Ranking Cases with Classification Rules

Jianping Zhang, Jerzy W. Bala, Ali Hadjarian, and Brent Han

The MITRE Corporation, 7515 Colshire Drive
McLean, Virginia 22102-7508, USA
jzhang@mitre.org, jerzy.w.bala@gmail.com {ahadjarian/bhan}@mitre.org

Abstract. Many real-world machine learning applications require a ranking of cases, in addition to their classification. While classification rules are not a good representation for ranking, the human comprehensibility aspect of rules makes them an attractive option for many ranking problems where such model transparency is desired. There have been numerous studies on ranking with decision trees, but not many on ranking with decision rules. Although rules are similar to decision trees in many respects, there are important differences between them when used for ranking. In this chapter, we propose a framework for ranking with rules. The framework extends and substantially improves on the reported methods for ranking with decision trees. It introduces three types of rule-based ranking methods: post analysis of rules, hybrid methods, and multiple rule set analysis. We also study the impact of rule learning bias on the ranking performance. While traditional measures used for ranking performance evaluation tend to focus on the entire rank ordered list, the aim of many ranking applications is to optimize the performance on only a small portion of the top ranked cases. Accordingly, we propose a simple method for measuring the performance of a classification or ranking algorithm that focuses on these top ranked cases. Empirical studies have been conducted to evaluate some of the proposed methods.

Key words: Classification Rules, Ranking, ROC, Probability Estimation Trees

1 Introduction

Many real-world machine learning applications require a ranking of cases, in addition to their classification. Such ranking is often based on some measure of reliability or likelihood or a numeric assessment of the quality of each classification (e.g., probability value of a class membership). In other words, the decision-making process extends the class membership prediction to include an estimate of the reliability for this prediction. For example, in credit application processing, the goal is to rank applicants in terms of their likelihoods of profitability or loan defaults. This is significantly different than simply classifying them into qualified versus non-qualified groups. Other decision-making applications where case ranking could be of importance include bankruptcy prediction, medical diagnosis, customer targeting for marketing campaigns, and customer churn prediction.

Ranking of cases is particularly important for those decision-making applications where it is preferable to abstain from decision making altogether in the absence of sufficient support. Examples include medical and military applications.

While classification rules are not a good representation for ranking, the human comprehensibility aspect of rules makes them an attractive option for many ranking applications where such model transparency is desired or even an essential requirement. There have been numerous studies on ranking with decision trees [8,9,10,1], but not many on ranking with decision rules. Although rules are similar to decision trees in many respects, there are important differences between them when used for ranking. Separate-&-conquer (covering) techniques of rule learning algorithms may generate rules that overlap, whereas divide-&-conquer techniques of decision trees do not result in such overlapping of decisions. Rules may not cover some areas of a feature space, but the leaf nodes of a decision tree cover the entire area of the feature space (see Figure 1). In addition, a rule learning algorithm for a two class problem may only learn rules for one class, but a decision tree always includes leaf nodes for both classes. Rule learning algorithms tend to generate fewer rules than leaf nodes of a decision tree. Such differences bring both research challenges and opportunities for developing methods for ranking cases with rules.

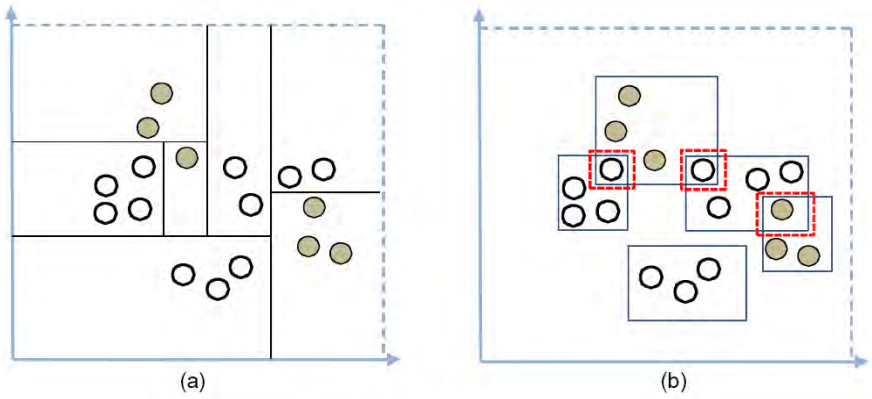


Fig. 1. Decision boundaries of a decision tree (a) vs. those of a rule-based classifier (b).

In this chapter, we propose a framework for ranking with rules. The framework extends and substantially improves on the reported methods for ranking with decision trees. It introduces three types of rule-based ranking methods: post analysis of rules, hybrid methods, and multiple rule set analysis (i.e., rule ensembles and redundant rules). Methods for combining scores from overlapping rules are also proposed and studied.

We also study the impact of rule learning bias on the ranking performance. While traditional measures used for ranking performance evaluation tend to

focus on the entire rank ordered list, the aim of many ranking applications is to optimize the performance on only a small portion of the top ranked cases. Accordingly, we propose a simple method for measuring the performance of a classification or ranking algorithm that focuses on these top ranked cases. More specifically, instead of measuring the entire area under the ROC curve, the proposed method computes only the left most part of it (i.e., the part covering only the top n% of the ranked cases). Empirical studies have been conducted to evaluate some of the more popular post analysis methods, methods for combining scores of overlapping rules, and the impact of rule learning bias.

The remainder of the chapter is organized as follows. Related work is discussed in Section 2. Section 3 describes the proposed framework. Section 4 describes the empirical results of two separate studies: (1) a real-world application in investigations of companies suspected of financial fraud, and (2) performance evaluation using six of the UCI Machine Learning Repository data sets. Finally, Section 5 concludes the chapter with future research.

2 Related Work

2.1 Expert Systems and Fuzzy Logic

Ranking examples using rules is not a novel undertaking. There have been many attempts in the past, including those by the researchers in the fields of expert systems, fuzzy logic, and cognitive science [2]. MYCIN [4] is a well known rule-based expert system, in which each rule is assigned a certainty factor (CF) by domain experts. CF of different rules may be combined or propagated to produce the CF of a decision inferred by MYCIN. In PROSPECTOR [12], an expert system to assist geologists working in mineral exploration, rules are assigned probability by human experts and are propagated and combined using Bayesian inference.

Development of fuzzy rules has been studied widely in fuzzy logic, e.g. [13]. Here, a general method is developed to generate fuzzy rules from numeric data, using linguistic variables. The degree of class membership of an example depends on the degree of match of the example to a fuzzy rule.

2.2 Machine Learning

There are also several related rule based machine learning studies, e.g., [13]. These studies generally focus on methods for generating partial matching, whereby the scores for individual examples are computed based on how well they match the rules. In these approaches, examples that satisfy all conditions of a rule share the same score.

