Towards the Development of a Problem Solver for the Monitoring and Control of Instrumentation in a Grid Environment

Tatiana Kalganova, Somjet Suppharangsan, Russell Taylor, Mujtaba Alsaif
School of Engineering and Design
Brunel University
Uxbridge, UB8 3PH, United Kingdom
Tatiana.Kalganova@brunel.ac.uk

Francesco Lelli
INFN
National Laboratory of Legnaro
Via Università 2, Legnaro, Italy
Francesco.Lelli@lnl.infn.it

Abstract **– This paper considers the issues involved in developing a generic problem solver to be used within a grid environment for the monitoring and control of instrumentation. The specific feature of such an environment is that the type of data to be processed, as well as the problem, is not always known in advance. Therefore, it is necessary to develop a problem solver architecture that will address this issue. We propose to analyze the performance of the problem solving algorithms available within the WEKA toolkit and determine a decision tree of the best performing algorithm for a given type of data. For this purpose the algorithms have been tested using 51 datasets either drawn from publicly available repositories or generated in a grid-enabled environment.**

I INTRODUCTION

The rapid development of grid-enabled services is mainly driven by the need to use large computational resources in such applications as meteorology, Human Proteome Folding or the processing of medical data. While remote control of, and data collection from, instrumentation was part of the initial grid concept most recent grid developments have been concentrated on the sharing of distributed computational and storage resources.

In this scenario applications that need computational power have just to use these grid elements in order to access an unlimited amount of computational power and disk storage. Existing grid architectures are therefore not appropriate for applications incorporating real-time measurements from instrumentation, where there is a need for a strong interaction between the instrumentation and the computational grid. GRIDCC, a European Commission-funded project, is developing an architecture and set of services that will enable the monitoring and control of instrumentation in a grid environment [1], [2].

The importance of data mining services in grid architectures have been highlighted in [3] and further reviewed in [4]. A number of projects have been established with the aim of implementing grid-enabled data mining interfaces and services, where the major focus was on the development of services for data grid architectures. Examples of such projects include GridMiner [5] and DataMiningGrid [6]. The introduction of instrumentation into a grid architecture elicits a new role for data mining in a grid environment, where the data should be processed from at least two points of view: (1) data processing and (2) the fault management of instrumentation. In the second case, it should be noted that different kinds of instrumentation will have completely different ways of collecting information and that, in contrast to a particular implementation of a classical grid, this information will be markedly heterogeneous. This makes it necessary to develop a problem solver with a generic structure – in other words, a problem solver that is able to process data efficiently irrespective of the size of the processed dataset and its type.

The diverse roles of problem solvers, such as problem recognition, definition and analysis, data management and collection and solution development demonstrate the complexity of developing a generic problem solver [7]. Moreover, in developing the problem solver we should concentrate not only on processing and analysis techniques but also on the development of explanation techniques [8].

We will refer to a *generic problem solver* as being a problem solver that is capable of adapting to and solving a generic problem [9]. One of the ways to approach the development of a generic problem solver is to develop one algorithm along with a technique to “fit” into the problem domain, for example by utilizing the domain ontology while acquiring human expert knowledge [10] or by utilizing genetic programming principles [11]. The obvious advantage of such an approach is in the use of just one algorithm that is capable of solving a number of different problems. At the same time, a significant disadvantage is that its performance will vary depending on the problem tackled. In order to avoid this disadvantage we propose to develop an algorithm-based decision tree so that when running the problem solver, the best performing algorithm for a given data type will be chosen. Both problem-specific and problem-generic algorithms will be able to participate in the decision tree – it being an essential feature of the GRIDCC architecture that the problem solver should be flexible enough to include problem specific algorithms should they exist. This can be easily achieved if the WEKA environment [12] is utilised as the “container” for such algorithms.

Therefore, the purpose of this paper is to investigate the behaviour of the existing algorithms integrated into the WEKA toolkit, and to analyse their behaviour using various applications and data types. From the results obtained, we aim to develop an algorithm-based decision tree that selects the best performing algorithm for a given type of data. The performance of the algorithms has been tested based on datasets taken from publicly available repositories, as well as several generated within a grid-enabled environment.

