Pattern mining for label ranking

Pinho Rebelo de Sá, C.F.

Chapter 2

Preference Rules

Cláudio Rebelo de Sá, Paulo Azevedo, Carlos Soares, Alípio Mário Jorge, Arno Knobbe

submitted to Information Fusion Journal, 2016

Abstract

In this paper we investigate two variants of association rules for preference data, Label Ranking Association Rules and Pairwise Association Rules. Label Ranking Association Rules (LRAR) are the equivalent of Class Association Rules (CAR) for the Label Ranking task. In CAR, the consequent is a single class, to which the example is expected to belong to. In LRAR, the consequent is a ranking of the labels. The generation of LRAR requires special support and confidence measures to assess the similarity of rankings. In this work, we carry out a sensitivity analysis of these similarity-based measures. We want to understand which datasets benefit more from such measures and which parameters have more influence in the accuracy of the model. Furthermore, we propose an alternative type of rules, the Pairwise Association Rules (PAR), which are defined as association rules with a set of pairwise preferences in the consequent. While PAR can be used both as descriptive and predictive models, they are essentially descriptive models. Experimental results show the potential of both approaches.

2.1 Introduction

Label ranking is a topic in the machine learning literature [57, 26, 123] that studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels. One characteristic that clearly distinguishes label ranking problems from classification problems is the order relation between the labels. While a classifier aims at finding the true class on a given unclassified example, the label ranker will focus on the relative preferences between a set of labels/classes. These relations represent relevant information from a decision support perspective, with possible applications in various fields such as elections, dominance of certain species over the others, user preferences, etc.

Due to its intuitive representation, Association Rules [4] have become very popular in data mining and machine learning tasks (e.g. Mining rankings [70], Classification [97] and even Label Ranking [36], etc). The adaptation of AR for label ranking, Label Ranking Association Rules (LRAR) [36], are similar to their classification counterpart, Class Association Rules (CAR) [97]. LRAR can be used for predictive or descriptive purposes.

LRAR are relations, like typical association rules, between an antecedent and a consequent ($A \rightarrow C$), defined by interest measures. The distinction lies in the fact that the consequent is a complete ranking. Because the degree of similarity between rankings can vary, it lead to several interesting challenges. For instance, how to treat rankings that are very similar but not exactly equal. To tackle this problem, similarity-based interest measures were defined to evaluate LRAR. Such measures can be applied to existing rule generation methods [36] (e.g. APRIORI [4]).

One important issue for the use of LRAR is the threshold that determines what should and should not be considered sufficiently similar. Here we present the results of sensitivity analysis study to show how LRAR behave in different scenarios, to understand the effect of this threshold better. Whether there is a rule of thumb or this threshold is data-specific is the type of questions we investigate here. Ultimately we also want to understand which parameters have more influence in the predictive accuracy of the method.

Another important issue is related to the large number of distinct rankings. Despite the existence of many competitive approaches in Label Ranking, Decision trees [120, 26], k-Nearest Neighbor [17, 26] or LRAR [36], problems with a large number of distinct rankings can be hard to predict. One real-world example with a relatively large number of rankings, is the sushi

dataset [81]. This dataset compares demographics of 5000 Japanese citizens with their preferred sushi types. With only 10 labels, it has more than 4900 distinct rankings. Even though it has been known in the preference learning community for a while, no results with high predictive accuracy have been published, to the best of our knowledge. Cases like this have motivated the appearance of new approaches, e.g. to mine ranking data [70], where association rules are used to find patterns within rankings.

We propose a method which combines the two approaches mentioned above [36, 70], because it can could contribute to a better understanding of the datasets mentioned above. We define Pairwise Association Rules (PAR) as association rules with one or more pairwise comparisons in the consequent. In this work we present an approach to identify PAR and analyze the findings in two real world datasets.

By decomposing rankings into the unitary preference relation i.e. *pairwise comparisons*, we can look for sub-ranking patterns. From which, as explained before, we expect to find more frequent patterns than with complete rankings.

LRAR and PARs can be regarded as a specialization of general association rules that are obtained from data containing preferences, which we refer to as *Preference Rules*. These two approaches are complementary in the sense that they can give different insights from preference data. We use LRAR and PAR in this work as predictive and descriptive models, respectively.

The paper is organized as follows: Sections 2.2 and2.3 introduce the task of association rule mining and the label ranking problem, respectively; Section 2.4 describes the Label Ranking Association Rules and Section 2.5 the Pairwise Association Rules proposed here; Section 2.6 presents the experimental setup and discusses the results; finally, Section 2.7 concludes this paper.

2.2 Association Rule Mining

An association rule (AR) is an implication: $A \rightarrow C$ where $A \bigcap C = \emptyset$ and $A, C \subseteq desc(\mathbb{X})$, where $desc(\mathbb{X})$ is the set of descriptors of instances in the instance space $\mathbb{X}$, typically pairs $\langle attribute, value \rangle$. The training data is represented as $D = \{\langle x_i \rangle\}$, $i = 1, \ldots, n$, where x_i is a vector containing the values $x_i^j, j = 1, \ldots, m$ of m independent variables, $\mathcal{A}$, describing instance i. We also denote $desc(x_i)$ as the set of descriptors of instance x_i.

2.2.1 Interest measures

There are many interest measures to evaluate association rules [106], but typically they are characterized by *support* and *confidence*. Here, we summarize some of the most common, assuming a rule $A \rightarrow C$ in D.

Support percentage of the instances in D that contain A and C:

$$sup\,(A \rightarrow C) = \frac{\#\{x_i | A \cup C \subseteq desc(x_i), x_i \in D\}}{n}$$

Confidence percentage of instances that contain C from the set of instances that contain A:

$$conf\,(A \rightarrow C) = \frac{sup\,(A \rightarrow C)}{sup\,(A)}$$

Coverage proportion of examples in D that contain the antecedent of a rule: *coverage* [65]:

$$coverage\,(A \rightarrow C) = sup\,(A)$$

We say that a rule $A \rightarrow C$ covers an instance x, if $A \subseteq desc\,(x)$.

Lift measures the independence of the consequent, C, relative to the antecedent, A:

$$lift\,(A \rightarrow C) = \frac{sup(A \rightarrow C)}{sup(A) \cdot sup(C)}$$

Lift values vary from 0 to $+\infty$. If A is independent from C then $lift(A \rightarrow C) \sim 1$.

2.2.2 APRIORI Algorithm

The original method for induction of AR is the APRIORI algorithm, proposed in 1994 [4]. APRIORI identifies all AR that have support and confidence higher than a given minimal support threshold ($minsup$) and a minimal confidence threshold ($minconf$), respectively. Thus, the model generated is a set of AR, $\mathcal{R}$, of the form $A \rightarrow C$, where $A, C \subseteq desc\,(\mathbb{X})$, and $sup(A \rightarrow C) \geq minsup$ and $conf(A \rightarrow C) \geq minconf$. For a more detailed description see [4].

Despite the usefulness and simplicity of APRIORI, it runs a time consuming candidate generation process and needs substantial time and memory space, proportional to the number of possible combinations of the descriptors. Additionally it needs multiple scans of the data and typically generates a very large number of rules. Because of this, many alternative methods were previously proposed, such as hashing [107], dynamic itemset counting [21], parallel and distributed mining [108] and mining integrated into relational database systems [119].

In contrast to itemset-based algorithms, which compute frequent itemsets and rule generation in two steps, there are rule-based approaches such as FP-Growth (Frequent pattern growth method) [67]. This means that, rules are generated at the same time as frequent itemsets are computed.

2.2.3 Pruning

AR algorithms typically generate a large number of rules (possibly tens of thousands), some of which represent only small variations from others. This is known as the rule explosion problem [80] which should be dealt with by pruning mechanisms. Many rules must be discarded for computational and simplicity reasons.

Pruning methods are usually employed to reduce the amount of rules without reducing the quality of the model. For example, an AR algorithm might find rules for which the confidence is only marginally improved by adding further conditions to their antecedent.Another example is when the consequent C of a rule $A \rightarrow C$ has the same distribution independently of the antecedent A. In these cases, we should not consider these rules as meaningful.