Extensive studies have been dedicated to the incorporation of ranking capabilities into the decision tree learning paradigm.

Related work generally falls into the following four groups of methods: learning probability estimation trees [10], geometric methods [1], hybrid trees including: the Perceptron Tree [11] and NBTree [8], and ensembles of trees [3]. Other methods have also been reported in [7] [9].

3 Framework for Ranking with Classification Rules

A rule-based classifier, typically defined as a disjunctive normal form of conditional rules, is a mapping function from a set of m arguments or attributes (which can be either nominal or numeric) to a single nominal value, known as the class. Let us represent by D the set of d classification decisions, generally labeled by numbers $0, 1, 2, \ldots d-1$, and by E the set of unlabeled examples. A classifier is a function $f : E \rightarrow D$. In a simplified scenario, a classifier with ranking capability computes a number or score for every example $e \in E$ and for every class $i \in D$.

A score may be interpreted differently depending on the application. In statistics, a score ranges from 0 to 1 and indicates the probability of an example belonging to a given class. In expert systems, a score may be interpreted as a certainty factor. In fuzzy logic, a score represents the degree of membership in a class. Similarly, in cognitive science, a score could be defined as the typicality of the example being a member of the class. In other applications, a score is just a measure for ranking cases.

The framework for ranking cases with classification rules presented in this chapter consists of a grouping of methods that can be used independently or jointly. Figure 2 depicts the taxonomy of these methods.

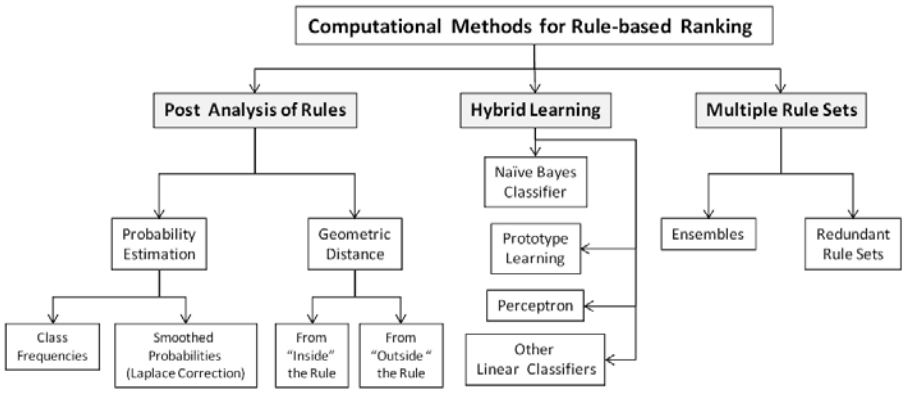


Fig. 2. Computational methods for rule-based ranking.

The framework extends and substantially improves on the reported methods for ranking with decision trees. It introduces the following three groups of rule-based ranking computation methods:

- Post analysis of rules
- Hybrid methods
- Multiple rule set analysis (i.e., rule ensembles and redundant rules)

In addition to the three groups of rule-based ranking methods, we introduce methods for combining the scores of overlapping rules. We also study the impact of the inductive bias on the ranking performance.

3.1 Post Analysis of Rules

In this group of methods, ranking scores are computed using the rules generated by some rule induction algorithm plus some additional information such as the number of positive and negative examples covered or the ranges of attribute values. There are two subgroups of methods under this group: probability estimation and geometric methods.

Probability Estimation With probability estimation, the score of the example covered by a rule is computed as the ratio of the number of positive examples covered and the total number of examples covered by a rule. This approach has been explored in Probability Estimation Trees.

In the simple probability estimation method, scores assigned by two different rules are identical as long as the above mentioned ratio is the same, no matter how many positive examples are actually covered by each rule. For example, all rules covering no negative examples result in the same score, namely one. A simple method for overcoming this problem is the Laplace correction, in which the score is computed using $\frac{k+1}{n+C}$, where k and n are the numbers of positive examples and the total number of examples covered, respectively, and C is the number of classes. Here, the scores are the same for all the examples covered by a given rule. This becomes a serious problem when the number of rules generated by a rule induction algorithm is small, as many of the examples will end up with identical scores.

The probability estimation method, even with the Laplace correction, ignores the absolute number of examples covered by the rule. A rule covering 9 positive examples and 10 negative examples receives the same score as a rule covering 90 positive examples and 100 negative examples. In real applications, however, even when having similar precisions, rules covering more examples are generally preferred. An alternative option would be to use the F-measure for scoring:

$$F\text{-}measure(r) = \frac{\beta^2 + 1}{\frac{\beta^2}{recall(r)} + \frac{1}{precision(r)}} \tag{1}$$

where β is a parameter for assigning relative weights to recall and precision. When β is set to 1, recall and precision are weighted equally. F-measure favors recall with $\beta > 1$ and favors precision with $\beta < 1$. Namely, an F-measure with a large β value favors more general and less precise rules, while one with a small β value favors more specific and more precise rules. When $\beta = 0$, F-measure score is the same as probability estimation.

Geometric Methods As indicated above, the probability estimation method assigns the same score to all examples covered by the same rule. When the number of rules generated is small, this causes a problem for applications that need a fine grained ranking. For example, in one of our data mining applications, only 0.1% of the top ranked cases are selected for further investigations. If all rules cover more than 0.1% of the examples, we have no way to accurately select 0.1% of the top ranked cases. Geometric methods can help generate such fine grained rankings.

Geometric methods have been used in decision trees [1] and assume that classifications/rankings of the examples near the rule boundary are less certain. In ranking with rules, there are two types of geometric methods, one for examples covered by a rule and one for examples that are not covered by any rules. The latter is also called partial matching. For an example covered by a rule, we can measure the distance between the example and the rule boundary or the center of the rule. The closer the example is to the boundary (or the farther from the center), the smaller its score gets. The distance may also be weighted by the estimated probability of the rule. The geometric method for covered examples works for numeric attributes.

Partial matching also computes the distance of an uncovered example to the boundary of a rule, but from outside of the rule. The closer to the boundary, the larger the score is. Again, the distance could be weighted by the estimated probability of the rule. Partial matching is different from strict matching, where an example has to satisfy all the conditions of the matched rule. Partial matching computes a degree of match between an example and a rule. The degree of match can vary in the range of 0 (matches no condition) to 1.0 (matches all conditions). Partial matching works for both numeric and nominal attributes.

