Consultant-2 : Pre- and Post-Processing of Machine Learning Applications

D. Sleeman M. Rissakis, S. Craw*,
N. Graner, S. Sharma

Computing Science Department,
University of Aberdeen,
Aberdeen AB9 2UE, Scotland, UK

To Appear in Int. J. Human-Computer Studies

Abstract

The knowledge acquisition bottleneck in the development of large knowledge based applications has not yet been resolved. One approach which has been advocated is the systematic use of Machine Learning (ML) techniques. However, ML technology poses difficulties to domain experts and knowledge engineers who are not familiar with it. This paper discusses Consultant-2, a system which makes a first step towards providing system support for a *"Pre- and Post- Processing"* methodology where a cyclic process of experiments with an ML tool, its data, data description language and parameters attempts to optimise learning performance.

Consultant-2 has been developed to support the use of the Machine Learning Toolbox (MLT), an integrated architecture of ten ML tools, and has evolved from a series of earlier systems. Consultant-O and Consultant-1 had knowledge only about how to choose an ML algorithm based on the nature of the domain data. Consultant-2 is the most sophisticated. It, additionally, has knowledge about how ML experts and domain experts pre-process domain data before a run with the ML algorithm, and how they further manipulate the data and reset parameters after a run of the selected ML algorithm, to achieve a more acceptable result. How these several KBs were acquired and encoded is described. In fact, this knowledge has been acquired by interacting both with the ML algorithm developers and with domain experts who had been using the MLT toolbox on real-world tasks. A major aim of the MLT project was to enable a domain expert to use the toolbox directly; i.e. without necessarily having to involve either a ML specialist or a knowledge engineer. Consultant's principal goal was to provide specific advice to ease this process.

*Seconded to the MLT project at the University of Aberdeen, 1990/91; permanent address, The Robert Gordon University, Aberdeen

1 Introduction

Machine Learning (ML) has been seen as one approach to overcome the classical knowledge acquisition bottleneck in the development of knowledge based applications [4]. Originally, it was envisaged that the principal task which would face the domain specialist would be the choice of an appropriate ML algorithm. This was the position which we held when we began the MLT (Machine Learning Toolbox) project in 1989. (Section 2 gives more details of the several aspects of the project.) Thus one of the objectives of the MLT project was to produce an advisor program, we called it a Consultant, to give such advice. Additionally, as part of this project, we observed ML specialists and domain experts using ML systems on a number of industrially important tasks and we soon realised:

- the examples to be used for learning often have to be "massaged" into a form suitable for a particular Machine Learning algorithm; and
- the domain expert often is unhappy with the results produced by a Machine Learning algorithm, and hence results also have to be "massaged" or the algorithm re-run with different data and parameters once the initial results are obtained. So we have realised that the processing of data by a ML algorithm is not a single-step, but a cyclic process, [15].

The first version of the Consultant implemented, Consultant-O [16], basically gave advice on the selection of a ML algorithm from a pre-specified set of algorithms. The knowledge needed to make this decision had been acquired from both algorithm experts and domain experts who had used ML algorithms. (This knowledge was essentially standardised by the developers of the advisory system.) The next advisory system produced, Consultant-1 [5], had essentially the same knowledge base but had a better interface, allowed the user access to a glossary, as well as an example-bank, and most significantly it allowed the user to modify decisions and inputs provided as his/her knowledge of the domain improved. The version of the Consultant, Consultant-2, discussed in this paper is an enhancement of Consultant-1. In addition, it also tries to provide support for the user in preparing his data and secondly support when the results of the initial run with the algorithm do not satisfy the expert. Thus it attempts to provide support for the 2 points above.

When the advisory system was first conceived, it was intended that these systems should make it possible for domain experts with no knowledge of Machine Learning to use the Consultant, and thereby use the toolbox to acquire knowledge about his/her chosen task domain. Given the difficulties which we encountered with Consultant-0, we weakened our position slightly, and our objectives became that of providing an advisory system and toolbox which could be used by knowledge engineers (who again would have no detailed knowledge of ML). In section 6, we summarise the types of users who have actually used the Consultants, and to what effect.