Improvement A common pruning method is based on the improvement that a refined rule yields in comparison to the original one [80]. The *improvement* of a rule is defined as the smallest difference between the confidence of a rule and the confidence of all sub-rules sharing the same consequent:

$$imp(A \rightarrow C) = min(\forall A' \subset A, conf(A \rightarrow C) - conf(A' \rightarrow C))$$

As an example, if one defines a minimum improvement $minImp = 1\%$, the rule $A' \rightarrow C$ will be kept if $conf(A' \rightarrow C) - conf(A \rightarrow C) \geq 1\%$, where $A \subset A'$.

If $imp(A \rightarrow C) > 0$ we say that $A \rightarrow C$ is a productive rule.

Significant rules Another way to prune non productive rules is to use statistical tests [125]. A rule is *significant* if the confidence improvement over all its generalizations is statistically significant. The rule $A \to C$ is significant if $\forall A' \to C, A' \subset A$ the difference $conf(A \to C) - conf(A' \to C)$ is statistically significant for a given significance level (α).

2.3 Label Ranking

In Label Ranking (LR), given an instance x from the instance space $\mathbb{X}$, the goal is to predict the ranking of the labels $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$ associated with x [74]. A ranking can be represented as a *strict total order* over $\mathcal{L}$, defined on the permutation space Ω.

The LR task is similar to the classification task, where instead of a class we want to predict a ranking of labels. 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 contains 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, $\mathcal{A}$, describing instance i and π_i is the corresponding target ranking.

The rankings can be either total or partial orders.

Total orders A *strict total order* over $\mathcal{L}$ is defined as:[1]

$$\{\forall\, (\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succ \lambda_b \vee \lambda_b \succ \lambda_a\}$$

which represents a *strict ranking* [123], a *complete ranking* [57], or simply a *ranking*. A strict total order can also be represented as a permutation π of the set $\{1, \ldots, k\}$, such that $\pi(a)$ is the position, or *rank*, of λ_a in π. For example, the *strict total order* $\lambda_1 \succ \lambda_2 \succ \lambda_3 \succ \lambda_4$ can be represented as $\pi = (1, 2, 3, 4)$.

However, in real-world ranking data, we do not always have clear and unambiguous preferences, i.e. strict total orders [15]. Hence, sometimes we have

[1] For convenience, we say *total order* but in fact we mean a *totally ordered set*. Strictly speaking, a *total order* is a binary relation.

to deal with *indifference* and *incomparability*. For illustration purposes, let us consider the scenario of elections, where a set of n voters vote on k candidates. If a voter feels that two candidates have identical proposals, then these can be expressed as indifferent so they are assigned the same rank (i.e. a tie).

To represent ties, we need a more relaxed setting, called *non-strict total orders*, or simply *total orders*, over $\mathcal{L}$, by replacing the binary strict order relation, $\succ$, with the binary partial order relation, $\succeq$:

$$\{\forall\,(\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succeq \lambda_b \vee \lambda_b \succeq \lambda_a\}$$

These non-strict total orders can represent *partial rankings* (rankings with ties) [123]. For example, the *non-strict total order* $\lambda_1 \succ \lambda_2 = \lambda_3 \succ \lambda_4$ can be represented as $\pi = (1, 2, 2, 3)$.

Additionally, real-world data may lack preference data regarding two or more labels, which is known as *incomparability*. Continuing with the elections example, the lack of information about one or two of the candidates, λ_a and λ_b, leads to incomparability, $\lambda_a \perp \lambda_b$. In other words, the voter cannot decide whether the candidates are equivalent or select one as the preferred, because he does not know the candidates. Incomparability should not be confused with intrinsic properties of the objects, as if we are comparing apples and oranges. Instead, it is like trying to compare two different types of apple without ever having tried either. In this cases, we can use *partial orders*.

Partial orders Similarly to *total orders*, there are *strict* and *non-strict partial orders*. Let us consider the *non-strict partial orders* (which can also be referred to as *partial orders*) over $\mathcal{L}$:

$$\{\forall\,(\lambda_a, \lambda_b) \in \mathcal{L} | \lambda_a \succeq \lambda_b \vee \lambda_b \succeq \lambda_a \ \vee \lambda_a \perp \lambda_b\}$$

We can represent partial orders with *subrankings* [70]. For example, the *partial order* $\lambda_1 \succ \lambda_2 \succ \lambda_4$ can be represented as $\pi = (1, 2, 0, 4)$, where 0 represents $\lambda_1, \lambda_2, \lambda_4 \perp \lambda_3$.

2.3.1 Methods

Several learning algorithms were proposed for modeling label ranking data in recent years. These can be grouped as decomposition-based or direct.

Decomposition-based methods divide the problem into several simpler problems (e.g., multiple binary problems). An example is ranking by pairwise comparisons [57] and mining rank data [70]. *Direct methods* treat the rankings as target objects without any decomposition. Examples of that include decision trees [120, 26], k-Nearest Neighbors [17, 26] and the linear utility transformation [68, 41]. This second group of algorithms can be divided into two approaches. The first one contains methods that are based on statistical distributions of rankings (e.g. [26]), such as Mallows [91], or Plackett-Luce [24]. The other group of methods are based on measures of similarity or correlation between rankings (e.g. [120, 6]).

LR-specific preprocessing methods have also been proposed, e.g. MDLP-R [40] and EDiRa [39]. Both are *direct methods* and based on measures of similarity. Considering that supervised discretization approaches usually provide better results than unsupervised methods [46], such methods can be of a great importance in the field. In particular, for AR-like algorithms, such as the ones proposed in this work, which are typically not suitable for numerical data.

For more information on label ranking learning methods, more information ca be found in [57].

Label Ranking by Learning Pairwise Preferences

Ranking by pairwise comparisons basically consists of reducing the problem of ranking into several classification problems. In the learning phase, the original problem is formulated as a set of pairwise preferences problem. Each problem is concerned with one pair of labels of the ranking, $(\lambda_i, \lambda_j) \in \mathcal{L}, 1 \leq i < j \leq k$. The target attribute is the relative order between them, $\lambda_i \succ \lambda_j$. Then, a separate model $\mathcal{M}_{ij}$ is obtained for each pair of labels. Considering $\mathcal{L} = \{\lambda_1, \ldots, \lambda_k\}$, there will be $h = \frac{k(k-1)}{2}$ classification problems to model.

In the prediction phase, each model is applied to every pair of labels to obtain a prediction of their relative order. The predictions are then combined to derive rankings, which can be done in several ways. The simplest is to order the labels, for each example, considering the predictions of the models $\mathcal{M}_{ij}$ as votes. This topic has been well studied and documented [55, 74].

2.3.2 Evaluation

Given an instance x_i with label ranking π_i and a ranking $\hat{\pi}_i$ predicted by a LR model, several loss functions on Ω can be used to evaluate the accuracy of the prediction. One such function is 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 there are no discordant label pairs, the distance $\mathcal{D} = 0$. Alternatively, the function to define the number of concordant pairs is:

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

Kendall Tau Kendall's τ coefficient [85] is the normalized difference between the number of concordant, $\mathcal{C}$, and discordant pairs, $\mathcal{D}$:

$$\tau(\pi, \hat{\pi}) = \frac{\mathcal{C} - \mathcal{D}}{\frac{1}{2}k(k-1)}$$

where $\frac{1}{2}k(k-1)$ is the number of possible pairwise combinations, $\binom{k}{2}$. The values of this coefficient range from $[-1, 1]$, where $\tau(\pi, \pi) = 1$ if the rankings are equal and $\tau(\pi, \pi^{-1}) = -1$ if π^{-1} denotes the inverse order of π (e.g. $\pi = (1, 2, 3, 4)$ and $\pi^{-1} = (4, 3, 2, 1)$). Kendall's τ can also be computed in the presence of ties, using tau-b [5].

An alternative measure is the Spearman's rank correlation coefficient [118].

Gamma coefficient If we want to measure the correlation between two partial orders (subrankings), or between total and partial orders, we can use the Gamma coefficient [93]:

$$\gamma(\pi, \hat{\pi}) = \frac{\mathcal{C} - \mathcal{D}}{\mathcal{C} + \mathcal{D}}$$

Which is identical to Kendall's τ coefficient in the presence of strict total orders, because $\mathcal{C} + \mathcal{D} = \frac{1}{2}k(k-1)$.