3.2 Hybrid Methods

Hybrid methods integrate rule induction with other learning techniques that have ranking capabilities. These latter techniques include the Perceptron algorithm, Naïve Bayes, instance-based learning, and prototype-based learning. For example, a separate Perceptron may be learned for examples covered by each rule. Figure 3 shows a rule with a linear classifier in a two dimensional space. The rectangle represents the rule and the line inside the rectangle represents the linear classifier. The white circles represent the positive examples, while the gray ones represent the negative examples. The linear classifier is used to assign a score to each of the examples covered by the rule.

Rules and linear or Naïve Bayes models may be learned together to optimize the performance of the hybrid method. Alternatively, rules may be learned first and other models then generated for each rule. As discussed in Section 2, there are many studies on building hybrid decision trees, e.g., Perceptron Tree and NBTree.

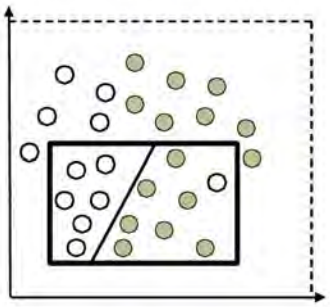


Fig. 3. A rule and the corresponding linear classifier.

3.3 Rule Ensembles and Redundant Rules

Previous studies have shown that ensemble techniques can significantly improve the ranking performance of decision trees [3,10]. In ensemble learning, multiple classifiers are learned, using approaches such as bagging and boosting. Typically, in such ensembles, a majority vote technique is used to determine the final classification of an example and the average score can be used as the final score for each example. Although little work has been done in building ensembles of rules, we believe such an ensemble can yield improved ranking performance.

Previous studies have also shown how redundant rules could improve classification performance. Since redundant rules allow for finer grained rankings, it is worthwhile to design a rule induction algorithm for generating redundant rules for ranking.

3.4 Combining Scores from Overlapping Rules

A rule induction algorithm usually generates a set of overlapping rules. In this section, we discuss three simple methods: *Max*, *Average*, and *Probabilistic Sum (P-Sum)*, for combining the scores of an example covered by more than one rule. The *Max* approach simply takes the largest score of all the rules that cover the example. Given an example e and a set of l rules $RS = \{R_1, \ldots, R_l\}$, the combined score of e using *Max* is computed as follows:

$$score(e, RS) = \max_{i=1}^{l} score(e, R_i) \tag{2}$$

where $score(e, R_i)$ is the score of e assigned by Rule R_i. Similarly, the combined score of e using *Average* is computed as follows:

$$score(e, RS) = \frac{1}{l} \cdot \sum_{i=1}^{l} score(e, R_i) \tag{3}$$

For rules that do not cover e, $score(e, R_i) = 0$ unless partial matching is applied. For the *P-Sum* method, the formula can be defined recursively as follows:

$$
\begin{aligned}
score(e, \{R_1\}) &= score(e, R_1) \\
score(e, \{R_1, R_2\} &= score(e, R_1) + score(e, R_2) \\
&\quad - score(e, R_1) \times score(e, R_2) \\
score(e, \{R_1, \ldots, R_n\} &= score(e, \{R_1, \ldots, R_{n-1}\}) + score(e, R_n) \\
&\quad - score(e, \{R_1, \ldots, R_{n-1}\}) \times score(e, R_n)
\end{aligned}
$$

Both *Average* and *P-Sum* generate a finer grained ranking than *Max*. These three methods may also be used to combine the scores in a rule ensemble.

3.5 Impact of the Rule Induction Algorithm

Most rule induction algorithms have been designed for maximizing the classification accuracy. Recently, some learning algorithms have been proposed that optimize the AUC (Area Under the Curve) of a ROC (Receiver Operating Characteristics) curve [6]. The design of a rule induction algorithm for optimizing the AUC could be an interesting future research objective.

Rule induction algorithms typically include parameter settings which allow the users to trade generality for accuracy. When such parameters are set to favor accuracy, more rules may be generated. On the one hand, more rules result in a finer grained ranking, but on the other hand, the rules themselves tend to be over-specific.

4 Empirical Studies

4.1 Study 1: Detection of Public Companies Suspected of Financial Fraud

This section focuses on a real-world data mining application to prioritize human investigations of companies suspected of engaging in fraudulent behavior based on their financial filings.

The Problem Incidents of financial statement fraud have increased substantially over the past two decades and affected individuals, especially investors and creditors, have lost billions of dollars as a result. Financial statement fraud involves the misstatement of a company's financial information with the intent to mislead users of such information, such as investors and creditors. Typical falsifications of financial statements include manipulation of assets, sales, profits, liabilities, expenses, or losses. Such frauds are often difficult to detect, because they are typically committed by a company's top management team, who understands the limitations of an audit, with an opportunity and motive to distort the financial statements.

Prevention of financial statement fraud by internal or external auditors is probably the best strategy; however, sometimes the auditors themselves are involved in the fraud activity or make mistakes. Thus, in order to minimize the potential economic impact, there is a need for early detection of financial statement fraud by regulatory organizations. There have been several past studies to address such a need.

This section focuses on a data mining application to prioritize a list of target companies suspected of engaging in fraudulent behavior based on their financial statements. Such prioritization of potentially large volumes of targets is particularly crucial for applications such as this one where the decisions ultimately rest with human examiners with limited resources. In other words, only companies deemed as highly suspicious, normally a small percentage of the total population, will eventually go through the vigorous task of in-depth examination.

The application involves building predictive models for identifying companies that may have failed to comply with accounting principles or violated securities laws with respect to the accuracy of public disclosure information. The raw data used in this application contains SEC registered public organizations' financial filings (publicly available SEC EDGAR filings), including those of companies with materially misstated financial statements. Each training example corresponds to a company filing and is a vector of about 100 attribute values that are primarily numeric. Positive examples are all filings for companies that had issued material restatements, while negative examples are filings of companies that had not issued such restatements.

As mentioned previously, the learned predictive models are intended to help the human investigators prioritize which companies to look at more closely and to optimize investigative resources and increase efficiency by focusing examiners on highly suspicious companies that warrant the most attention. The investigators use the predictive models as a tool to generate a rank ordered list of target companies to be examined. These predictive models will neither replace human inspectors nor fully automate the suspicious behavior detection processes.

A Ranking Performance Metric Since only a very small percentage, typically 1 to 2 percent depending on the resources available, of top ranked companies may be selected for detailed inspections, it is important to maximize the classification precision (or true positive rate) on these top ranked companies. The classifier's performance on lower ranked companies, as such, becomes rather irrelevant. This problem is similar to that of web search, in which a search engine might return thousands of web pages for a given query, but only a small number of which, typically the top ranked pages, is ever viewed by the user. Thus, it is much more important for an effective search engine to optimize the relevance on top ranked pages than the lower ones.

