Classification tree chaining in data analysis

Dunja Mladenić[1], Marko Grobelnik[2], Ray J. Paul[3], Ivan Bratko[1,2]

[1]J. Stefan Institute, Artificial Intelligence Lab., Jamova 39, 61000 Ljubljana, Slovenia

Phone: (+38)(61) 1773 272, Fax: (+38)(61) 1258-158

[2]Faculty of Electr. Eng. and Computer Sc., Tržaška 25, Ljubljana University

[3]Brunel University, Dept. of Computer Sc., Uxbridge, England

E-mail: Dunja.Mladenic@ijs.si, http://www-ai.ijs.si/DunjaMladenic/home.html

Abstract

Understanding of discrete event simulation systems is a difficult task. Our approach involves consultation with a domain expert, and the use of discrete event simulation model and machine learning as tools for the intelligent analysis of simulated system. Current methods for the analysis and interpretation of such systems are restricted to statistical techniques that say much about the reliability of an output, but little about the output inter connectivity. The objective of our work is to improve the ability to interpret the model to the level of explanation that might loosely be described as "How the simulated system works". The new interpretation techniques are based on machine learning that includes classification tree chaining. "How the simulated system works" is important for a decision maker's understanding of the system in terms of relationships between the various parameters, the utilisation of resources and the location of (potential) bottlenecks. This sort of understanding is important when managing queuing problems in production systems, or when planning a new system. The output from a discrete event simulator was processed as machine learning problem. In these experiments, the simulation output was interpreted by attribute-based learning system Magnus Assistant. The knowledge gained from the machine learning tools was evaluated by the domain expert, who considered that this knowledge contributed to an understanding of the system, and how it might profitably be modified.

keywords: data analysis, discrete event simulation

1 Introduction

Discrete simulation involves several stages: design and implementation of a model for the system under study, performing experiments with the simulator, analysis and presentation of experimental results. The analysis and presentation of discrete event simulation results is usually done intuitively. Deriving useful conclusions from the simulation results is a demanding task. In this paper we propose automatisation of this task by means of machine learning techniques. Our approach involves consultation with a domain expert, the use of discrete event simulation model and the machine learning as an intelligent analysis of the simulated systems (see Bratko, Mozetič, Lavrač 1989). It should be emphasised that the goal of the presented work is not a prediction nor an optimisation. The main goal here is to find interpretations of the simulation output and discovering regularities, thus helping the user to develop an intuitive understanding of the domain (see Fig. 1).

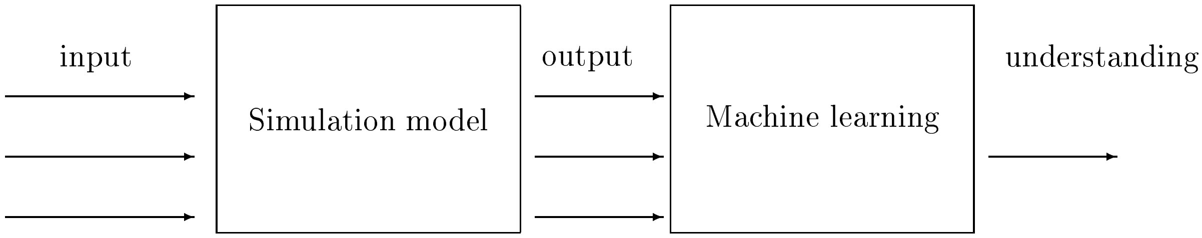

Figure 1: Using machine learning for simulation model analysis

Discrete event simulation produces example situations that can be used as input data for machine learning tools. In the presented research, a non-trivial and commonly used discrete event simulator was interpreted using machine learning tool. Some simple discrete event simulators were already analysed using machine learning techniques but in a slightly different way (see Mladenić et al. 1994). The attribute based learning system Magnus Assistant (Mladenić 1990) was chosen as appropriate for learning in noisy, real-world domains.

Section 2 describes the proposed classification tree chaining method. Section 3 describes the discrete event simulation domain used in our experiments with machine learning. The experiments and the proposed methodology are discussed in Section 4.

2 Classification tree chaining

The machine learning system we used for the data analysis is Magnus Assistant, based on the Assistant (Cestnik, Kononenko and Bratko 1987), algorithm for the construction of binary classification trees. It belongs to the Top Down Induction of Decision Trees (Quinlan 1986) family of algorithms.