II Datasets

The 51 datasets have been collected from the UCI Machine Learning Repository [13], from URLs [14], [15], and from a grid CE (Compute Element) cluster at INFN (Istituto Nazionale di Fisica Nucleare), Legnaro. The 'grid' datasets have been incorporated in the benchmark analysis because, like these, the data from instrumentation will typically be unclassified datasets. The sizes of the datasets range from 36 instances to 67557 instances. They are divided into four types, namely 1) small size and none-missing datasets, 2) small size and missing datasets, 3) large size and none-missing datasets and 4) large size and missing datasets. A "none-missing" dataset is one where every attribute in every instance contains a valid value, whereas in a dataset described as "missing" some attributes of some instances do not have valid values. The boundary separating small and large datasets is 1000 instances, since the datasets vary from tens to thousands of instances. Also, the class type and type of attribute are employed to categorise the datasets. There are 29 small supervised datasets, 16 large supervised datasets, 2 datasets for regression (housing and abalone), and 4 unsupervised datasets (those from the grid cluster: grid700, grid1750, grid3500 and grid7000). Of the 29 small datasets, there are 17 small and none-missing datasets, of which 8 have nominal (non-numeric) class and numeric attribute, 3 nominal class and nominal attribute and 6 nominal class and mixed (combination of numeric and nominal) attribute, while there are 12 small and missing datasets, of which 1 has nominal class and numeric attribute, 2 nominal class and nominal attribute and 9 nominal class and mixed attribute. Of the 16 large datasets, there are 13 large and none-missing datasets, of which 7 have nominal class and numeric attribute, 5 nominal class and nominal attribute and 1 nominal class and mixed attribute, while there are 3 large and missing datasets, of which 1 has nominal class and nominal attribute and 2 have nominal class and mixed attribute.

III Classification

Databases can have nominal, numeric or mixed attributes and classes. Not all classification algorithms perform well for different types of attributes and classes as well as for different size databases. In aiming to design a generic classification tool, one should consider the behaviour of various existing classification algorithms on different datasets. WEKA is an excellent tool for such an investigation since it can be easily integrated into JavaScript and new algorithms can be added. Our aim at this stage is to analyse the existing classification algorithms implemented in the WEKA toolkit and define a decision tree according to their performance. There exist many classification algorithms [16] that can be classified according to design methodology. Here we analyse the tree and rule based classification algorithms provided in WEKA [12]. Several tree and rule algorithms are applied to each dataset and then evaluated for accuracy by using 10-cross-validation strategy [17]. 10-cross-validation (10-CV) is a standard way of predicting the error rate. To perform 10-CV a dataset is separated into ten approximately equal portions, each of which is used in turn for testing with the other nine being used for training (meaning that ten iterations are performed in total).

A Tree Algorithms

Tree algorithms generate a model by constructing a tree where each internal node is a feature or attribute. The leaf nodes are class outputs. Each dataset is tested using the following tree algorithms: ADTree [12], DecisionStump [18], ID3 or Inductive Decision trees [19], J48 (which is based on C4.5R8 algorithm [20] and the original C4.5 algorithm [21]), LMT (Logistic Model Trees) as developed by Landwehr [22], M5P (originally called M5') according to Holmes et al [23], NBTree or Naïve Bayes Trees created by Holmes et al [23], RandomTree as explained by Tan in [24] and its extension RandomForest [24], which simply generates a specified number of RandomTrees and finally REPTree [25].

B Rule Induction

Rule Induction algorithms generate a model as a set of rules. The rules are in the form of standard IF-THEN rules. Most rule algorithms rely on tree algorithms. Each dataset is tested using the following rule algorithms: ConjunctiveRule [26], which generates a single rule; DecisionTable or DecisionTableMajority (DTM) [27]; JRip, which is based on Cohen's RIPPER algorithm [28]; M5Rules, which generates rules using the M5' described in [23]; NNge (Nearest Neighbour using Generalized Exemplar) [29]; OneR based on the 1R algorithm [30]; PART, named because it uses a PARTial tree to generate its knowledge base [31]; PRISM [32]; Ridor or Ripple Down Rule learner [33] and finally ZeroR.