While existing research offers an array of machine learning algorithms that can accommodate such ranking of classification decisions, these algorithms, and measures such as the area under the ROC curve (AUC) used to evaluate their performance, generally tend to focus on the entire rank ordered list. Based on

the need for our application, we propose a simple method for measuring the performance of classification/ranking algorithms that instead of measuring the entire area under the ROC curve, it computes only the left most portion of it (i.e., the part covering only the top n% of the ranked cases). We have named the aforementioned area as LAUC (Left-most portion of the Area Under the Curve). Figure 4 shows the ROC curves obtained from two different learning algorithms. With LAUC as a measure (i.e., to the left of the cutoff point), Algorithm 2 achieves a better performance than Algorithm 1, whereas Algorithm 1 is preferred over Algorithm 2 with the AUC. The have been some recent studies by machine learning researchers focusing on the notion of LAUC [9].

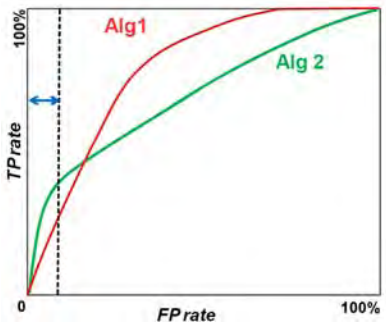


Fig. 4. Example ROC curves. The LAUC cutoff point is depicted by the dashed line.

Data and Setting For these experiments, we create input records for each company based on cases that are publicly available on the SEC EDGAR filing site. Each input record is a vector of attribute values that are primarily numeric. These are a combination of original data that is contained in sources such as SEC required financial filings as well as calculated and derived attributes. These input records are split into two datasets based on the year, one for training and one for testing. The training set includes the data from 2003 to 2004, while the test set contains the data from 2005. Examples in both sets are described by 130 attributes, 95 of which are numeric and the other 35 are symbolic. The training set includes 4,932 positive and 38,792 negative examples. The test set is composed of 836 positive and 18,780 negative examples. In our experiments, we randomly selected 50% of the examples from the training set to learn the rules and the process was repeated five times. Rules were then tested on the test set and the results were reported using the average of the five runs. The classification performance has been evaluated with both the AUC as well as the LAUC as the measure. Here, LAUC is the area under the ROC curve at the left of the 1% false positive rate cutoff point, normalized by the total area to the left of that point. LAUC is 0.005 for random selection.

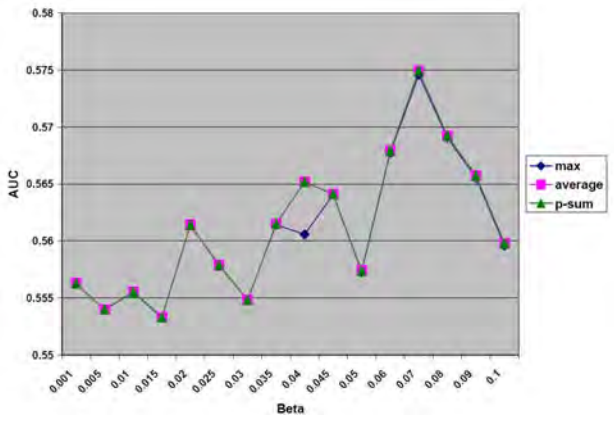


Fig. 5. AUC values reported for the first experiment.

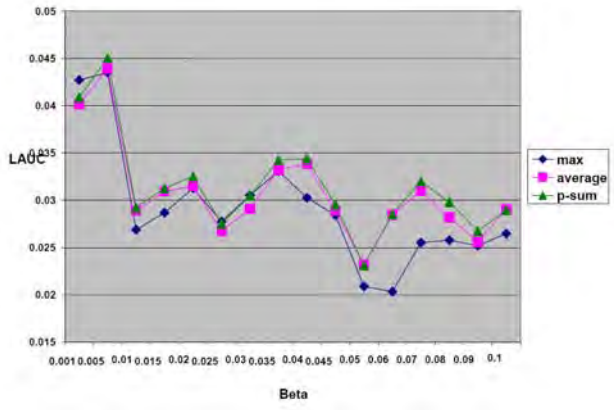


Fig. 6. LAUC values reported for the first experiment.

The classification task is quite challenging due to factors such as noisy data, which is generally the result of the mechanism by which the positive and negative examples have been labeled. Here, negative examples are not truly negative, rather their class memberships are not known. As such, it is not quite possible to learn rules with high recall and precision. For example, when minimum precision and minimum recall were set to 0.6 and 0.03 respectively, no rules were generated. Only a small number of positive examples could be covered by rules with high precision. Since there is no well established baseline performance data for the comparison of our results, we compared the performance of our model with the random choice model by assuming that target class examples are drawn at random from all the potential examples.

Results In the first experiment, with minimum recall and minimum precision parameters set to 0.01 and 0.6, respectively, we varied β from 0.001 to 0.1. Accordingly, all the rules learned have a recall larger than 1% and a precision larger than 60%. With $\beta = 0.001$, a set of highly specific and precise rules was generated. Each increase in β resulted in more general and less precise rules. Probability estimation with Laplace correction, using three different score combining methods: *Max*, *Average*, and *P-Sum*, was applied to the learned rules to obtain the scores of all the test examples. Figure 5 shows the AUCs for different β values and different score combining methods. The general trend of the performances of all three score combining methods seems to be upward with an increase in β until it reaches 0.07 and then the performances start to drop sharply. When β is small, the generated rules are highly specific and precise, so they cover only a small number of positive examples. Therefore, many of the positive examples not covered by these rules are assigned a score of 0 and are consequently ranked low. When β increases, rules become more general and cover more positive examples and the ranks of these newly covered positive examples move up, so the performance increases accordingly. When β becomes too large, however, rules tend to get much more general and much less precise, so that many negative examples are also covered. This causes the performance to drop. The three score combining methods perform similarly. It can also be seen that the reported AUC values barely beat the random performance. This shows the difficulty of the problem. Many positive examples cannot be covered by any rules with a recall larger than 1% and a precision larger than 60%.

Figure 6 shows the LAUC values. In contrast to the AUC, the best performances are achieved with smaller β values. This is what we had initially expected because a small β value results in highly specific and precise rules. As such, not many negative examples were covered by these rules. When β gets larger, however, rules cover more negative examples because they are more general and less precise. These covered negative examples are ranked high, so the performances on top ranked examples degrade. The three score combining methods seem to perform about the same for smaller β values and more specific rules. For larger β values, however, *P-Sum* and *Average* are better than *Max*, with *P-Sum* working slightly better than *Average*. This is because the chance of specific rules overlapping is smaller than that of general rules. Figure 7 displays the true positive rate at the cutoff point of 1% false positive rate. They are about 5 to 7 times better than random.