The main goal is to find interpretations of the simulation output and discovering regularities in the data. The idea is to describe the target situation (simulation model) from different view points. For each target problem formulation classification tree was generated. Generated trees were used to design further experiments, where each node-split was considered as a Boolean class of some new problem. For example, the classification tree in Figure 2 enables automatic formulation of the following problems: explain when $A_1 = V_{11}$ and explain when $A_2 = V_{21}$, whose solution (classification trees) will give some more insights into the target problem (tree in Figure 2).

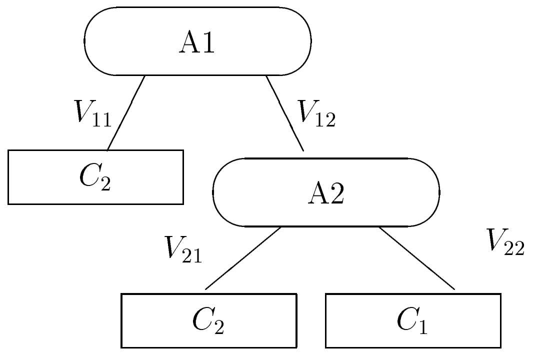

Figure 2: Example classification tree used for tree chaining.

The result is a set of classification trees that are "chained" and give some interpretation of the target problem. This approach enables a more flexible analysis of the simulator from different, dynamically chosen, view points. The developed methodology of tree chaining can be described with the following algorithm:

input: problem definition (eg. domain with class and attributes)
begin
generate description for chosen problem (eg. classification tree);
if domain expert is satisfied with the description(s) generated so far **then** finish
else
  repeat
    select a node-split from the current description to be a class for a new problem;
    exclude the selected node attribute from the attribute set;
    recursively repeat the whole algorithm on the new problem;
  until all nodes from the description are explained;
endif
end
output: set of descriptions (eg. classification trees)

The algorithm is recursive but the depth of recursion depends on the domain expert. If the set of classification trees describes the target problem sufficiently or no interesting tree can be obtained, the recursion should stop. In our experiments, there are many equally informative attributes that enabled us to generate different classification trees of the same quality and size; and it turned out that in the above algorithm recursion depth one is a good choice (the amount of new knowledge domain expert gained from the deeper recursion was law).

3 Domain descriptions

Discrete event simulator of steelworks was used in the development of a methodology for classification tree chaining. The simulator was used with the run-in time parameter set to 4000 minutes and the simulation duration set to 10000 minutes (circa 7 days). The simulators were generated using *The Extended Lancaster Simulation Environment eLSE* (Paul and Balmer 1993). The interaction of objects for each example simulator are described by *Activity Cycle Diagrams* (Fig. 3). These diagrams use two types of objects: circles to denote queues and rectangles to denote activities. There are some additional conditions incorporated in the simulators that are (for the purpose of simplicity) not included in the diagrams.