Weighted rank correlation measures When it is important to give more relevance to higher ranks, a weighted rank correlation coefficient can be used. They are typically adaptations of existing similarity measures, such as ρ_w [110], which is based on Spearman's coefficient.

These correlation measures are not only used for evaluation estimation, they can be used within learning [36] or preprocessing [39] models. Since Kendall's τ has been used for evaluation in many recent LR studies [26, 40], we use it here as well.

The accuracy of a label ranker can be estimated by averaging the values of any of the measures explained here, over the rankings predicted for a set of test examples. Given a dataset, $D = \{\langle x_i, \pi_i \rangle\}$, $i = 1, \ldots, n$, the usual resampling strategies, such as holdout or cross-validation, can be used to estimate the accuracy of a LR algorithm.

2.4 Label Ranking Association Rules

Association rules were originally proposed for descriptive purposes. However, they have been adapted for predictive tasks such as classification (e.g., [97]). Given that label ranking is a predictive task, the adaptation of AR for label ranking comes in a natural way. A *Label Ranking Association Rule* (LRAR) [36] is defined as:

$$A \rightarrow \pi$$

where $A \subseteq desc\,(\mathbb{X})$ and $\pi \in \Omega$. Let $\mathcal{R}_\pi$ be the set of *label ranking association rules* generated from a given dataset. When an instance x is covered by the rule $A \rightarrow \pi$, the predicted ranking is π. A rule $r_\pi : A \rightarrow \pi, r_\pi \in \mathcal{R}_\pi$, covers an instance x, if $A \subseteq desc(x)$.

We can use the CAR framework[97] for LRAR. However this approach has two important problems. First, the number of classes can be extremely large, up to a maximum of $k!$, where k is the size of the set of labels, $\mathcal{L}$. This means that the amount of data required to learn a reasonable mapping $\mathbb{X} \rightarrow \Omega$ is unreasonably large.

The second disadvantage is that this approach does not take into account the differences in nature between label rankings and classes. In classification, two examples either have the same class or not. In this regard, label ranking is more similar to regression than to classification. In regression, a large number of observations with a given target value, say 5.3, increases the probability of observing similar values, say 5.4 or 5.2, but not so much for very different values, say -3.1 or 100.2. This property must be taken into account in the induction of prediction models. A similar reasoning can be made in label ranking. Let us consider the case of a data set in which

ranking $\pi_a = (1, 2, 3, 4)$ occurs in 1% of the examples. Treating rankings as classes would mean that $P(\pi_a) = 0.01$. Let us further consider that the rankings $\pi_b = (1, 2, 4, 3), \pi_c = (1, 3, 2, 4)$ and $\pi_d = (2, 1, 3, 4)$, which are obtained from π_a by swapping a single pair of adjacent labels, occur in 50% of the examples. Taking into account the stochastic nature of these rankings [26], $P(\pi_a) = 0.01$ seems to underestimate the probability of observing π_a. In other words it is expected that the observation of π_b, π_c and π_d increases the probability of observing π_a and vice-versa, because they are similar to each other.

This affects even rankings which are not observed in the available data. For example, even though a ranking is not present in the dataset it would not be entirely unexpected to see it in future data. This also means that it is possible to compute the probability of unseen rankings.

To take all this into account, similarity-based interestingness measures were proposed to deal with rankings [36].

2.4.1 Interestingness measures in Label Ranking

As mentioned before, because the degree of similarity between rankings can vary, similarity-based measures can be used to evaluate LRAR. These measures are able to distinguish rankings that are *very similar* from rankings that are very *very distinct*. In practice, the measures described below can be applied to existing rule generation methods [36] (e.g. APRIORI [4]).

Support The support of a ranking π should increase with the observation of similar rankings and that variation should be proportional to the similarity. Given a measure of similarity between rankings $s(\pi_a, \pi_b)$, we can adapt the concept of support of the rule $A \rightarrow \pi$ as follows:

$$sup_{lr}(A \rightarrow \pi) = \frac{\sum_{i:A \subseteq desc(x_i)} s(\pi_i, \pi)}{n}$$

Essentially, what we are doing is assigning a weight to each target ranking π_i in the training data that represents its contribution to the probability that π may be observed. Some instances $x_i \in \mathbb{X}$ give a strong contribution to the support count (i.e., 1), while others will give a weaker or even no contribution at all.

Table 2.1: An example of a label ranking dataset.

		π_1	π_2	π_3
TID	$\mathcal{A}_1$	$(1,3,2)$	$(2,1,3)$	$(2,3,1)$
1	L	0.33	0.00	1.00
2	L	0.00	1.00	0.00
3	L	1.00	0.00	0.33

Any function that measures the similarity between two rankings or permutations can be used, such as Kendall's τ [85] or Spearman's ρ [118]. The function used here is of the form:

$$s(\pi_a, \pi_b) = \begin{cases} s'(\pi_a, \pi_b) & \text{if } s'(\pi_a, \pi_b) \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where s' is a similarity function. This general form assumes that below a given threshold, θ, is not useful to discriminate between different rankings, as they are so different from π_a. This means that, the support sup_{lr} of $A \to \pi_a$ will be based only on the items of the form $\langle A, \pi_b \rangle$, for all π_b where $s'(\pi_a, \pi_b) > \theta$).

Many functions can be used as s'. However, given that the loss function we aim to minimize is known beforehand, it makes sense to use it to measure the similarity between rankings. Therefore, we use Kendall's τ as s'.

Concerning the threshold, given that anti-monotonicity can only be guaranteed with non-negative values [109], it implies that $\theta \geq 0$. Therefore we think that $\theta = 0$ is a reasonable default value, because it separates between the positive and negative correlation between rankings.

Table 2.1 shows an example of a label ranking dataset represented according to this approach. Instance $(\{\mathcal{A}_1 = L, \pi_3\})$ (TID=1) contributes to the support count of ruleitem $\langle \{\mathcal{A}_1 = L\}, \pi_3 \rangle$ with 1, as expected. However, that same instance, will also give a contribution of 0.33 to the support count of ruleitem $\langle \{\mathcal{A}_1 = L\}, \pi_1 \rangle$, given their ranking similarity. On the other hand, no contribution to the support of ruleitem $\langle \{\mathcal{A}_1 = L\}, \pi_2 \rangle$ is given, because these rankings are clearly different. This means that $sup_{lr}\left(\langle \{\mathcal{A}_1 = L\}, \pi_3 \rangle\right) = \frac{1+0.33}{3}$.

Confidence The confidence of a rule $A \to \pi$ comes in a natural way if we replace the classical measure of support with the similarity-based sup_{lr}.

$$conf_{lr}\left(A \to \pi\right) = \frac{sup_{lr}\left(A \to \pi\right)}{sup\left(A\right)}$$

Improvement Improvement in association rule mining is defined as the smallest difference between the confidence of a rule and the confidence of all sub-rules sharing the same consequent [80]. In LR it is not suitable to compare targets simply as equal or different (Section 2.4). Therefore, to implement pruning based on improvement for LR, some adaptation is required as well. Given that the relation between target values is different from classification, as discussed in Section 2.4.1, we have to limit the comparison between rules with different consequents, if $S'(\pi, \pi') \geq \theta$.

Improvement for Label Ranking is defined as:

$$imp_{lr}(A \to \pi) = min(conf_{lr}(A \to \pi) - conf_{lr}(A' \to \pi'))$$

for $\forall A' \subset A$, and $\forall (\pi, \pi')$ where $S'(\pi', \pi) \geq \theta$. As an illustrative example, consider the two rules $r_1 : A_1 \to (1, 2, 3, 4)$ and $r_2 : A_2 \to (1, 2, 4, 3)$, where A_2 is a superset of A_1, $A_1 \subset A_2$. If $S'((1, 2, 3, 4), (1, 2, 4, 3)) \geq \theta$ then r_2 will only be kept if, and only if, $conf(r_1) - conf(r_2) \geq minImp$.