IV Experiments and Results

The original datasets are converted to ARFF (Attribute Relation File Format), this being the input file format for WEKA. At this stage the dataset is ready for classification, regression or clustering, depending on dataset's characteristics. Most of datasets fall under classification; a few datasets, e.g. housing and abalone, fall under regression. The datasets from the grid-enabled cluster (grid700 etc.) are unsupervised and need to be clustered before classifying. WEKA provides numerous classification algorithms but only tree and rule algorithms are used here because they have easily understandable behaviour. The ten tree and ten rule algorithms identified in section III are tested on each dataset with the option of 10-CV enabled. The results are represented as the average accuracy over the ten iterations. Here we are interested in the percentage of correctly classified instances of the algorithms. The algorithms giving the most accurate estimate, in other words the algorithms with the lowest estimated error, are chosen. Table I shows the algorithms that yield the highest accuracy results for each of the small datasets whereas Table II shows the algorithms that yield the highest accuracy results for each of the large datasets. The first column is the criteria for classifying the datasets: the type of class, i.e. nominal or numeric, type of attribute, i.e. nominal, numeric or mixed, as well as whether there is missing data, is taken into account. The second column is the dataset name followed by two numbers in parenthesis.

The former is the number of attributes and the latter is the number of instances. The third and fourth columns are the chosen tree and rule algorithms, respectively, including their corresponding accuracy percentages. The last column is the best algorithm overall, based on which one offers the highest accuracy. For certain datasets, some algorithms are faced with a memory problem in WEKA but other algorithms are still able to deal with the classification in the datasets. The problem normally occurs in datasets with large numbers of instances or attributes. For such datasets, the candidate algorithm given is the most accurate of those that were able to run successfully.

Table I
Results of Small Datasets

Category	**Datasets**	**Tree (T)**		**Rule Induction (R)**		**Best Algorithms**		
		Algorithms	**Per cent**	**Algorithms**	**Per cent**	**Types**	**Algorithms**	**Per cent**
Small None Missing dataset Nominal Class, Numeric-Attribute	iris(4,150)	J48	96.00%	NNge	96.00%	T,R	J48,Nnge	96.00%
	bupa(6,345)	RandomForest	68.99%	NNge	66.67%	T	RandomForest	68.99%
	pima-indians-diabetes (8,768)	LMT	77.47%	JRip	75.13%	T	LMT	77.47%
	glass(9,214)	RandomForest, REPTree	98.60%	DecisionTable	98.13%	T	RandomForest,REPTree	98.60%
	vehicle(18,846)	LMT	82.98%	PART	71.51%	T	LMT	82.98%
	aminoacid(20,698)	LMT	45.99%	NNge	42.84%	T	LMT	45.99%
	ionosphere(34,351)	ADTree,LMT	93.16%	PART	91.74%	T	ADTree,LMT	93.16%
	sonar (60,208)	RandomForest	80.77%	PART	80.29%	T	RandomForest	80.77%
Small None Missing dataset Nominal Class, Nominal-Attribute	balance-scale(4,625)	LMT	93.12%	PART	77.28%	T	LMT	93.12%
	tic-tac-toe(9,958)	LMT	98.23%	Ridor	99.69%	R	Ridor	99.69%
	spect(22,267)	LMT	83.52%	JRip	84.64%	R	Jrip	84.64%
Small None Missing dataset Nominal Class, Mixed attribute	tae(5,151)	RandomTree	61.69%	NNge	63.64%	R	NNge	63.64%
	grub-damage(8,155)	NBTree	46.45%	OneR	41.94%	T	NBTree	46.45%
	vowel(13,990)	RandomForest	95.96%	NNge	87.47%	T	RandomForest	95.96%
	lymph(18,148)	LMT	83.11%	Ridor	85.14%	R	Ridor	85.14%
	pasture(22,36)	RandomForest	83.33%	Ridor	83.33%	T,R	RandomForest,Ridor	83.33%
	white-cover(31,63)	LMT	71.43%	Jrip	65.08%	T	LMT	71.43%
	grid700(154,700)*	J48	92.57%	PART	93.86%	R	PART	93.86%
Small None Missing dataset Numeric Class, Numeric-Attribute	housing(13,506)	M5P	62.42%	M5Ruls	60.16%	T	M5P	62.42%
Small and Missing dataset Nominal Class, Numeric-Attribute	breast cancer wisconsin(9,699)	NBTree	96.42%	NNge	96.28%	T	NBTree	96.42%
Small and Missing dataset Nominal Class, Nominal-Attribute	breast cancer(9,286)	LMT	76.22%	OneR	78.32%	R	OneR	78.32%
	voting-records(16,435)	LMT	96.55%	NNge	96.09%	T	LMT	96.55%
Small and Missing dataset Nominal Class, Mixed Attribute	post-operative(8,90)	J48,LMT,REPTree	70.00%	Ridor	71.11%	R	Ridor	71.11%
	credit(15,690)	J48	86.09%	Jrip	85.80%	T	J48	86.09%
	hepatitis(19,155)	J48	83.87%	PART	84.52%	R	PART	84.52%
	eucalyptus(19,736)	LMT	65.76%	Jrip	61.01%	T	LMT	65.76%
	colic(22,368)	RandomForest	86.14%	PART	84.78%	T	RandomForest	86.14%
	squash-unstored(23,52)	J48	82.69%	PART	80.77%	T	J48	82.69%
	squash-stored(24,52)	NBTree	73.08%	PART	65.38%	T	NBTree	73.08%
	autos(25,205)	RandomForest	83.41%	NNge	80.00%	T	RandomForest	83.41%
	dermatology(34,366)	LMT	97.54%	NNge	96.17%	T	LMT	97.54%