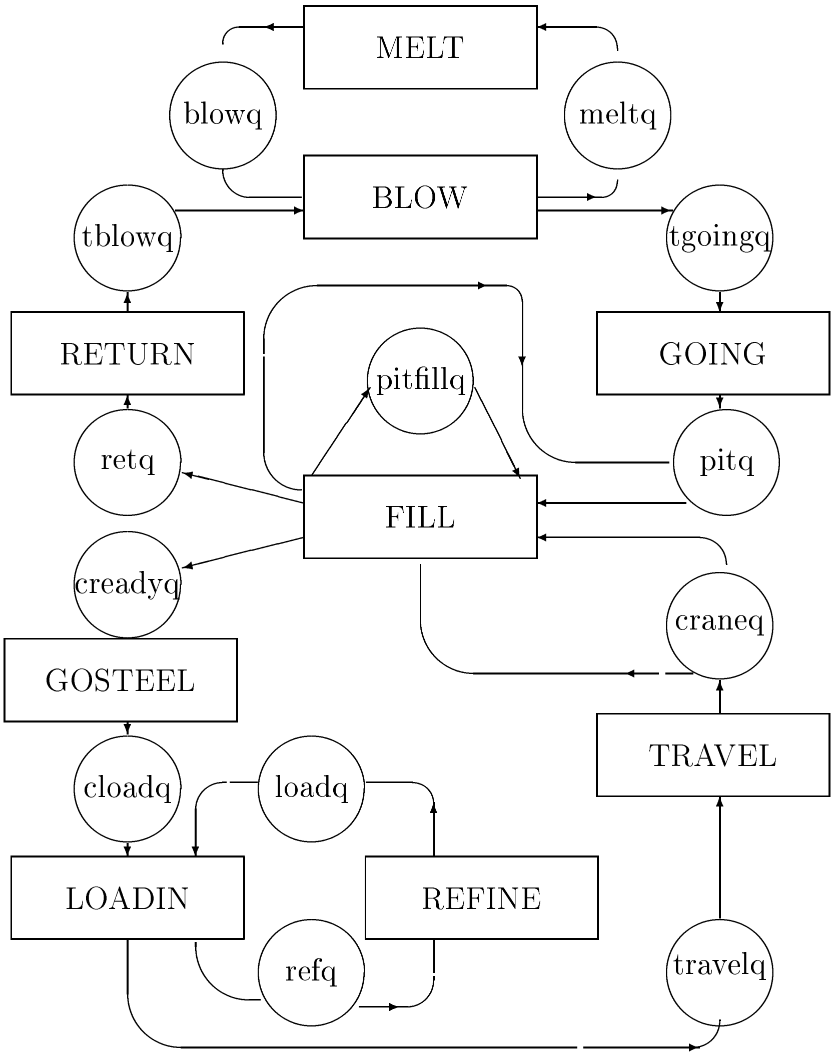

Figure 3: Activity Cycle Diagram for the steelworks simulation

In the steelworks simulator (Fig. 3) molten iron is cast in one of two blast furnaces. When a blast furnace empties, the molten iron is put into one or more railway trucks called torpedos (if sufficient are available, currently 2) and transported to an unloading area called a pit. If torpedo is not available, the molten iron is dropped on the floor and waste is produced. At the pit a crane is loaded with exactly enough iron to charge a steel furnace. When a steel furnace is empty and a crane is properly loaded, the crane moves to the steel furnace, loads the steel furnace, and then travels back to the pit. The steel furnace converts the iron to steel after a time, and is then available again. All times in the system are sampled from stochastic distributions, and the transfer of iron is in variable quantities. Whilst ostensibly one might be trying to determine the number of torpedos and cranes such a system should have, it is in fact more useful to know why such combinations produce the statistical results that they do. There are non-linearities in the system that statistics does not explain, and above average common sense rarely gets there (about 5% of students, who are bright for their equivalent peer groups, find the cause of the major non-linearity). Each discrete event simulator has parameters that can be changed by the user and some statistical indicators calculated by the simulator. All the attributes can be grouped as follows:

input to simulator
- number of entities; blast furnaces (set to 2), pit (set to 1), steel furnaces (set to 5), `Cranes` (varied from 1 to 5), `Torpedo` (varied from 1 to 12)

output of simulator
- utilisation of entities; `BlastUtil`, `TorpUtil`, `PitUtil`, `CraneUtil`, `SteelUtil`
- number of activations of each activity
- nontrivial statistics for time and length of each queue

4 Experiments and results

There are several interpretations generated for the steelworks system using the same simulator. Some of the typical interpretations generated by the students are as follows:

- The results obtained from the computer model show that operating two cranes instead of one has the most impact in reducing the amount of waste iron. Using two cranes and two torpedos will result in a 60% reduction in the amount of molten iron wasted in comparison with a single crane and the same number of torpedos.

- It was felt that in comparing the system with different configurations a two crane system using 6 torpedos was far superior and is highly recommended. Even when using one crane with 8 torpedos the system will generate approximately 12% of wastage in comparison with 0% in the case of two cranes.

The main disadvantage of the system interpretations generated without machine learning tools is lack of explanation "How the simulated system works". Provided knowledge hardly goes beyond descriptions concerning mainly optimisation problem (optimal number of cranes and torpedos considering waste). Using machine learning tools for data analysis enables deeper understanding of the simulated system. In our experiments discrete event simulator produces example situations that were used as input for different machine learning problems.

Combining all interesting inputs to the simulator different (values for number of torpedos and cranes), domain was generated having $5 \times 12 = 60$ examples. We decided to analyse the steelworks simulator using user-defined and statistical parameters, in the light of waste of melted iron that occurs during the simulation period (circa 7 days). The amount of waste is continuous but domain expert emphasised that only several levels of waste are interesting. We wanted to explain the occurrence of the waste at the following levels: 0, 200, 500, 1000, 1500, 2000, 5000, 10000, 20000, 50000. From the above parameters, several Boolean class learning problems were formulated (i.e. Class value > threshold). For this purpose, additional machine learning domains were constructed using a dynamic selection of attributes and class, namely classification tree chaining described in Section 2.