Lift The *lift* measures the independence between the consequent and the antecedent of the rule [9]. The adaptation of lift for LRAR is straightforward since it only depends the concept of support, for which a version for LRAR already exists:

$$lift_{lr}(A \to \pi) = \frac{sup_{lr}(A \to \pi)}{sup(A) \cdot sup_{lr}(\pi)}$$

2.4.2 Generation of LRAR

Given the adaptations of the interestingness measures proposed, the task of learning LRAR can be defined essentially in the same way as the task of learning AR, i.e. to identify the set of LRAR that has a support and a confidence higher than the thresholds defined by the user. More formally, given a training set $D = \{\langle x_i, \pi_i \rangle\}, i = 1, \ldots, n$, the algorithm aims to create a set of high accuracy rules $\mathcal{R}_\pi = \{r_\pi : A \to \pi\}$ to cover a test set $T = \{\langle x_j \rangle\}, j = 1, \ldots, s$. If $\mathcal{R}_\pi$ does not cover some $x_j \in T$, a *DefaultRanking* (Section 2.4.3) is assigned to it.

Implementation of LRAR in CAREN

The association rule generator we are using is CAREN [10]. [2] CAREN implements an association rule algorithm to derive rule-based prediction models, like CAR and LRAR. For Label Ranking datasets, CAREN derives association rules where the consequent is a complete ranking.

CAREN is specialized in generating association rules for predictive models and employs a bitwise depth-first frequent pattern mining algorithm. Rule pruning is performed using a Fisher exact test [10]. Like CMAR [95], CAREN is a rule-based algorithm rather than itemset-based. This means that, frequent itemsets are derived at the same time as rules are generated, whereas itemset-based algorithms carry out the two tasks in two separated steps.

Rule-based approaches allow for different pruning methods. For example, let us consider the rule $A \rightarrow \lambda$, where λ is the most frequent class in the examples covering A. If $sup\,(A \rightarrow \lambda) < minsup$ then there is no need to search for a superset of A, A^*, since any rule of the form $A^* \rightarrow \lambda, A \subset A^*$ cannot have a support higher than $minsup$.

CAREN generates significant rules [125]. Statistical significance of a rule is evaluated using a Fisher Exact Test by comparing its support to the support of its direct generalizations. The direct generalizations of a rule $A \rightarrow C$ are $\emptyset \rightarrow C$ and $(A \setminus \{a\}) \rightarrow C$ where a is a single item.

The final set of rules obtained define the label ranking prediction model, which we can also refer as the *label ranker*.

CAREN also employs prediction for strict rankings using *consensus ranking* (Section 2.4.3), best rule, among others.

2.4.3 Prediction

A very straightforward method to generate predictions using a label ranker is used. The set of rules $\mathcal{R}_\pi$ can be represented as an ordered list of rules, by some user defined measure of relevance:

$$< r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_t} >$$

As mentioned before, a rule $r^*_\pi : A^* \rightarrow \pi^*$ covers (or matches) an instance $x_i \in T$, if $A^* \subseteq desc(x_i)$. If only one rule, r^*_π, matches x_i, the predicted ranking

[2] `http://www4.di.uminho.pt/~pja/class/caren.html`

for x_i is π^*. However, in practice, it is quite common to have more than one rule covering the same instance x_i, $\mathcal{R}^*_\pi(x_j) \subseteq \mathcal{R}_\pi$. In $\mathcal{R}^*_\pi(x_j)$ there can be rules with conflicting ranking recommendations. There are several methods to address those conflicts, such as selecting the best rule, calculating the majority ranking, etc. However, it has been shown that a ranking obtained by ordering the average ranks of the labels across all rankings minimizes the euclidean distance to all those rankings [84]. In other words, it maximizes the similarity according to Spearman's ρ [118]. This can be referred to as the *average ranking* [17].

Given any set of rankings $\{\pi_i\}$ $(i = 1, \ldots, s)$ with k labels, we compute the *average ranking* as:

$$\overline{\pi}(j) = \frac{\sum_{i=1}^{s} \pi_i(j)}{s}, j = 1, \ldots, k \tag{2.2}$$

The average ranking $\overline{\pi}$ can be obtained if we rank the values of $\overline{\pi}(j), j = 1, \ldots, k$. A weighted version of this method can be obtained by using the *confidence* or *support* of the rules in $\mathcal{R}^*_\pi(x_j)$ as weights.

Default rules

As in classification, in some cases, the label ranker might not find any rule that covers a given instance x_j, so $\mathcal{R}^*_\pi(x_j) = \emptyset$. To avoid this, we need to define a *default rule*, $r_\emptyset$, which can be used in such cases:

$$\{\emptyset\} \rightarrow \textit{default ranking}$$

A *default class* is also often used in classification tasks [66], which is usually the majority class of the training set D. In a similar way, we could define the majority ranking as our *default ranking*. However, some label ranking datasets have as many rankings as instances, making the majority ranking not so representative.

As mentioned before, the *average ranking* (Equation 2.2) of a set of rankings, minimizes the distance to all rankings in that set [84]. Hence we can use the *average ranking* as the *default ranking*.

2.4.4 Parameter tuning

Due to the intrinsic nature of each different dataset, or even of the preprocessing methods used to prepare the data (e.g., the discretization method), the maximum *minsup*/*minconf* needed to obtain a rule set $\mathcal{R}_\pi$, that covers all the examples, may vary significantly [98]. The trivial solution would be, for example, to set $minconf = 0$ which would generate many rules, hence increasing the coverage. However, this rule would probably lead to a lot of uninteresting rules as well, as the model would overfit the data. Then, our goal is to obtain a rule set $\mathcal{R}_\pi$ which gives maximal coverage while keeping high confidence rules.

Let us define M as the coverage of the model i.e. the coverage of the set of rules $\mathcal{R}_\pi$. Algorithm 1 represents a simple, heuristic method to determine the *minconf* that obtains the rule set such that a certain minimal coverage is guaranteed $minM$.

Algorithm 1 Confidence tuning algorithm

```
Given minsup and step
minconf = 100%
while M < minM do
    minconf = minconf − step
    Run CAREN with (minsup,minconf) and determine M
end while
    return minconf
```

This procedure has the important advantage that it does not take into account the accuracy of the rule sets generated, thus reducing the risk of overfitting.

2.5 Pairwise Association Rules

Association rules use a sets of descriptors to represent meaningful subsets of the data [69], hence providing an easy interpretation of the patterns mined. Due to the intuitive representation, since its first application in the market basket analysis [2], they have become very popular in data mining and machine learning tasks (Mining rankings [70], Classification [97], Label Ranking [36], etc).

LRAR proved to be an effective predictive model, however they are designed to find complete rankings. Despite its similarity measures, which take into account possible ranking noise, it does not capture subranking patterns because it will always try to infer complete rankings. On the other hand, association rules were used to find patterns within rankings [70], however, they do not relate it with the independent variables. Besides, in [70], the consequent is limited to one pairwise comparison.

In this work, we propose a decomposition method to look for meaningful associations between independent variables and preferences (in the form of pairwise comparisons), the Pairwise Association Rules (PAR), which can be regarded as predictive or descriptive model. We define PAR as:

$$A \rightarrow \{\lambda_a \succeq \lambda_b \vee \lambda_a \perp \lambda_b \ \vee \lambda_a = \lambda_b | \lambda_a, \lambda_b \in \mathcal{L}\}$$

where, as in the original AR paper [4], we allow rules with multiple items, not only in the antecedent but also in the consequent, i.e. PAR can have multiple sets of pairwise comparisons in the consequent.

Similarly to RPC (Section 2.3.1), we decompose the target rankings into pairwise comparisons. Therefore, PAR can be obtained from data with strict rankings, partial rankings and subrankings. [3]

Contrary to LRAR, we use the same interestingness measures that are also used in typical AR approaches, instead of the similarity-based versions defined for LR problems, i.e. *sup*, *conf*, etc. This allows PAR to filter out non-frequent/interesting patterns and makes it more difficult to derive strict rankings. When methods cannot find interesting rules with enough pairwise comparisons to define a strict ranking, partial rankings, subrankings or even with sets of disjoint pairwise comparisons can be found. This is, interest measures are defining the borders between what the model will keep or abstain.

Abstention is used in machine learning to describe the option to not make a prediction when the confidence in the output of a model is insufficient. The simplest case is classification, where the model can abstain itself to make a decision [11]. In the label ranking task, a method that makes partial abstentions was proposed in [28]. A similar reasoning is used here both for predictive and descriptive models.