Table II
Results of Large Datasets

Categories	Datasets	Tree (T)		Rule Induction (R)		Best Algorithms		
		Algorithms	Per cent	Algorithms	Per cent	Types	Algorithms	Per cent
Large and None Missing dataset Nominal Class, Numeric-Attribute	shuttle(2) (9,14500)	RandomForest	99.93%	PART	99.89%	T	RandomForest	99.93%
	shuttle(1) (9,43500)	RandomForest	99.98%	PART	99.97%	T	RandomForest	99.98%
	page-Blocks (10,5473)	NBTree, RandomForest	97.24%	PART	97.06%	T	NBTree, RandomForest	97.24%
	letterrecognition (16,20000)*	RandomForest	94.46%	PART	89.05%	T	RandomForest	94.46%
	segment(19,2310)	RandomForest	97.88%	PART	96.28%	T	RandomForest	97.88%
	segmentation (19,2310)	RandomForest	97.62%	PART	96.45%	T	RandomForest	97.62%
	waveform(40,5000)	LMT	86.96%	JRip	79.20%	T	LMT	86.96%
Large and None Missing dataset Nominal Class, Nominal-Attribute	car(6,1728)	LMT	98.78%	Ridor	96.30%	T	LMT	98.78%
	krkopt(6,28056)*	J48	56.58%	PART	54.09%	T	J48	56.58%
	nursery(8,12960)	LMT	98.99%	PART	99.21%	R	PART	99.21%
	connect-4(42,67557)*	J48	80.97%	PART	79.25%	T	J48	80.97%
	splice(61,3190)*	NBTree	95.30%	JRip	94.45%	T	NBTree	95.30%
Large and None Missing dataset Nominal Class, Mixed Attribute	cmc(9,1473)	LMT	53.02%	DecisionTable	54.99%	R	DecisionTable	54.99%
	grid1750(154,1750)*	J48	94.06%	PART	95.26%	R	PART	95.26%
	grid3500(154,3500)*	J48	98.51%	JRip	98.91%	R	JRip	98.91%
	grid7000(154,7000)*	J48	99.13%	JRip	99.39%	R	JRip	99.39%
Large and None Missing dataset Numeric Class, Mixed Attribute	abalone(8,4177)	M5P	36.25%	M5Rules	35.40%	T	M5P	36.25%
Large and Missing dataset Nominal Class, Nominal-Attribute	mushroom(22,8124)	J48,NBTree, RandomForest	100.00%	DecisionTable, JRip,NNge, PART	100.00%	T,R	J48,NBTree, RandomForest, DecisionTable, JRip,NNge, PART	100.00%
Large and Missing dataset Nominal Class, Mixed Attribute	sick-euthyroid (25,3164)	J48	97.88%	Ridor	97.53%	T	J48	97.88%
	hypothyroid (29,3772)	J48,REPTree	99.58%	Ridor	99.44%	T	J48,REPTree	99.58%