On the same simulator output, several machine learning problems were defined to enable analysis from different aspects. Knowledge generated in a such way provides more complete understanding of the simulated system. This knowledge is currently represented with a set of classification trees that are very simple even if they exactly match the data (no pruning was used). A strong correlation between the attributes enables equivalences (from the system's point of view) of different explanations for the same situation. These equivalences appear in the tree construction process as the node-splits (attributes) with equal informativity. Notice that for such data analysis (tree construction), system calculates a quality for each attribute and user selects the actual attribute to be used in the current node. This turned out to be very beneficial, since some attributes could be preferred by domain expert, although equally (or even less) informative. On the other hand, all equally informative attributes could be used at each level for generation of different classification trees. For example, for domain with class `Waste = 0` and for each tree node, there is a set of equally informative attributes:

1. level node (root)	2. level node	3. level node
`tblowqTime > 65.64`	`tblowqTime <= 117.5`	`cranes > 3`
`tblowqLen > 2.18`	`tblowqLen <= 3.89`	`craneUtil <= 0.31`
		`pitqTime <= 3.9`
		`pitqLen <= 0.37`
		`craneqTime > 12.54`
		`craneqLen > 1.2`

The set of equally informative attributes for the root of the tree (`tblowqTime > 65.64`, `tblowqLen > 2.18`) means that selection of any of the give node-split is equivalent for the machine learning system. Thus, we can generate a classification tree for each of the attributes from the set (see Fig. 4).

The left most part of the left hand side tree in Fig. 4, shows that waste is greater than 0 if empty torpedos are not waiting long (≤ 65.64). With longer (but still not long enough) waiting time (≤ 117.5) the number of cranes becomes important (`cranes` ≤ 3). The right most branch of this classification tree guarantees waste $= 0$ under the condition of enough empty

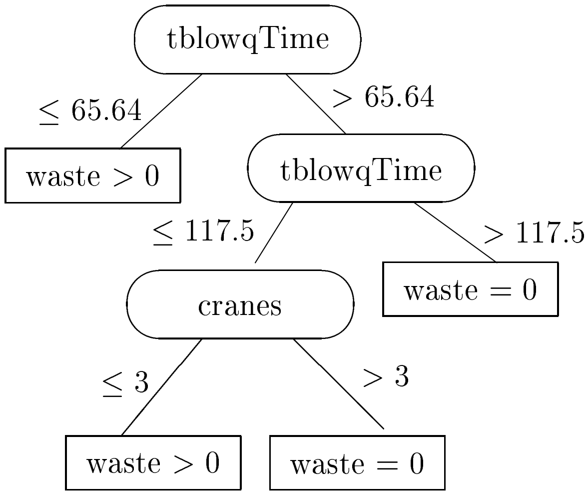

Figure 4: Classification trees for waste threshold 0 generated by Magnus Assistant.

torpedos (waiting time > 117.5). Equivalent to the torpedos waiting time is, as expected, the torpedos queue length (see level 1 and level 2 above). Domain expert found the equivalences for the number of cranes (or crane utilisation) nontrivial and thus more interesting (see level 3 above). The lower number of cranes, the higher their utilisation (`cranes > 3` is equivalent to `craneUtil < 0.31`) and the lower number of cranes, the shorter empty cranes queue (`cranes > 3` is equivalent to `craneqLen > 1.2`) (or waiting time: `craneqTime > 12.54`). In the first moment, domain expert was surprised by the connection of the queue of full torpedos and the number of cranes in the system. Short queue of full torpedos waiting for pit and cranes (`pitqLen <= 0.37`) is equivalent to the greater number of cranes. Originally, short queue of full torpedos indicates fast fluctuation of full torpedos that can appear only if the number of cranes is sufficiently high.

At this point, the developed methodology explained in Section 2 is applied. The new problem with the Boolean class `tblowqTime > 65.64` is considered with the same set of attributes (except `tblowqTime`) and classification tree for it generated. The tree is very simple, having only one node indicating that empty torpedos are waiting more that 65.64 minutes if action of filling crane from torpedo occurs more than 558 times (`tblowqTime > 65.64` if `fillAct > 558`); in the selected simulation period of circa 7 days. There is again number of equally informative attributes at the root level: `fillAct > 558, returnAct > 195, loadinAct > 375, travelAct > 375, refineAct > 375, pitUtil > 0.46, tblowqLen > 1.88, pitfillqTime <= 5.69, loadqTime <= 14.72`.