The paper is structured as follows. Section 2 describes the context of the system within the MLT project, the available ML tools and the Case Studies. Section 3 discusses methodological issues of using ML, the Pre- and Post-Processing methodology, and the functional specification of Consultant-2. The implemented architecture of the system is presented in section 4. The knowledge incorporated in this architecture is then outlined in section 5. Section 6 shows a particular run (a protocol) for Consultant-2 in a medical domain; this protocol particularly illustrates the post-run phase. Section 7 outlines an alternative perspective of the system in terms of learning bias. Finally, section 8 presents conclusions and discusses future directions.

Figure 1: **The Machine Learning Toolbox Architecture**

The architecture of the MLT is outlined in figure 1. The main parts of the toolbox are:

the learning tools (MLT tools): ten ML tools with a common Human-Computer Interface (HCI), a Common Knowledge Representation Language (CKRL), and Programming Interfaces (PIs),

the Consultant: a central advice-giving expert system for supporting the use of the MLT, and

other general purpose tools: an evaluation module, an example post-processor and a CKRL editor.

The learning tools have been selected to provide a wide range of learning capabilities. There are knowledge-free induction algorithms (NewID, CN2, LASH), knowledge-intensive concept learners (Cigol, Makey), symbolic clustering algorithms (KBG, DMP), statistical data analysis tools (SICLA) and more integrated systems for the acquisition, modelling and refinement of knowledge (Mobal, APT).

The diversity and complexity of the toolbox has raised several issues which are important from the practical and research viewpoints: the selection of appropriate tools, their actual usage and various knowledge integration issues. The selection and initialisation of tools is addressed by the Consultant and will be discussed in the rest of this paper. Knowledge integration issues are addressed by work on the Common Knowledge Representation Language (CKRL); for details see [10].

The iterative process of building the MLT permitted its developers to undertake ongoing evaluation of the MLT tools, the Consultant and the CKRL using the Case Studies as a testbed. Thirteen Case Studies exist [8], taken from the following application areas:

- Fault diagnosis in electronic devices and networks
- Computer aided design
- Temporal sequence prediction
- Image recognition and clustering
- Medical diagnosis and planning.

Given the sophistication of the individual MLT tools, and the complexity of the target applications there is a need for an advice-giving system to support the use of the MLT; three versions of the Consultant have been built to date[1].

The goal of Consultant-0, [16], was to support the selection of a suitable MLT tool. This was viewed as a classification task; it takes as input a description of the task and the application data, and discriminates among the ten classes (the ten MLT tools). Consultant-0 was built as a simple classification expert system using the Nexpert^{TM} shell. Substantial interviewing with ML experts and domain experts took place to acquire the knowledge embedded in the system.

Consultant-1 [5, 6] was built after an extensive critical analysis of Consultant-0. The discrimination knowledge of Consultant-0 was judged satisfactory and so this knowledge was only slightly revised for Consultant-1. On the other hand, Consultant-0 was substantially criticised for rigidity during its interaction with the user, as it did not allow the user to revise answers once the overall task had become better understood by the domain expert. The architecture of the system was redesigned to provide flexibility during the acquisition of task- and data- descriptions. A help system was also added to provide the necessary ML-specific knowledge that the intended user often lacked.

In Consultant-2 (C-2), the functionality of the system is further enhanced to support the pre- and post-processing phases discussed earlier. This functionality is discussed in the next section. We should mention that, although the new functionality represented a substantial increase in the scope of the system, there was, surprisingly, little need to change the overall

[1]In the rest of the paper, when the version of the MLT or the Consultant is not explicitly stated, the latest one, MLT-2 or Consultant-2, is implied.

architecture of the system from that used in Consultant-1; the increased functionality was mainly achieved by acquiring and embedding functionally different knowledge in the system.