More formally, let us define $D = \{\langle x_i, \pi_i \rangle\}, i = 1, \ldots, n$ where π_i can be a *complete ranking*, *partial ranking* or a *sub-ranking*. For each π of size k we

[3] To derive the PAR, we added a pairwise decomposition method to the CAREN [10] software.

can extract up to h pairwise comparisons. We consider 4 possible outcomes for each pairwise comparison:

- $\lambda_a \succeq \lambda_b$
- $\lambda_b \succeq \lambda_a$
- $\lambda_a = \lambda_b$ (indifference)
- $\lambda_a \perp \lambda_b$ (incomparability)

As an example, a PAR can be of the form:

$$A \rightarrow \lambda_1 \succ \lambda_4 \wedge \lambda_3 \succ \lambda_1 \wedge \lambda_1 \perp \lambda_2$$

The consequent can be simplified into $\lambda_3 \succ \lambda_1 \succ \lambda_4$ or represented as a subranking $\pi = (2, 0, 1, 3)$.

2.6 Experimental Results

In this section we start by describing the datasets used in the experiments, then we introduce the experimental setup and finally present the results obtained.

2.6.1 Datasets

The data sets in this work were taken from KEBI Data Repository in the Philipps University of Marburg [26] (Table 2.2).

To illustrate domain-specific interpretations of the results, we experiment with two additional datasets. We use an adapted dataset from the 1999 COIL Competition [96], Algae [34], concerning the frequencies of algae populations in different environments. The original dataset consisted of 340 examples, each representing measurements of a sample of water from different European rivers on different periods. The measurements include concentrations of chemical substances like nitrogen (in the form of nitrates, nitrites and ammonia), oxygen and chlorine. Also the pH, season, river size and its flow velocity were registered. For each sample, the frequencies of 7 types of algae were also measured. In this work, we considered the algae concentrations as preference relations by ordering them from larger to smaller concentrations. Those with 0 frequency are placed in last position and equal frequencies are

Table 2.2: Summary of the datasets

Datasets	type	#examples	#labels	#attributes	U_π
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%
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%
Algae (COIL)		316	7	10	72%
Sushi		5000	10	10	98%

represented with ties. Missing values in the independent variables were set to 0.

Finally, the Sushi preference dataset [81], which is composed of demographic data about 5000 people and sushi preferences is also used. Each person sorted a set of 10 different sushi types by preference. The 10 types of sushi, are a) shrimp, b) sea eel, c) tuna, d) squid, e) sea urchin, f) salmon roe, g) egg h) fatty tuna, i) tuna roll and j) cucumber roll. Since the attribute names were not transformed in this dataset, we can make a richer analysis of it.

Table 2.2 presents a simple measure of the diversity of the target rankings, the *Unique Ranking's Proportion*, U_π. U_π is the proportion of distinct target rankings for a given dataset. As a practical example, the *iris* dataset has 5 distinct rankings for 150 instances, which results in $U_\pi = \frac{5}{150} \approx 3\%$.

2.6.2 Experimental setup

Continuous variables were discretized with two distinct methods: (1) *Entropy-based Discretization for Ranking data (EDiRa)* ([39]) and (2) *equal width* bins. *EDiRa* is the state of the art supervised discretization method in Label Ranking, while *equal width* is a simple, general method that serves as baseline.

The evaluation measure used in all experiments is Kendall's τ. A ten-fold cross-validation was used to estimate the value for each experiment. The generation of Label Ranking Association Rules (LRAR) and PAR was performed with CAREN [10] which uses a depth-first based approach.

The confidence tuning Algorithm 1 was used to set parameters. We consider that 5% seems a reasonable step value because the *minconf* can be found in, at most, 20 iterations. Given that a common value for the *minsup* in Association Rules (AR) mining is 1%, we use it as default for all datasets. We define the *minM* as 95% to get a reasonable coverage, and $minImp = 1\%$ to avoid rule explosion.

In terms of similarity functions, we use a normalized Kendall τ between the interval $[0, 1]$ as our similarity function s (Equation 2.1).

2.6.3 Results with LRAR

In the experiments described in this section we analyze the performance from different perspectives, *accuracy*, *number of rules* and *average confidence* as the similarity threshold θ varies. We expect to understand the impact of using similarity measures in the generation of LRAR and provide some insights about its usage.

LRAR, despite being based on similarity measures, are consistent with the classical concepts underlying association rules. A special case is when $\theta = 1$, where, as in CAR, only equal rankings are considered. Therefore, by varying the threshold θ we also understand how similarity-based interest measures ($0 \leq \theta < 1$) contribute to the accuracy of the model, in comparison to frequency-based approaches ($\theta = 1$).

We would also like to understand how some properties of the data relate the sensitivity to θ. We can extract two simple measures of ranking diversity from the datasets, the *Unique Ranking's Proportion* (U_π), mentioned before, and the *ranking entropy* [39].

Sensitivity analysis

Here we analyze how the similarity threshold θ affects the accuracy, number and quality (in terms of confidence) of LRAR.

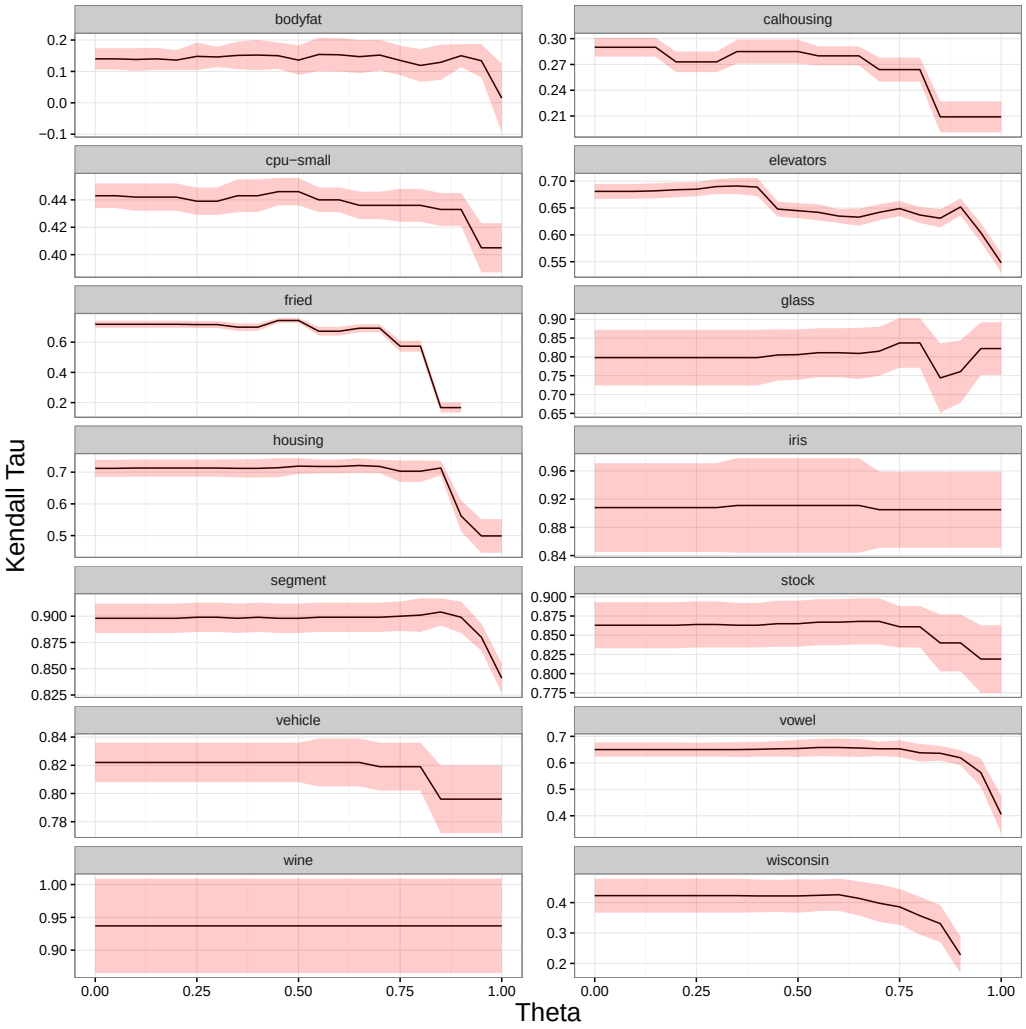

Figure 2.1: Average accuracy (Kendall τ) of CAREN as the θ varies

Accuracy In Figure 2.1 we can see the behavior of the accuracy of CAREN in terms of θ. It shows that, in general, there is a tendency for the accuracy to decrease as θ gets closer to 1. This happens in 12 out of the 14 datasets analyzed. On the other hand, in 9 out of 14 datasets, the accuracy is rather stable in the range $\theta \in [0, 0.6]$.