* “not enough memory” occurring in some algorithms

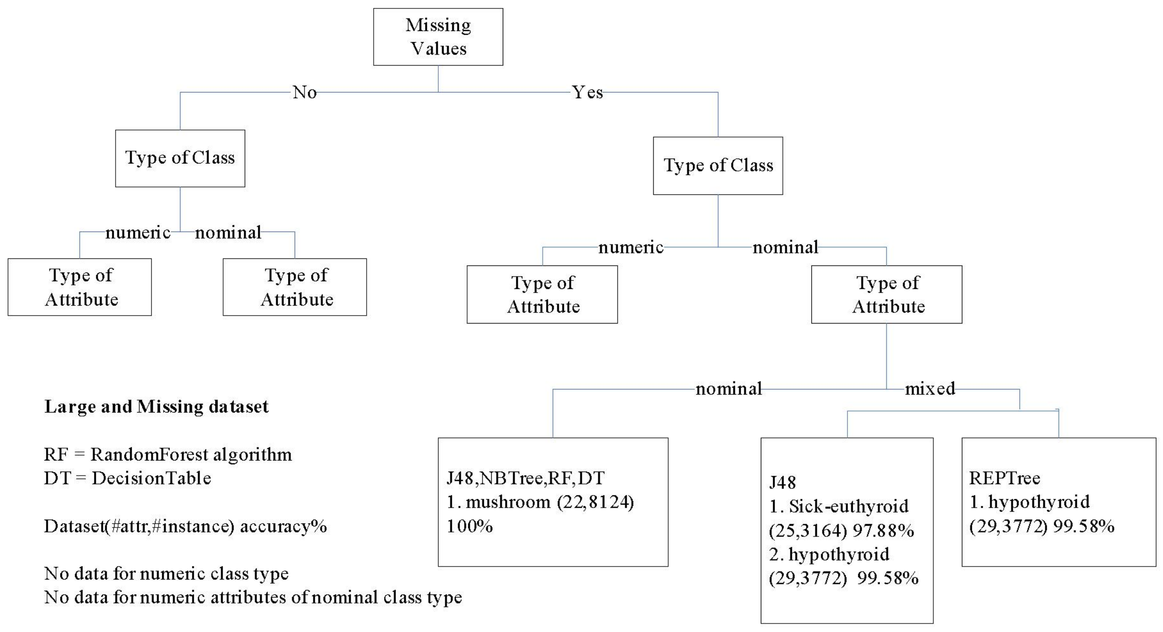
Missing Values
No
Yes
Type of Class
Type of Class
numeric
nominal
numeric
nominal
Type of Attribute
Type of Attribute
Type of Attribute
Type of Attribute
nominal
mixed
Large and Missing dataset
RF = RandomForest algorithm
DT = DecisionTable
Dataset(#attr,#instance) accuracy%
No data for numeric class type
No data for numeric attributes of nominal class type
J48,NBTree,RF,DT
1. mushroom (22,8124)
100%
J48
1. Sick-euthyroid (25,3164) 97.88%
2. hypothyroid (29,3772) 99.58%
REPTree
1. hypothyroid (29,3772) 99.58%