Figure 8 shows the ROC curves for $\beta = 0.001$ and $\beta = 0.07$ for the *Max* method. It can be seen that the performance on top ranked cases is better when $\beta = 0.001$, while the performance on all ranked cases is better when $\beta = 0.07$. The results of this study tend to suggest that smaller β values are better for achieving a high performance on top ranked cases.

In the second experiment, we varied the minimum recall parameter value from 0.005 to 0.04 with $\beta = 0.001$ and minimum precision = 0.4. Figure 9 shows the reported AUC values, which tend to go down with an increase in minimum recall. Since the rule induction algorithm is forced to generate rules with a high

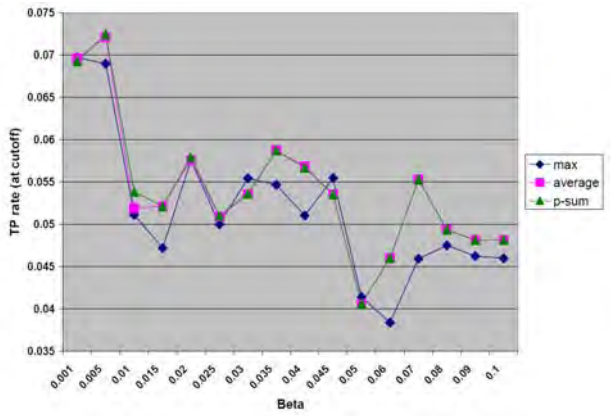


Fig. 7. True positive rates at cutoff point of 1% false positive rate reported for the first experiment.

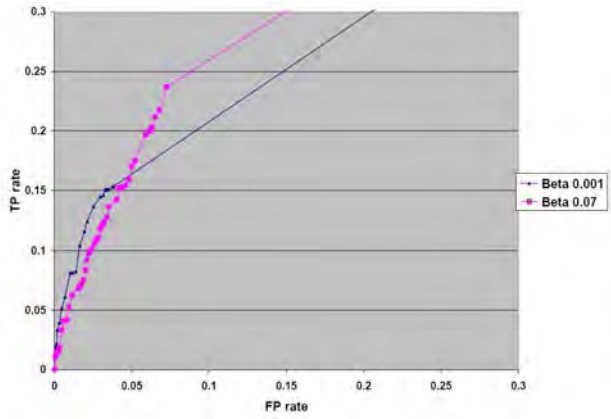


Fig. 8. ROC curves for $\beta = 0.001$ and $\beta = 0.07$ plotted for the first experiment.

recall and a low precision, many negative examples are covered by these rules and correspondingly ranked higher. When both minimum recall and minimum precision are low, the rule induction algorithm is able to generate more rules so that more positive examples are covered. Figure 10 reports the LAUC values. Again, following the same reasoning, an increase in minimum recall results in a decrease in performance. Here, *P-Sum* and *Max* tend to do better than *Average* for lower minimum recall values.

In the third experiment, we varied the minimum recall parameter value from 0.005 to 0.04 with $\beta = 0.001$ and minimum precision = 0.7. Figure 11 shows the reported AUC values, which tend to go down with an increase in minimum recall for reasons similar to those discussed above for the second experiment. The difference is that when the minimum recall is increased, the number of

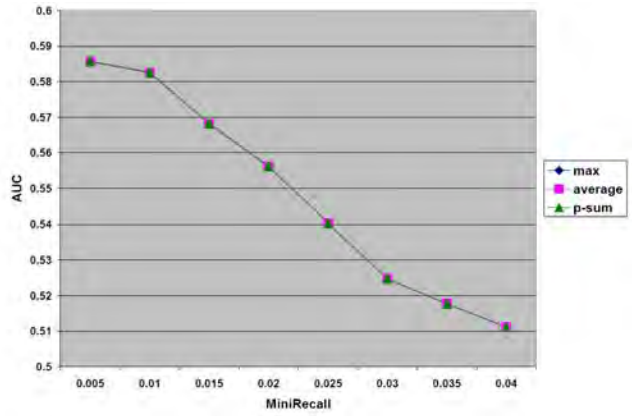


Fig. 9. AUC values reported for the second experiment.

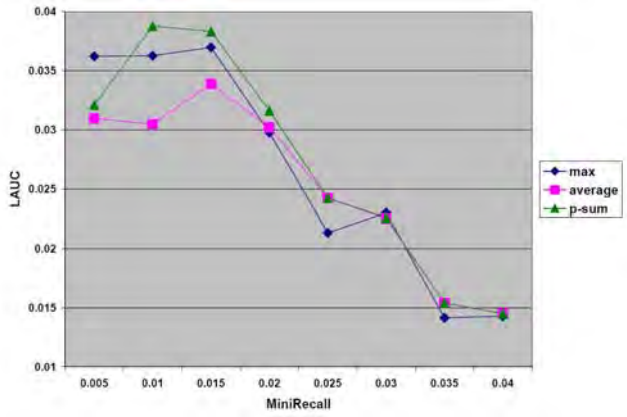


Fig. 10. LAUC values reported for the second experiment.

rules generated goes down quickly. Actually, when the minimum recall is larger than 0.03, no rule is generated, so the reported AUC values are the same as those of a random selection. Figure 12 reports the LAUC values. Here, the same reason causes the performance to decrease quickly when the minimum recall is increased.

In the fourth experiment, we varied the minimum precision parameter value from 0.2 to 0.9 with $\beta = 0.001$ and minimum recall = 0.01. Figure 13 shows the reported AUC values, which seem to go down with an increase in minimum precision. When the minimum precision is increased, fewer rules are generated and so fewer positive examples are covered and the performance goes down as a result. When the minimum precision is kept low, the performance is better than those in all previous experiments. Figure 14 reports the LAUC values. While the three score combining methods perform similarly for larger values of

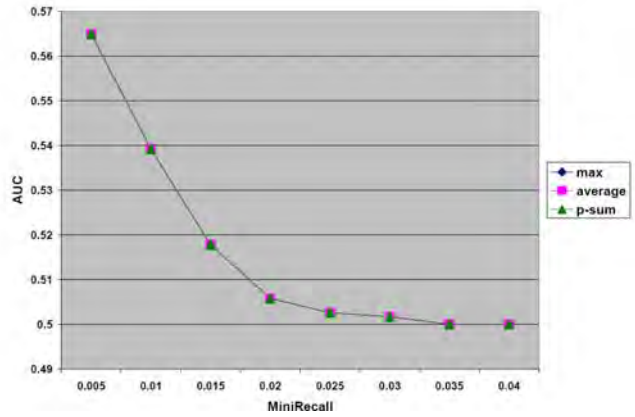


Fig. 11. AUC values reported for the third experiment.