This explanation can be "chained" with the previous knowledge (classification trees). Instead of just saying that waste is greater that 0 if empty torpedos are not waiting much (`tblowqTime <= 65.64`), it can be added that in this situation fill activity occurs less than 558 times or return activity occurs less than 195 times etc. (we are considering the above attributes). In further experiments `fillAct > 558` is considered as a Boolean class. There is

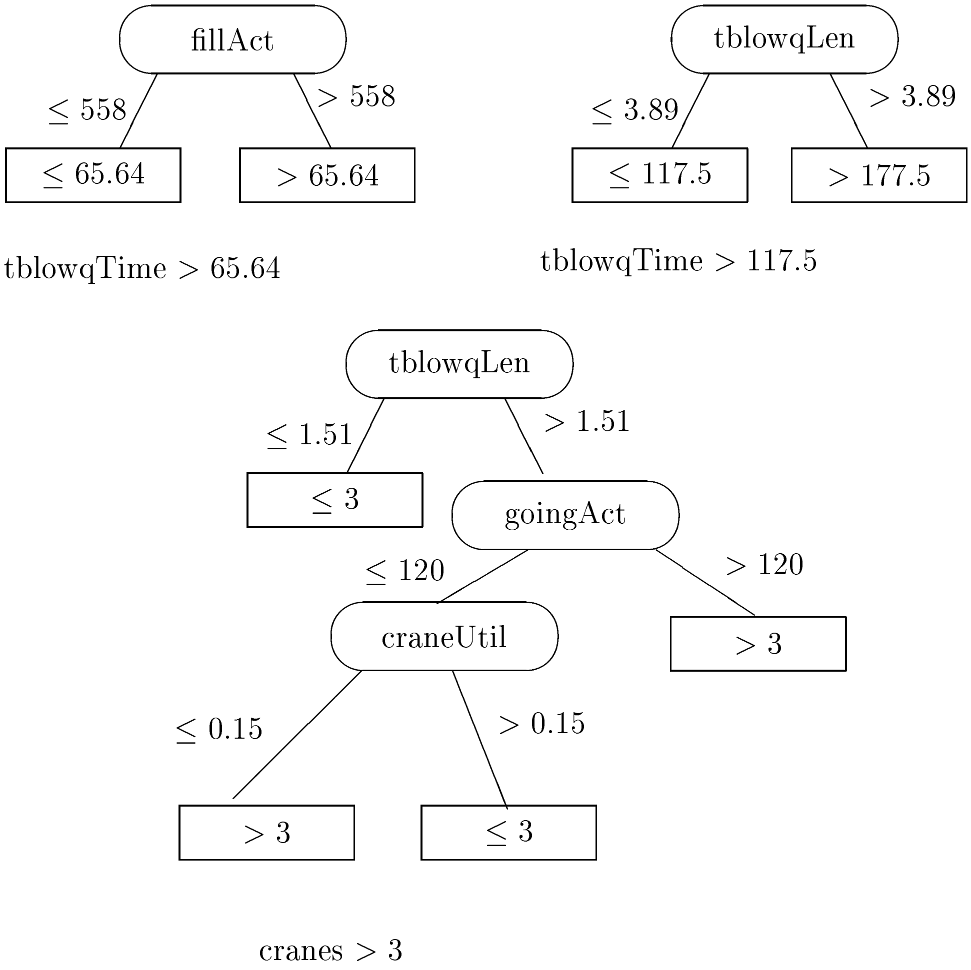

Figure 5: Classification trees for `tblowqTime > 65.64, tblowqTime > 117.5, cranes > 3` generated by Magnus Assistant.

again a set of equally informative attributes: `returnAct > 195`,`loadinAct > 375`, `travelAct > 375`, `refineAct > 375`,`pitUtil > 0.46`, `tblowqLen > 1.88`, `pitfillqTime <= 5.69`, `loadqTime <= 14.72`. The tree generated with any of these best attributes in the root is very simple. For example, taking the first attribute as the root the generated tree can be interpreted as follows: if `returnAct > 195` then `fillAct > 558`; if `returnAct <= 195` then `fillAct <= 558`. Using the same methodology, explanations for `tblowqTime > 117.5` and `cranes > 3` were generated (see Fig. 5).