3 The Functionality of Consultant-2: Pre- and Post- Processing

ML tools can, in some circumstances, be used to extract knowledge required for the development of knowledge based systems. When this is done, it can radically reduce the development time of the applications. This contribution of the ML field to knowledge acquisition can be substantial, but we are still quite far from completely automating such tasks. It has been realised that the process of using ML tools is not as straight forward as was initially expected. Thinking of an ML tool as a black box, that can be applied, once, in a straight forward manner to a data set, to produce useful and possibly complete knowledge, is certainly appealing, but rather unrealistic, when it comes to real world tasks. Working with the MLT Case Studies showed that the right way to view the development process is not as a one-off application of an ML tool but as a *cyclic* process that closely resembles the traditional process of systems development. We have abstracted our experience gained from the Case Studies, as to what is typically done during this cyclic process, into a "methodology for applying ML on real world problems". The methodology was first published in [8], and is outlined in figure 2.

1. Decide on the appropriate problem solving method [3, 19].
2. Decide on the appropriate problem solving tool; this decision will imply specific data structures as the desired format of the output of the learning process.
3. Decide on the appropriate ML tool, based on descriptions of the task, the data and the desired format of the output.
4. Use data transformation tools to adapt the data format.
5. Use data manipulation tools to form the training set.
6. Given the task and data description, select promising values for the ML tool's parameters.
7. Run the ML tool.
8. Evaluate results using statistical, knowledge-based or intuitive methods.
9. If evaluation is satisfactory, then FINISH.
10. Otherwise, modify some of the previously made decisions on tool parameters, training data, data representation, background knowledge, ML tool, problem solving tool and/or problem solving type AND goto step 7.

Figure 2: A methodology for applying ML on real world problems

In the rest of the paper we will use the term *"Pre-Processing"* to refer to all those actions undertaken to prepare the application for the *initial* use of the ML tool; i.e., the first six steps of the methodology. The term *"Post-Processing"* will refer to steps 8 to 10 i.e. the steps of evaluating the learning process and taking actions for improving it. The term *"Pre- and*

Post-Processing" will refer to the whole of the methodology, excluding actually running the ML tool.

It is clear that automated support for such a methodology would be of significant academic interest and would have significant industrial value. Initial steps towards this goal have been taken by building tools to assist subtasks within the overall process. Building an integrated evaluation package (LEM) is reported in [12]; a program for interactive data Post-Processing (SMILE) is described in [11]; and the WILA workbench for supporting a user throughout Pre- and Post-Processing with simple attribute-value ML tools is described in [17].

Consultant adopts a pro-active approach, rather than just a user-supportive role. The user is still present, and can always reject suggestions made by the system during the process, but it is the Consultant rather than the user that drives the process. There has been a gradual shift of responsibilities to the system, in the series of Consultants built. Consultant-0 performed only step 3 of the process; Consultant-1 performed step 3 and supported steps 4-6; while Consultant-2 attempts to automate steps 3-10. Consultant-3 is currently being specified to extend the scope of advice giving by also including steps 1 and 2.

The first stage in implementing Pre- and Post- Processing facilities was to identify clearly the factors that are manipulated during the two processes. We will refer to this list as the *L-factors* [2]. The ones we have identified as the most important are:

- The problem solving method chosen; i.e., whether the problem solving task is basically classification, propose-and-refine, optimisation, etc, [3, 19].
- The problem solving tool; this is linked to the previous factor, and specifies the actual algorithm to be used to solve the task.
- The ML tool; the algorithm chosen to perform the learning. (This choice is influenced by the input data and the form of the output required.)
- The formalism of the data; ie, whether the data is for example propositional, or first order etc.
- The representation and content of the background knowledge, i.e., what formalism has been used to represent the background knowledge, and
- Specific values for the parameters of the ML tool.

It is interesting to notice that all actions taken during Pre- and Post-Processing, as outlined, in figure 2, can actually be seen as decisions favouring specific choices of the L-factors. Notice also that the *same* L-factors are manipulated both in Pre-Processing and Post-Processing. Hence, the required output of both Pre-Processing and Post-Processing could be specified as decisions favouring specific choices of L-factors. The difference between Pre-Processing and Post-Processing lies in the way such choices are made. Pre-Processing is a process of making initial choices; e.g. to use algorithm NewID, with the parameter "Pruning Threshold" set to 0.1, using the attributes {colour, size}. Post-Processing, on the other hand, is a process of updating existing choices, made as a result of previous runs; e.g. continue using NewID, but increase the Pruning Threshold to 0.15, and discretise the numeric attribute "size".