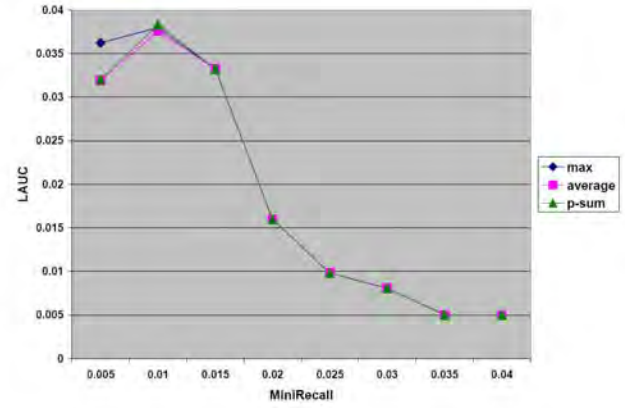


Fig. 12. LAUC values reported for the third experiment.

minimum precision, the performances are quite varied when minimum precision is smaller. In the latter case, *P-Sum* seems to do the best. This is because when the minimum precision is small, many rules are generated and as such may overlap more often. So *P-Sum* improves the performance. When the minimum precision is large, fewer rules are generated and they are also more specific. So rules overlap less often and *P-Sum* and *Average* cannot improve the performance.

Summary The experimental results clearly show that the performance measured by LAUC does not necessarily correlate with that measured by the AUC. They also show that the inductive bias impacts LAUC and AUC differently. More specific and more precise rules work better with LAUC as the measure, while more general rules work better with AUC. The three score combining methods

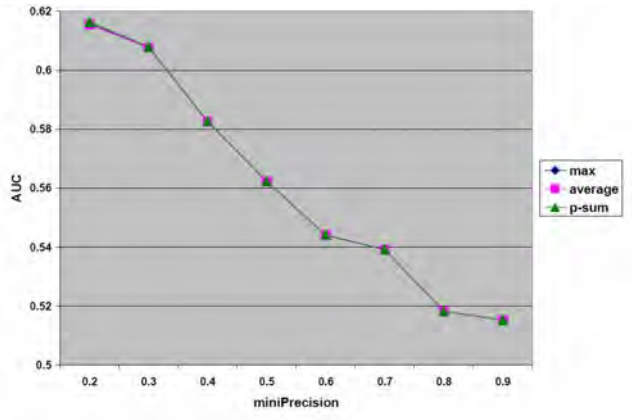


Fig. 13. AUC values reported for the fourth experiment.

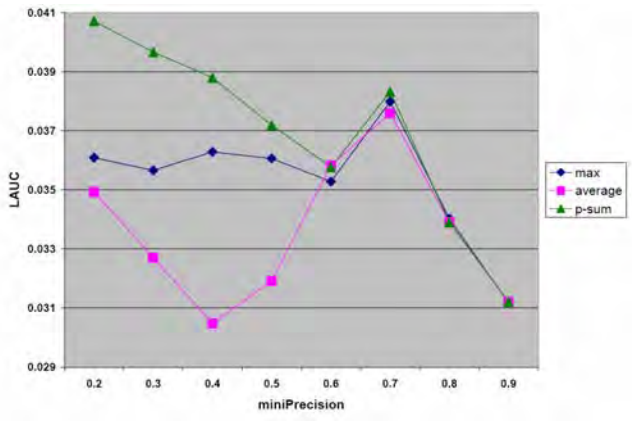


Fig. 14. LAUC values reported for the fourth experiment.

achieved about the same performance with AUC, while *P-Sum* generally did better than the other two methods with LAUC. The experimental results suggest that a smaller β, minimum recall and minimum precision values together with *P-Sum* should be used to achieve the optimal performance as measured by LAUC.

4.2 Study 2: Public Data Sets

We conducted experiments on six of the publicly available UCI Machine Learning Repository data sets. These are D1: Breast Cancer, D2: Chess (two class scenario), D3: German Credit, D4: Japanese Credit, D5: Magic, and D6: Yeast. The sample sets present a wide range of domains and cover a comprehensive

suite of data characteristics. To generate rules, we used the RIPPER algorithm [5] that induces classification rules from a set of pre-classified cases.

The reported AUC values for each data set were calculated using a 10-fold cross validation. Experiments were conducted with three groups of rules representing varying levels of simplifications for the generated rule sets (i.e., a larger number of more specific and precise rules or a smaller number of more general and overlapping rules). These variations were achieved by changing the RIPPER parameter settings. The training and test data sets were kept the same among these three groups of rule sets.

Six different scoring methods were used. The first three use probability estimation with three different methods, *Max*, *Average*, and *P-Sum*, for combining the scores of an example covered by multiple rules as describe in Section 3.4. These three methods are denoted as follows.

- AUC-FR0: *Max*
- AUC-FR1: *Average*
- AUC-FR2: *P-Sum*

The remaining group of methods utilizes the Laplace corrections, also with the three score combining methods. They are denoted as:

- AUC-LC0: *Max*
- AUC-LC1: *Average*
- AUC-LC2: *P-Sum*

We also use a measure that represents the average number of scores per example. It takes a value from 0 to 1 and is computed by dividing the number of unique scores divided by the number of test examples. For finer grained rankings, this number is generally closer to 1. In many applications, a finer grained ranking is preferred. This ranking-grain measure is used for the six different score computation methods as described above (*RankGrain-FR0, RankGrain-FR1, RankGrain-FR2, RankGrain-LC0, RankGrain-LC1,* and *RankGrain-LC2*).

Figure 15 depicts the summary of the reported AUC values for the six different methods. The reported values are the average on all six datasets. In the figure, Series 1 is the rule set with the most general rules, Series 2 is the rule set with less general rules, and Series 3 is the rule set with the most specific rules. Except for FR0, rules of Series 2 and Series 3 outperformed rules of Series 1. Namely, specific rules outperformed general rules. It means specific rules are better for ranking than general rules, because specific rules are able to produce finer grained rankings than general rules. However, rules of Series 2 and Series 3 performed about the same. This suggests that highly specific rules may help with ranking. Highly specific rules tend to include more perfect rules, which could be better for ranking. It is also shown in the figure that *P-Sum* performed better than *Average*, which in turn did better than *Max*. *P-Sum* and *Average* produce finer grained rankings than *Max* and assume that examples covered by multiple rules should be ranked higher. Laplace correction achieved about the same performance as the simple probability estimation.

Table 1. Reported AUC values.