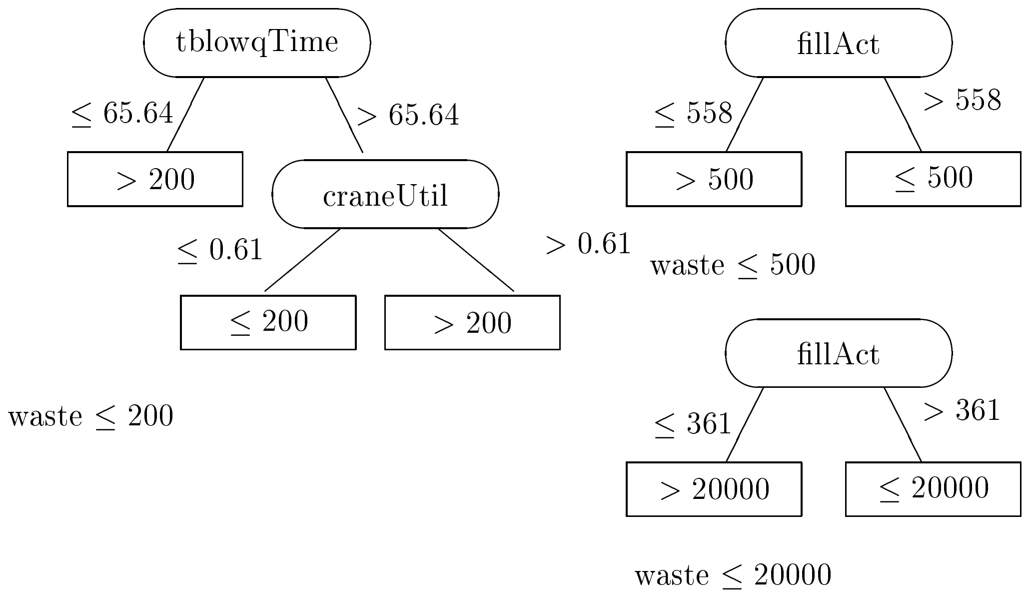

Figure 6: Classification trees for waste thresholds 200, 500, 20000 generated by Magnus Assistant.

Experiments with waste threshold 200 resulted with similar trees (see Fig. 6). Attribute `tblowqTime > 65.64` is again the most important and combined with `craneUtil <= 0.61` (higher than in previous problem `Waste = 0`) results with `Waste <= 200`. In all other situations, the waste is predicted to be greater than 200. Notice that in the problem for `Waste = 0` lower crane utilisation is required (`craneUtil <= 0.31`) for no waste. For both problems, the same attribute with the same threshold `tblowqTime > 65.64` seems to be the most important and its explanation would help in understanding the simulated steelworks.

Experiments with waste threshold higher than 200 resulted with very simple trees (see Fig. 6) explaining waste with fill or going activity or some of their equivalences.

5 Conclusions

In the paper the classification tree chaining has been investigated to help in the analysis of discrete event simulation system. The knowledge gained from the machine learning tools was evaluated by the domain expert, who considered that this knowledge contributed to an understanding of the system, and how it might profitably be modified. This is important because the success of the present study should not be measured in terms of prediction accuracy, but in terms of the interpretative ability of the induced descriptions.

The knowledge provided by classification trees generated for the target problem (waste in steelworks) was analysed by domain expert and decided to be helpful but not enough detailed. Thus, depending on the already generated classification tree, a new target problem was formulated and the experiments repeated. This approach enables a more flexible analysis of the simulator from different, dynamically chosen, views and was considered by domain expert to be very beneficial for the simulated system understanding.

6 Acknowledgements

This work was partially supported by the Slovenian Ministry for Science and Technology and the British Council in Slovenia.

References

[1] Cestnik, B., Kononenko, I. and Bratko, I., (1987), ASSISTANT 86: A Knowledge-Elicitation Tool for Sophisticated Users, in *Progress in Machine Learning*, ed. by I.Bratko and N.Lavrač, Sigma Press, Wilmslow.

[2] Mladenić, D. (1990) Machine learning system Magnus Assistant (in Slovene). Unpublished *BSc Thesis*, University of Ljubljana, Faculty of Electrical Engineering and Computer Science.

[3] Mladenić, D., Bratko, I., Paul, R.J., Grobelnik, M. (1994) Using machine learning techniques to interpret results from discrete event simulation. *Proceedings of the European Conference on Machine Learning ECML94*. Springer-Verlag 1994, pp. 399–402.

[4] Paul, R.J., Balmer, D.W. (1993) *Simulation Modelling*, Chartwell-Bratt, .

[5] Quinlan, J.R. (1986) Induction of Decision Trees, *Machine Learning Journal*, Vol. 1, pp. 81-106.