If we take into consideration that the model ignores all similarities between rankings for $\theta = 1$, the observed behavior seems to favor the similarity-based approach. In line with that, two extreme cases can be seen with *fried* and *wisconsin* datasets, where CAREN was not able to find any LRAR for

$\theta = 1$. [4]

Let us consider the *accuracy range*, the maximum accuracy minus the minimum accuracy. To find out which datasets are more likely to be affected by the choice of θ, we can compare their ranking entropy with the measured *accuracy range* from Figure 2.1. In Figure 2.2 we compare the accuracy range with the *ranking entropy* [39]. We can see that, the higher the entropy, the more the accuracy can be affected by the choice of θ.

Results seem to indicate that, when mining LRAR in datasets with low ranking entropy, the choice of θ is not so relevant. On the other hand, as the entropy gets bigger, a reasonable value should be $0 \leq \theta \leq 0.6$.

One interesting behavior can be found in the dataset *fried*. Despite the fact that it has a very low proportion of unique rankings, $U_\pi\,(\textit{fried}) = 0.3\%$ (Table 2.2) its entropy is quite high (Figure 2.2). For this reason, it makes it more sensitive to θ, as seen in Figure 2.1. On the other hand, *iris* and *wine*, with very low entropy, seem unaffected by θ.

Number of rules Ideally, we would like to obtain a small number of rules with high accuracy. However, such a balance is not expected to happen frequently. Ultimately, as accuracy is the most important evaluation criterion, if a reduction in the number of rules comes with a high cost in accuracy, it is better to have more rules. Thus, it is important to understand how the number of LRAR varies with the similarity threshold θ, while taking the impact in the accuracy of the model into account as well.

In Figure 2.3 we see how many LRAR are generated per dataset as θ varies. The majority of the plots, 10 out of 14, show a decrease in the number of rules as θ gets closer to 1. As discussed before, the accuracy in general also decreases as $\theta \geq 0.6$, so let us focus on $\theta \in [0, 0.6]$.

In the interval $\theta \in [0, 0.6]$, the number of rules generated is quite stable in 9 out of 14 datasets. In the first half of this interval, $\theta \in [0, 0.3]$, it is even more remarkable for 13 datasets.

We expect the number of rules to decrease as θ increases, however, results show that the number of rules does not decrease so much, especially for values up to 0.3. This is due to the fact that θ is also used in the pruning step (Section 2.4.1), reducing the number of rules against which the improvement of an extension is measured and, thus, increasing the probability of an ex-

[4] The *default rule* was not used in these experiments because it is not related with θ.

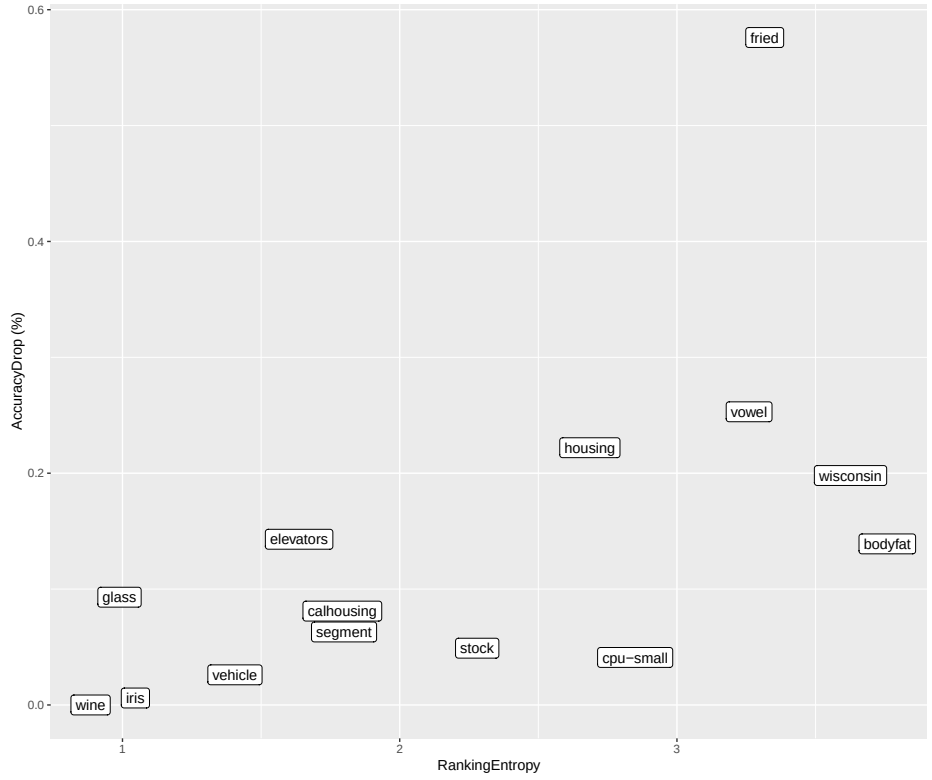

Figure 2.2: Measured *accuracy range* (Kendall τ) of CAREN in comparison to ranking entropy.

tension not being kept in the model. This means that, $minImp_{lr}$ is being effective in the reduction of LRAR.

As mentioned before, $imp_{lr}(A \to \pi)$ not only compares rules $A' \to \pi$ where $A' \subset A$, but also rules $A \to \pi'$ where $S'(\pi', \pi) \geq \theta$. In other words, with the $minImp_{lr}$ we are pruning LRAR with similar rankings too.

These results do not lead to any strong conclusions about the ideal value for θ regarding the number of rules. However, they are in line with the previous analysis of *accuracy*.

Minimum Confidence As mentioned before, we use a greedy algorithm to automatically adjust the minimum confidence in order to reduce the number of examples that are not covered by any rule. This means that the method has to adapt the value of *minconf* per dataset per θ, as seen in Figure 2.4.