[2] L-factors is an abbreviation for Learning-Factors.

Figure 3: **The functional specification of Consultant-2**

It is intuitively useful to think of Pre-Processing and Post-Processing in figure 3 as respectively the initial setting and the subsequent modification steps of an iterative search for optimal learning performance. The implied optimisation problem is multi-variate because many L-factors are considered, and multi-functional since several evaluation criteria might be important for the application in hand. In order to converge rapidly, the decisions taken at each step of the search need to involve informative heuristics, and probably interaction with the user. The implementation of these requirements is discussed in the next section.

4 The Architecture of Consultant-2

Consultant-2's design is based on two sub-architectures, one for Pre-Processing and one for Post-Processing as shown in figures 4 and 5. The two architectures share a common, *"butterfly"*, architectural style and the same representation mechanism. Historically, the Pre-Processing architecture resulted from the architecture of the previous version of the system, Consultant-1, and is almost identical to it. The Post-Processing architecture was built as an *extension* of the Pre-Processing one, sharing its style and representation. The "butterfly" style of the two architectures has been imposed, in the first place, by a need to structurally acquire, and flexibly maintain, information coming from a *variety* of sources, and subsequently, the desire to use this information for a *variety* of analyses.

The Pre-Processing architecture, figure 4, focuses on the acquisition and usage of an explicit description of the application task and data, called the *Task Description*. The acquisition can be guided by the combination of three different modes of interaction with the user: fixed path, intelligent, or user-browsing. After acquiring the Task Description, a number of different "knowledge functions" can be applied to it, to yield several types of advice such as relative suitability of the MLT tools, providing explanations for choice of tools, comparing tools, etc. If a particular tool is now selected by the user, Consultant asks some more tool-specific questions and then applies other knowledge functions to the Task Description to yield plausible Pre-Processing initialisations for the relevant L-factors.

In the Post-Processing architecture, figure 5, the front-end Acquisition Controller manages the acquisition of information about the task, the data, the run of a specific MLT tool and the output of the MLT tool as provided by four different sources. The MLT tool provides tool-specific statistics about the run and its output. The evaluation module, LEM, provides more general evaluation metrics about the output. The user provides subjective and domain specific assessments. Finally, the Task Description that was acquired during Pre-Processing provides information about the task and the data.

The various types of information are combined by the Acquisition Controller to build a structured description of the run and its output, called the *Run Description*, and a new updated version of the Task Description. These descriptions are then used to provide advice on how to improve the next run. The heuristic knowledge required for this task is provided by the Knowledge Base (KB). The potentially large number of advice fragments are processed by the Advice Controller to yield the final recommendation. As discussed earlier, these pieces of advice are settings or modifications of choices made for one or more of the four relevant L-factors i.e. parameters, data, data description language, or ML tool.

Both the Pre- and Post-Processing architectures are focused on building and using three knowledge sources consisting of sets of descriptor-value pairs:

Task Description: explicitly describes the task and the data of the specific application,

Run Description: explicitly describes a specific algorithm run that has just terminated and its output,

L-factors description: explicitly describes the choices made (or to be made) for the L-factors during a specific run.

The role of Consultant-2's inference mechanism is to control the acquisition and use of these Descriptions. The meta-knowledge required for this control task is located in three knowledge bases, called *Generic Descriptions*:

Figure 4: **The Pre-Processing architecture of Consultant-2**

GTD : the Generic Task (and Data) Description,

GRD : the Generic Run (and Output) Description, and

GLFD : the Generic L-Factor Description.

Each descriptor in a Description is associated with a frame structure, which keeps meta-knowledge about the specific descriptor. As an example, figure 6 shows the frame associated with the GTD descriptor 'data.negative-examples'. Probably, the most important type of meta-knowledge included in the Generic Descriptions is dependency relationships among descriptors. Such dependency knowledge is coded as pre-condition- and propagation- slots in the descriptor frames. These dependencies support Consultant's structured acquisition of information.

