent artificial LR problems. We plan to apply these approaches on real world datasets e.g. related with user preferences [81].

Acknowledgments

This work is financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project ≪POCI-01-0145-FEDER-006961≫, and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.