In general, the *minconf* decreases in a monotonic way as θ increases. As

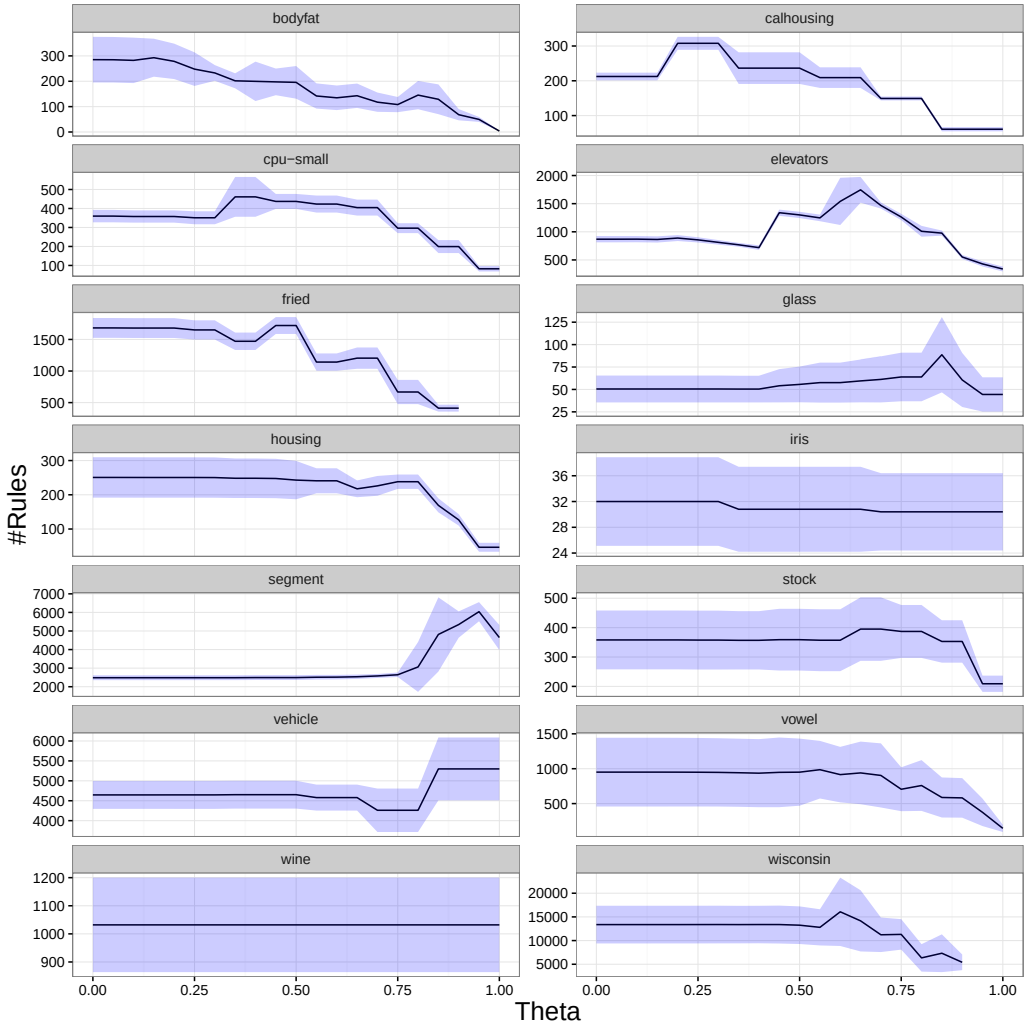

Figure 2.3: Number of Label Ranking Association Rules generated by CAREN as the θ varies

$\theta \approx 1$ the *minconf* gets to its minimum with 13 out of 14 datasets, which is consistent with the accuracy plots (Figure 2.1). This means that, if we want to generate rules with as much confidence as possible, we should use the minimum θ, i.e. $\theta = 0$.

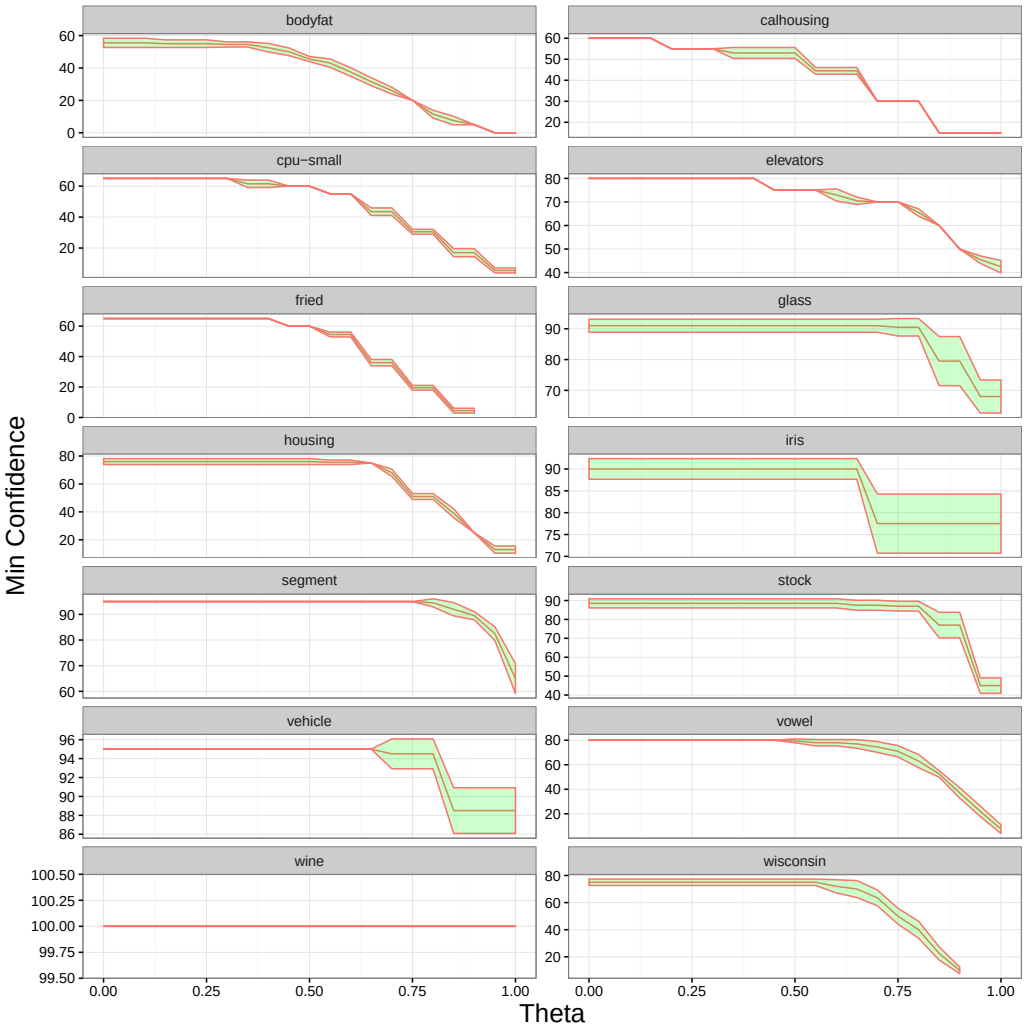

Figure 2.4: Mininum confidence adjusted to CAREN as the θ varies

Support versus accuracy We vary the minimum support threshold, *minsup*, to test how it affects the accuracy of our learner. A similar study has been carried out on CBA [75]. Specifically, we vary the *minsup* from 0.1% to 10%, using a step size of 0.1%. Due to the complexity of these experiments, we only considered the six smallest datasets.

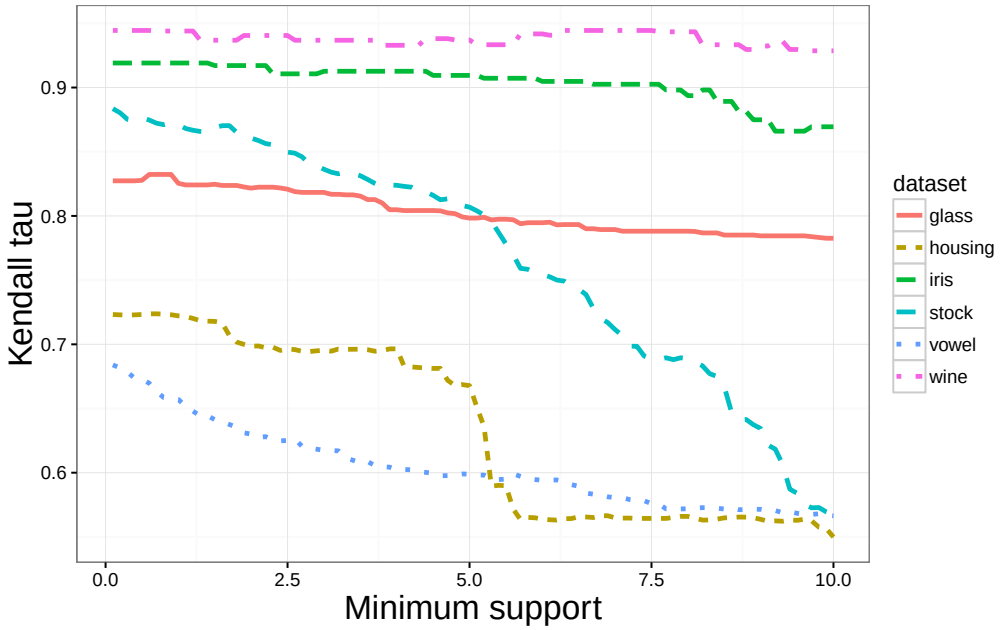

Figure 2.5: Average accuracy (Kendall τ) of CAREN as the *minsup* varies.

In general, as we increase *minsup* the accuracy decreases, which is a strong indicator that the support should be small (Figure 2.5). All lines are monotonically decreasing, i.e. either the values remain constant or they decrease as *minsup* increases.

From a different perspective, the changes are generally very small for $minsup \in [0.1\%, 1.0\%]$. Considering that lower *minsup* generate potentially more rules, we recommend $minsup = 1\%$ as a reasonable value to start experiments with.