Dataset	AUC-FR0	AUC-RF1	AUC-FR2	AUC-LC0	AUC-LC1	AUC-LC2
Series 1	$R = 0.19$					
D1	0.98	0.98	0.98	0.98	0.98	0.98
D2	0.77	0.79	0.79	0.77	0.78	0.79
D3	0.92	0.92	0.92	0.92	0.92	0.92
D4	0.82	0.84	0.84	0.82	0.84	0.84
D5	1.00	1.00	1.00	1.00	1.00	1.00
D6	0.91	0.91	0.91	0.91	0.91	0.91
Averages	*0.90*	*0.91*	*0.91*	*0.90*	*0.91*	*0.91*
Series 2	$R = 0.25$					
D1	0.99	0.99	0.99	0.99	0.99	0.99
D2	0.84	0.85	0.88	0.83	0.85	0.87
D3	0.95	0.95	0.96	0.95	0.95	0.96
D4	0.85	0.88	0.89	0.84	0.87	0.89
D5	1.00	1.00	1.00	1.00	1.00	1.00
D6	0.92	0.93	0.93	0.92	0.93	0.93
Averages	*0.92*	*0.93*	*0.94*	*0.92*	*0.93*	*0.94*
Series 3	$R = 0.28$					
D1	0.92	0.99	0.99	0.95	0.99	1.00
D2	0.84	0.85	0.87	0.83	0.85	0.87
D3	0.96	0.96	0.96	0.95	0.95	0.96
D4	0.85	0.88	0.89	0.84	0.87	0.89
D5	0.90	1.00	1.00	1.00	1.00	1.00
D6	0.93	0.93	0.93	0.93	0.93	0.93
Averages	*0.90*	*0.93*	*0.94*	*0.92*	*0.93*	*0.94*

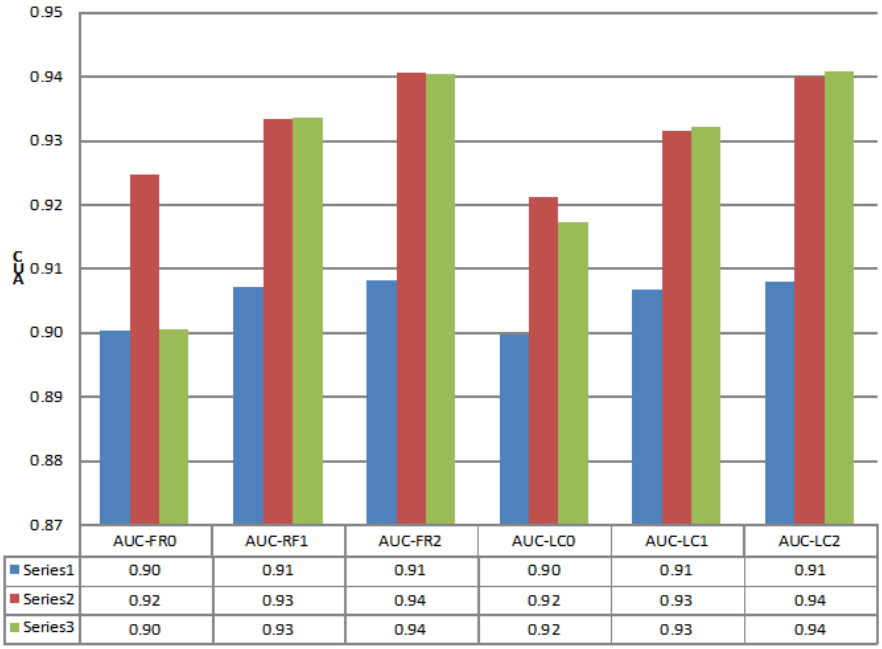
Table 1 reports the detailed AUC values for each of the six data sets. The R measurement represents the ratio of the number of perfect rules (rules that do not cover negative examples) to the total number of rules in a given rule set. Here the average value of R is depicted (for each of the three runs on the six data sets). The R measurement is affected by using RIPPER's s parameter which simplifies or specializes the generated rule set (i.e., trading off generality for accuracy by the degree of hypothesis simplification). When the parameter is set to favor the generation of more specific rules, then more accurate results can be achieved.

Figure 16 shows the summary of the *RankGrain* values graphically. Table 2 reports the *RankGrain* values. As seen in the figure, it seems that more specific rules result in finer grained rankings. The difference between Series 2 and Series 3

Table 2. The reported *RankGrain* values.

Dataset	RankGrain -FR0	RankGrain -FR1	RankGrain -FR2	RankGrain -LC0	RankGrain -LC1	RankGrain -LC2
Series 1	$R = 0.19$					
D1	0.11	0.31	0.24	0.11	0.31	0.31
D2	0.02	0.13	0.05	0.07	0.6	0.19
D3	0.13	0.29	0.29	0.13	0.29	0.29
D4	0.11	0.18	0.18	0.11	0.18	0.18
D5	0.03	0.37	0.24	0.03	0.37	0.25
D6	0.09	0.21	0.21	0.09	0.21	0.21
Averages	*0.08*	*0.25*	*0.20*	*0.09*	*0.27*	*0.24*
Series 2	$R = 0.25$					
D1	0.15	0.36	0.26	0.13	0.39	0.36
D2	0.02	0.13	0.05	0.07	0.27	0.19
D3	0.22	0.65	0.64	0.21	0.65	0.65
D4	0.19	0.45	0.44	0.18	0.45	0.45
D5	0.06	0.63	0.35	0.06	0.63	0.36
D6	0.15	0.49	0.44	0.14	0.49	0.45
Averages	*0.13*	*0.45*	*0.36*	*0.13*	*0.48*	*0.41*
Series 3	$R = 0.28$					
D1	0.14	0.40	0.28	0.13	0.42	0.37
D2	0.02	0.13	0.05	0.07	0.27	0.19
D3	0.23	0.68	0.66	0.23	0.68	0.68
D4	0.20	0.48	0.48	0.19	0.48	0.48
D5	0.06	0.63	0.35	0.06	0.63	0.35
D6	0.14	0.50	0.46	0.14	0.50	0.47
Averages	*0.13*	*0.47*	*0.38*	*0.14*	*0.50*	*0.42*

is small. *Average* and *P-Sum* produce more scores than *Max*. The number of different scores for *Max* cannot be larger than the number of rules and the number of scores for *Average* and *P-Sum* can never be smaller than *Max*. When there are many overlapping rules, *Average* and *P-Sum* may generate many more scores. It is interesting to see that *Average* produces more scores than *P-Sum*, which may be due to the number of perfect rules. As long as an example is covered by a perfect rule, *P-Sum* can assign it a perfect score of 1, no matter how many rules it is covered by. On the other hand, *Average* may give a different score to examples covered by some perfect rules. If all rules are perfect rules, both *Max* and *P-Sum* produce only two scores 0 and 1, but *Average* is still able to produce more scores. Laplace correction is able to produce slightly more scores than the simple probability estimation method because of the perfect rules. With



	AUC-FR0	AUC-RF1	AUC-FR2	AUC-LC0	AUC-LC1	AUC-LC2
Series1	0.90	0.91	0.91	0.90	0.91	0.91
Series2	0.92	0.93	0.94	0.92	0.93	0.94
Series3	0.90	0.93	0.94	0.92	0.93	0.94

Fig. 15. Graphical summary of the reported AUC values.