5 The Knowledge of Consultant-2

Consultant-2 is a knowledge-based system and therefore the knowledge itself will largely determine the performance of the system. This section describes the knowledge embodied in

Figure 5: **The Post-Processing architecture of Consultant-2**

Consultant-2, i.e. the three Generic Descriptions, GTD, GRD, and GLFD, as well as the KB.

Acquiring this knowledge was quite a complex task, as there were ten different algorithm experts, and each of them had knowledge essentially of only his own algorithm. The Aberdeen team had the task of developing an appropriate terminology and trying to ensure consistency across the knowledge acquired from the several experts. Knowledge acquisition involved substantial interviewing, formalising the information, refining informal rule sets in order to build the KB, iteratively integrating and compressing the knowledge acquired from the individual experts, to construct the GTD, GRD and GLFD. For example, the first rule in Table 1 was developed by the Aberdeen team, from "snippets" of information provided by the several ML algorithm experts.

```
(PROPERTY data.negative-examples
     :conditions ((problem.binary-decision yes) (data.examples-available yes))
     :prompt "Can you provide both positive and negative examples?"
     :help "You are trying to discriminate among two options, e.g.
            whether a decision should be taken or not. If you can
            provide examples of when to take the decision as well as
            when not to take it (or objects that belong to a certain
            class as well as objects that don't, etc.), answer YES. If
            you only have examples of one type, answer NO."
     :topics (examples)
     :range YESNO
     :value-strings ("you can provide both positive and negative examples"
                     "you can only provide positive examples")
     :coefficients (()                                              ; if YES
                    (newid -1 cn2 -1 cigol -0.8 kbg -0.2 lash -0.5)) ; if NO
     :advice (()
              (kbg "If you only have examples of one class, you can
               use KBG but it will just find a generalisation of your
               examples." lash "LASH can in principle learn from positive
               examples only, but it performs poorly in this case.")))
```

Figure 6: A GTD frame (associated to the descriptor "data.negative-examples")

5.1 The Generic Task Description

The Generic Task Description (GTD) is a dynamic knowledge base containing a domain-independent set of descriptors for characterising the application task and data along with the meta-knowledge required for handling these descriptors. The GTD currently contains seventy-two descriptor frames.

GTD descriptors are used in two ways. Many of the descriptors are used only for discriminating among MLT tools during the tool *selection* phase. The remaining descriptors are used after a tool has been selected to provide initial settings for the parameters with respect to domain characteristics; e.g. setting a plausible value for a tolerance parameter given that the dataset is known to be noisy. Clearly, both types of descriptors are used during Pre-Processing. Post-Processing also makes use of GTD descriptors but mainly for constraining the applicability of Post-Processing rules in the light of special domain characteristics; e.g. if there is no noise in the data, *hypothesis consistency* parameters need not be updated.

Most GTD descriptors contain actual facts about the domain, but some of them describe desired facts: for instance, "domain examples are described by 10 attributes" is a fact about the dataset, while "the output should be a decision tree" is a desired, and possibly flexible, constraint on the output of the learning process. A list of the main groups of descriptors found in the GTD is given below, together with representative examples from each group:

application type: the problem solving type is classification, clustering, optimisation, etc.

input data: the number of classes, the number of examples, the main attribute type, whether all possible examples are available, estimation of noise in the data, etc.

representation of the input: whether attributes can be defined, whether data are relational in nature, etc.

background knowledge (BK): whether it is available, whether it is essential, its format, whether the closed world assumption holds, etc.

output: the preferred output format, whether readability is required, the expected number of clusters, etc.

5.2 The Generic Run Description

The Generic Run Description (GRD) is a knowledge base containing a set of descriptors for characterising the run of a learning algorithm and its output along with the meta-knowledge required for handling these descriptors.