Discretization techniques To test the influence of the discretization method used, we performed the same analysis using a non-supervised discretization method, *equal width*. In general, the accuracy had the same behavior, as a function of θ, as with *EDiRa*, i.e. the results are highly correlated (Figure 2.6). However, the supervised approach is consistently better. These results add further evidence that *EDiRa* is a suitable discretization method for label ranking [39].

Similar behavior was observed concerning the number of rules generated and the minimum confidence.

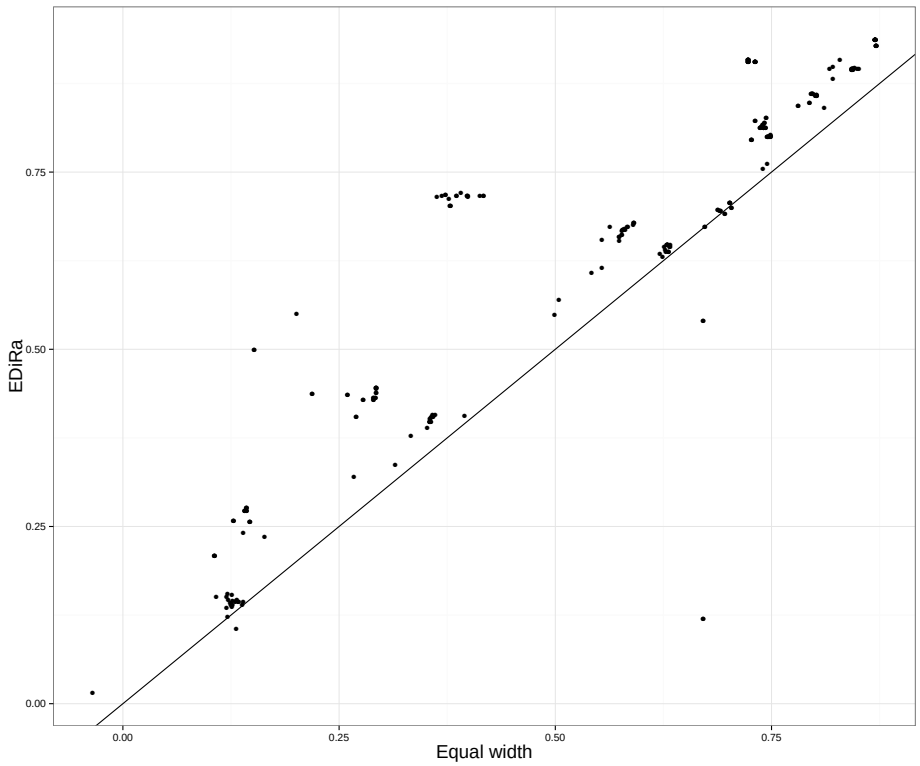

Figure 2.6: Ranking accuracy (Kendall τ) of CAREN after the discretization of data using *equal width* and *EDiRa*. This plot aggregates all the experiments carried out, concerning different issues, which means that each dataset is represented multiple times, with different parameter settings.

Summary It is well known that general, simple rules to set parameters of machine learning algorithms do not exist. Nevertheless it is good to know where reasonable values lie. Hence, we think that $\theta \in [0.5, 0.6]$ and $minsup = 1\%$ are good default values for LRAR with CAREN. In terms of the discretization methods, our results confirm that a supervised approach, such as *EDiRa*, is a good choice.

2.6.4 Results with PAR

In this work we use PAR, as a descriptive model, to find patterns concerning subsets of labels. We focus in the descriptive task for two reasons. One is to make the approach more simple and the other one is because this complements with the predictive LRAR approach.

The minimum support and confidence presented here are defining the abstention level of the model. The *minsup* and *minconf* were adjusted manually to generate a small set of rules between 150 to 200.

In the generation of PAR, we set the minimum lift to 1.5. Despite that many interesting rules were found, due to space limitations we only present the most relevant.

Algae data Using the Algae dataset, we found 179 PARs with $minsup = 2$ and $minconf = 90$. With $sup = 2.2\%$ and $conf = 100\%$ the rule with the highest lift (approx. 6) was:

$$\begin{aligned}&\texttt{Riversize} = \textit{small} \wedge \texttt{pH} \geq 37.9 \wedge \texttt{Flowvelocity} = \textit{high} \wedge \\ &\texttt{Chloride} \geq 3.4 \wedge \texttt{Nitrates\&Ammonia} \geq 18.5 \\ &\rightarrow L6 \succ L2 \wedge L5 \succ L7 \wedge L2 \succ L7\end{aligned}$$

The consequent of this rule can be represented as $L6 \succ L2 \succ L7 \wedge L5 \succ L7$. Considering that the labels represent algae populations, this rule states that it is always true that, under these conditions, type 6 is more prevalent than type 2. It also states that type 7 is less prevalent than types 2, 5 and 6.

The second rule with highest lift, with $sup = 3.1\%$ and $conf = 91\%$ is:

$$\begin{aligned}&\texttt{Flowvelocity} = \textit{medium} \wedge \texttt{Nitrates\&Ammonia} < 18.5 \wedge \\ &\texttt{Nitrogenasnitrates} < 7.9 \\ &\rightarrow L1 \succ L7 \wedge L7 \succ L3\end{aligned}$$

The target of this rule is the partial ranking $L1 \succ L7 \succ L3$.

If this PAR was used for prediction, the subranking $\pi = (1, 0, 3, 0, 0, 0, 2)$ would have been the prediction.

Sushi data When analyzing the sushi dataset we got 166 rules with $minconf = 70\%$ and the $minsup = 1\%$. With a lift of 1.95 the following rule was found:

$$\begin{aligned}&\texttt{Ageinterval} = 15 - 19 \wedge \texttt{Sex} = \textit{Male} \wedge \texttt{Livedin} = \textit{Eastern Japan} \\ &\rightarrow \texttt{egg} \succ \texttt{seaurchin} \wedge \texttt{shrimp} \succ \texttt{seaurchin}\end{aligned}$$

In the whole dataset, 37% of the people show this relative preferences $\texttt{egg} \succ \texttt{seaurchin} \wedge \texttt{shrimp} \succ \texttt{seaurchin}$. This PAR shows that this number almost

double (72%), if we consider males from Eastern Japan, aged between $15-19$.

A related rule was also found concerning a different group of people, with different age and from a different region ($sup = 1.1\%$, $conf = 71.6\%$ and $lift = 1.65$):

$$\begin{aligned}&\texttt{Ageinterval} = 30-39 \wedge \texttt{Sex} = \textit{Male} \wedge \\ &\texttt{Livesin} = \textit{Western Japan} \wedge \texttt{Changedcity} = \textit{Yes} \\ &\rightarrow \texttt{seaurchin} \succ \texttt{egg} \wedge \\ &\texttt{fattytuna} \succ \texttt{tunaroll} \wedge \\ &\texttt{tunaroll} \succ \texttt{cucumberroll} \wedge \\ &\texttt{fattytuna} \succ \texttt{egg}\end{aligned}$$

This rule includes one relative preference found in this group, `seaurchin` $\succ$ `egg`, which is the opposite to what was observed in the previous rule. Based on this information, we analyzed the data and found out that 75% of people that live in Eastern Japan prefer `egg` to `seaurchin` while 84% of people from Western Japan prefer `seaurchin` to `egg`.

2.7 Conclusions

In this paper we address the problem of finding association patterns in label rankings. We present an extensive empirical analysis on the behavior of a label ranking method, the CAREN implementation of Label Ranking Association Rules. The performance was analyzed from different perspectives, *accuracy*, *number of rules* and *average confidence*. The results show that, similarity-based interest measures contribute positively to the accuracy of the model, in comparison to frequency-based approaches, i.e. when $\theta = 1$. The results confirm that LRAR are a viable label ranking tool which helps solving complex label ranking problems (i.e. problems with high ranking entropy). The results also enabled the identification of some values for the parameters of the algorithm that are good candidates to be used as default values.

Results also seem to indicate that, the higher the entropy, the more the accuracy can be affected by the choice of θ. An user can measure the ranking entropy of a dataset beforehand and adjust θ accordingly.

Additionally, we propose Preference Association Rules (PAR), which are association rules where the consequent represents multiple pairwise preferences.

We illustrated the usefulness of this approach to identify interesting patterns in label ranking datasets, which cannot be obtained with LRAR.

In future work, we will use PAR for predictive tasks.