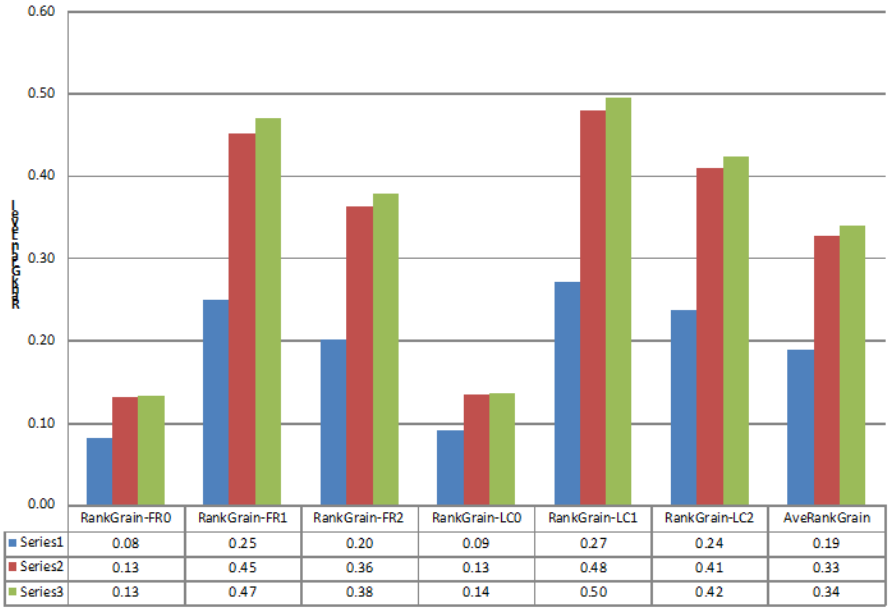
Laplace correction, perfect rules may produce different scores depending on the number of examples covered.

The following are some of the conclusions drawn from the results:

- Specific rules produce higher AUC values than general rules.
- *P-Sum* generates the best AUC results in comparison with *Average* and *Max*.
- Simple probability estimation method performs about the same as Laplace correction.
- Specific rules produce more scores than general rules.
- *Average* generates the best *RankGrain* results, while *Max* generates the worst *RankGrain* results.
- *RankGrain* values computed using the Laplace correction (*RankGrain-LCn*) are slightly better than the ones computed using class frequencies (*RankGrain-FRn*)
- The AUC value correlates with and increases with the number of scores.

5 Conclusions

In this chapter, we proposed a framework for ranking with rules. The framework introduces three types of rule-based ranking methods: post analysis of rules, hybrid methods, and multiple rule set analysis. We also proposed three methods:



	RankGrain-FR0	RankGrain-FR1	RankGrain-FR2	RankGrain-LC0	RankGrain-LC1	RankGrain-LC2	AveRankGrain
Series1	0.08	0.25	0.20	0.09	0.27	0.24	0.19
Series2	0.13	0.45	0.36	0.13	0.48	0.41	0.33
Series3	0.13	0.47	0.38	0.14	0.50	0.42	0.34

Fig. 16. Graphical summary of the reported *RankGrain* values.

Max, *Average*, and *Probabilistic Sum*, for combining the scores of an example covered by multiple rules. A successfully deployed data mining application in financial fraud detection was also discussed. The application aims at prioritizing human investigations of public companies suspected of engaging in fraudulent behavior based on their financial filings. Additional empirical studies were conducted using six of the UCI Machine Learning Repository data sets to evaluate the two simplest rule-based ranking methods: probability estimation with and without Laplace correction, as well as the three rule score combining methods. We investigated the impact of the inductive bias on the ranking performance. Experimental results clearly suggest that the inductive bias has an impact on the ranking performance. It is also shown that *Probabilistic Sum* and *Average* generally perform better than *Max*. It seems that a method that produces more scores usually outperforms a method that produces fewer scores. More experiments need to be conducted to verify this.

Future research will include an empirical validation of some of the other methods in the framework, specifically the ones that use geometric measures. Since with geometric methods more scores could be produced, it would be interesting to see whether such methods could improve the ranking performance. Also considered is a study comparing the performance of rule-based and decision tree-based ranking approaches.

References

1. Alvarez I. & Bernard S. (2005). Ranking Cases with Decision Trees: A Geometric Method that Preserves Intelligibility. In *Proceedings of the 19th International Joint Conference on AI (IJCAI-05)*, pp. 635-640.
2. Barsalou (1985). Ideals, Central Tendency, and Frequency of Instantiation as Determinants of Graded Structure In Categories. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 11:629–654
3. Bauer E. & Kohavi R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting And Variants. *Machine Learning*, 36, 105-142.
4. Buchanan B.G. and Shortliffe E.H. (eds.) (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison Wesley, Reading, MA.
5. Cohen W. (1995). Text Categorization and Relational Learning. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pp. 124-132.
6. Cortes C. & Mohri M. (2003). AUC Optimization vs. Error Rate Minimization. In *Advances in Neural Information Processing Systems (NIPS-03)*. MIT Press.
7. Ferri C., Flach, P.A., & Hernandez-Orallo, J. (2003). Improving the AUC of Probabilistic Estimation Trees. In *Proceedings of the 14th European Conference on Machine Learning (ECML-03)*. Springer.
8. Kohavi R. (1996). Scaling Up The Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*
9. Ling C.X. & Yan, R.J. (2003). Decision Tree with Better Ranking. In *Proceedings of the 20th International Conference on Machine Learning (ICML-01)*. Morgan Kaufmann.
10. Provost F. J. & Domingos, P. (2003). Tree induction for probability-based ranking. *Machine Learning*, 52(30), pp. 199-215.
11. Utgoff P. (1988). Perceptron Trees – A Case Study In Hybrid Concept Representation. In *Proceedings of The Seventh National Conference on Artificial Intelligence (AAAI-88)*, 601-606.
12. Waterman D.A. (1985). *A Guide to Expert Systems.* Addison-Wesley Publishing Company. Reading, Mass (USA).
13. Wang L.X. & Mendel J.M. (1992). Generating Fuzzy Rules by Learning from Examples, *IEEE Transactions on Systems, Man and Cybernetics*, 22(6), pp. 1414-1427.
14. Zhang J. & Michalski R.S. (1995). An Integration of Rule Induction and Exemplar-Based Learning for Graded Concepts. *Machine Learning* 21(3): 235-267.