Figure 1
Diagram of Large and Missing datasets

Fig. 1 shows a diagram of the large and missing datasets using data from Table II. The diagram is similar to a binary tree except that there is no restriction to only two children nodes for each parent node. For instance, the type of attribute node could have three children nodes: nominal, numeric or mixed attribute. (Although there are no data with numeric attributes in the large and missing datasets, so Fig. 1 shows only nominal and mixed attributes in the diagram.) To begin with, we consider if there is missing data or not in the dataset. Then the type of class in the dataset is considered: numeric or nominal. In this example, all large datasets are categorised to have nominal class. At this stage, there are three nominal class datasets, namely mushroom, sick-euthyroid and hypothyroid. These datasets are then classified by type of attribute. Mushroom is classified to have nominal attribute while sick-euthyroid, and hypothyroid fall into mixed attribute. There are many algorithms yielding 100% accuracy for the mushroom dataset such as J48, NBTree, RandomForest, DecisionTable, JRip, NNge, and PART. The J48 algorithm is the best algorithm for both the sick-euthyroid and hypothyroid datasets, while the REPTree algorithm also gives the same accuracy for the hypothyroid dataset. Therefore, J48 is likely to be the candidate algorithm for the large and missing datasets. Similar diagrams for small and missing or none-missing datasets, and for large and none-missing datasets, can be drawn using data from Tables I & II, in the same way as described above.

V Conclusions

Most of the datasets in the GRIDCC environment will be nominal class. The LMT and RandomForest algorithms are likely to be selected for small datasets with nominal class. For numeric class, M5P is likely to be the best candidate algorithm as a result of a few numeric regression algorithms implemented in the experiments. Tree algorithms are more likely to be chosen than rule algorithms. In this investigation, most tree algorithms give more accurate results than rule algorithms in none-missing datasets with nominal class and numeric attribute. For this kind of small dataset, LMT and RandomForest are often chosen, whereas RandomForest is often chosen for the same kind of large dataset. However, tree and rule algorithms seem equally likely to be selected for small and none-missing datasets with nominal class and nominal or mixed attribute. Nonetheless, there is a “not enough memory” problem occurring in some datasets while running certain algorithms. Therefore, the error rates of these algorithms cannot be measured and compared.

The results show that there is no single best algorithm to beat all others in all situations. In some cases there might be, depending on the characteristics of the data. There are some algorithms that seem to be good candidates in some general cases, as mentioned in the above discussion. To choose a suitable algorithm, a domain expert or expert system may employ the results of the classification in order to make better decisions.

Acknowledgment

This project is supported under EU FP6 contract 511382.

References

[1] K. H. Darby-Dowman et al., “GRIDCC - Bringing instrumentation (back) onto the Grid,” Conference on Computing in High Energy & Nuclear Physics (CHEP), Mumbai, 13-17 February 2006

[2] “GRIDCC – Grid Enabled Remote Instrumentation with Distributed Control and Computation,” http://www.gridcc.org

[3] A. K. T. P. Au, V. Curcin et al., “Why grid-based data mining matters? fighting natural disasters on the grid: from SARS to land slides,” In S. J. Cox (Ed.), UK e-science all-hands meeting (AHM 2004), Nottingham, UK, September 2004, EPSRC, 2004, pp. 121-126

[4] P. Brezany, I. Janciak and A Min Tjoa, “Data mining on the grid: Perspective from the GridMiner experience,” 5th Cracow Grid Workshop, Poland, November 21-23, 2005

[5] P. Brezany, I. Janciak and A Min Tjoa, “GridMiner: A fundamental infrastructure for building intelligent grid systems,” The 2005 IEEE/WIC/ACM Int. Conf. on Web Intelligence (WI'05), pp. 150-156

[6] “Data Mining Tools and Services for Grid Computing Environments,” http://www.datamininggrid.org

[7] L. M. Taylor, “Much ado about nothing: The problem with problem-solving,” The College Quarterly, Seneca College of Applied Arts and Technology, 1994 – Vol. 2 (1)

[8] Vladia Pinheiro, Vasco Furtado, Paulo Pinheiro da Silva and Deborah L. McGuinness, “Explaining problem solver answers,” Technical Report KSL-05-02, Knowledge Systems Laboratory, Stanford University, USA, 2005

[9] D. Mann, “The space Between ‘generic’ and ‘specific’ problem solutions,” The TRIZ Journal 2001 Vol. 6

[10] G. Beydoun and A. Hoffmann, “Building problem solvers based on search control knowledge,” Knowledge Acquisition Workshop (KAW’98)