There are currently 100 GRD descriptors. The instantiation of the GRD involves more complicated information management than that of the GTD, because, while most of the instantiation of the GTD is undertaken by the user, instantiating the GRD involves additional information sources (see figure 5). A particular measurement that relates to the output (e.g. tree size) might be provided either by the algorithm or by the Learning Evaluation Module (LEM). It might then be used for inference (e.g. if the "tree size" is less than 4 then the "tree size judgement" is set to "too small"). Or it might be assessed by the user, e.g., the user is asked to judge whether the value 4 is large or small for the descriptor tree.size for the specific domain. Hence, some GRD descriptors hold exact measurements about the run and output, while others hold user (or system) *assessments*. A list of the main groups of descriptors to be found in the GRD is given below, together with representative examples from each group:

the outcome of the run: whether the run was interrupted, the time spent for learning, whether the memory was filled up, etc

the output (in general): the actual format of the output, complexity assessment of the output, comprehensibility assessment, the exact coverage achieved, the exact accuracy achieved, user assessment of the accuracy achieved etc.

tree (or graph) output: the minimum/maximum/average path length, the number of leaf nodes, tree branchiness, overall size assessment, etc.

clusters output: number of clusters, average cluster size, average centre distance, etc.

rules output: the number of rules, the maximum path length, the average significance of rules, etc.

5.3 The Generic L-Factors Description

The Generic L-Factor Description (GLFD) contains explicit descriptions and knowledge about the L-factors; i.e., every factor that can affect learning performance and can be controlled by Consultant-2.

Several slots of the GLFD frames are used to store algorithm parameters. The slots describe parameter-specific knowledge such as the range of valid values of a parameter, information on typical values, default values and knowledge about how to update the suggested value when such an action has been decided.

The GLFD contains descriptors for the four L-factors handled by Consultant-2: MLT tool, data, data description language, and parameters for the chosen algorithm. The last category

of descriptors is the largest within the GLFD. We will further divided this category into groups, based on the way the corresponding parameters affect learning performance. This is usually done by imposing constraints on the way the tool generates and tests hypotheses. Learning tools have traditionally been seen as programs that search a space of hypotheses each of which can potentially and approximately fit the data, [9]. The way this space is constructed and searched has a large impact on the learning performance and is sometimes controlled by external parameters. Based on their role, the parameters represented in the GLFD can be categorised into:

- parameters that impose constraints on the language in which hypotheses are made,
- parameters that specify when a hypothesis is to be rejected or accepted,
- parameters that impose a preference (preferred order) among hypotheses,
- parameters that specify the strategy used during the search for valid hypotheses, or
- parameters that specify the role of background knowledge in the generation of hypotheses.

5.4 The Knowledge Base

Having described the GTD, GRD and GLFD, we can now present an overview of the heuristic knowledge of Consultant-2 that links situations described by Task- and Run- Descriptions to appropriate actions on the L-factor Descriptions. We have chosen to represent this expertise as rules. The premises of these rules are conjunctive conditions on the values of GTD and GRD descriptors. The rule actions consist of actions applied to L-factors. The actions either set a plausible initial choice for an L-factor or modify an existing one. Each rule has an attached coefficient which represents the certainty associated with the corresponding suggestion. When more than one rule is applicable, the certainties are combined to yield the overall certainty for the recommendation. The conflict resolution module bases its decisions on these several certainties.

Consultant's KB contains about 250 rules. Each of them is characterised by:

1. the phase in which the rule is applied: Pre-Processing or Post-Processing (i.e whether the rule action initialises or modifies a choice).
2. the L-factor affected by the rule, i.e. one of MLT Tool, Parameter, Description Language or Data.

These two "dimensions" define eight categories, summarised in table 1. Each cell of the table shows an example of a rule taken from the corresponding category. The rest of this section provides a general description of the rules included in each of these eight categories.