[11] A. Grigoryan, D. Kalina and J. Spiegel, “The generic genetic problem solver,” http://www.cs.columbia.edu/~evs/ais/finalprojs/kalina/

[12] I. H. Witten and E. Frank, “Data Mining: Practical machine learning tools and techniques,” 2nd edition, Morgan Kaufmann, San Francisco, 2005

[13] UCI Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html, Accessed: 5 Dec 2005

[14] S. Ji, Computational Biology, http://www.csc.lsu.edu/~ji/compbio/index.htm, Accessed: 5 Dec 2005

[15] Collection of datasets http://www.cs.waikato.ac.nz/~ml/weka/index_datasets.html, Accessed: 5 Dec 2005

[16] A. Küçükyılmaz, "Pattern classification: A survey and comparison," http://www.cs.bilkent.edu.tr/~guvenir/courses/cs550/Workshop/Ayse_Kucukyilmaz.pdf, Accessed: 15 Feb 2006

[17] R. R. Boukaert, "Choosing between two learning algorithms based on calibrated tests," In T. Fawcett and N. Mishra (eds.) Proc. of 20th Int. Conf. on Machine Learning, 2003, pp. 51-58

[18] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes and S. J. Cunningham, "Weka: Practical machine learning tools and techniques with Java implementations," In Kasabov, Nikola and Kitty Ko (eds.) Proc ICONIP/ANZIIS/ANNES’99 Int. Workshop: Emerging Knowledge Engineering and Connectionist-Based Information Systems, Dunedin, New Zealand, Nov. 1999, pp. 192-196

[19] J. R. Quinlan, "Discovering rules by induction from large numbers of examples: a case study," In D. Michie, editor, Expert systems in the microelectronic age, Edinburgh University Press, 1979

[20] Decision Trees for Supervised Learning, Toolshed Manual, http://grb.mnsu.edu/grbts/doc/manual/J48_Decision_Trees.html, Accessed: 8 Sep 2005

[21] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Mateo, CA: 1993

[22] N. Landwehr, M. Hall and E. Frank, "Logistic model trees," in Proc. of the 14th European Conf. on Machine Learning, LNCS 2837, Cavtat-Dubrovnik, Croatia, 2003, pp. 241-252, edited by N. Lavrac, et al. Springer-Verlag, Berlin

[23] G. Holmes, M. Hall and E. Frank, "Generating rule sets from model trees," in Proc. of the 12th Australian Joint Conf. on Artificial Intelligence, Sydney, Australia, Springer-Verlag, pp. 1-12

[24] Y. F. Tan, "Corpus based identification of light verb constructions," Undergraduate Thesis, School of Computing, National University of Singapore

[25] Class REPTree, Weka Toolkit API Documentation, http://alex.seewald.at/WEKA/doc_gui/weka/classifiers/trees/REPTree.html, Accessed: 8 Sep 2005

[26] Class ConjunctiveRule, Weka Toolkit API Documentation, http://weka.sourceforge.net/doc/weka/classifiers/rules/ConjunctiveRule.html, Accessed: 29 Jul 2005

[27] R. Kohavi, "The power of decision tables," in Proc. of the 8th European Conf. on Machine Learning, 1995, pp. 174-189

[28] W. Cohen, "Fast effective rule induction," in Pro. of the 12th Int. Conf. on Machine Learning, Tahoe City, CA, July 1995, pp. 115-123

[29] B. Martin "Instance-based learning: Nearest neighbor With generalization," Masters Thesis, University of Waikato, Hamilton, New Zealand

[30] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," Machine Learning, Vol. 11, 1993, pp. 63-91

[31] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in Proc. Int. Conf. on Machine Learning, Madison, Wisconsin, 1998, pp. 144-151

[32] G. Parker, Pseudo-code for Prism, Classification Algorithms, COM307: Machine Learning and Data Mining, http://cs.conncoll.edu/com307/lectureslides/DM4c.ppt#283,17,Pseudo-code for PRISM, Accessed: 11 Aug 2005

[33] B. Gaines and P. Compton, "Induction of ripple-down rules applied to modeling large database," Journal of Intelligent Information Systems, Vol. 3, No. 3, 1995, pp. 211-228