5.4.1 MLT tool

Pre-Processing: This category contains the rules used for the initial selection of a learning tool. This selection is based solely on the Task Description. 25 GTD descriptors are used, each of which contains evidence for, or against, the use of some MLT tools.
Post-Processing: Rules in this category would suggest modification of the choice of the

	Pre-Processing	Post-Processing
MLT Tool	IF examples cannot be represented as attribute/value pairs, THEN use Apt (0.2), use Cigol (0.2), do not use CN2 (0.8), etc.	*Left to the user.*
Parameters	IF you are using CN2, AND your examples cover the whole example space, THEN set the error estimate function to "Naive".	IF you are using Lash, AND the number of solutions found was too small, THEN increase the Maximal Search Depth by 1.
Description Language	IF you are using NewID, AND the possible values of a symbolic attribute are naturally ordered, THEN make this attribute numerical instead.	IF you are using CN2, AND an attribute has many possible values, AND you obtained too many rules, THEN try to merge some of these values.
Data	IF all possible examples are provided, AND you are using CN2, AND size of dataset >= 25000, THEN use SMILE to select the most informative examples.	IF the learned rules perform poorly on test examples, AND your examples are sparse in the set of possible examples, AND an expert can provide new examples interactively, THEN use SMILE to acquire new examples.

Table 1: Sample rules from Consultant's knowledge base.

MLT tool, after running it. This is usually advisable when a tool has been tried with many other types of L-factors none of which gives satisfactory results. However, Consultant does not currently include rules to make such a decision; this type of decision is left to the user.

5.4.2 Parameters

Once a tool has been selected, Consultant begins to search for optimal parameter values.
Pre-Processing: provides the starting values for this search. All MLT tools provide default parameter values which are suitable for many applications, but when information from the GTD allows Consultant-2 to select better values, the search is likely to be more efficient.
Post-Processing: modifies *previous* parameter values. Three types of modifications are suggested by rules in this category: set parameter to a specific value, increase or decrease value. The premises of these rules involve mostly GRD descriptors. This is the largest category of rules.

5.4.3 Description Language

Processing *Representations*. A learning tool usually requires its data to be in a particular formalism. Therefore, if the tool has been selected without regard to the formalism of the available data, changes of formalism will often be required; such changes are often complex, and can rarely be automated. The approach taken in Consultant is to use the formalism of the available data as a crucial constraint during the *selection* of an MLT tool. Thus Consultant only recommends a tool if it can use the available data with little or no modifications on

its formalism. There are therefore no rules in the knowledge base that explicitly recommend a change of formalism. On the other hand, the KB does contain rules which advise on the description language.

Pre-Processing phase: An initial description language can sometimes be suggested and the Consultant includes various rules for this.

Post-Processing phase: After running a tool, Consultant can often recommend the creation of new attributes, the merging of attribute values or other modifications of the description language that can result in better performance with the specified ML algorithm [3].

5.4.4 Data

Pre- and Post-Processing can also operate directly on data. *Manual* Data Pre- and Post-Processing can be performed using the WILA workbench [17]. *Semi-automatic* Data Post-Processing is performed by the SMILE scenario generator [11]. SMILE takes the output from an MLT tool and generates templates (scenarios) that describe potentially interesting, but unclassified, new examples. These can then be classified by an expert, that is acting as an oracle, and added to the example set.

6 A session with Consultant-2

The goal of this session is to suggest one out of 6 possible treatments for a patient with Maldescensous Testis; thus the task is essentially classificatory. Data and domain expertise has been provided by the FORTH Institute of Heraklion, in Crete.

The protocol has essentially three phases: pre-processing; selection of algorithm and setting of parameters; and post-processing. At several places in this interaction the system gives the user a series of choices and the user selects the one he/she wants by clicking on the appropriate option. To make this dialogue understandable we give the list of options and we show the user's reply (as if they had typed in their reply). The list of options are given in bold as are the Consultant's questions, which are also preceded by Q. The User's answers are preceded by A. Commentary on the dialogue are given in italics.

I. THE PRE-PROCESSING PHASE.

Q: Which of the following best describes what you want to do?

- **Group entities with similar properties**
- **Select the most appropriate from 2 or more options**
- **Optimise the performance of a particular system**
- **Predict recurring events from past behaviour**
- **Find an ordered sequence of actions to solve a problem**

[3] For simple propositional algorithms, such modifications of the description language together with the necessary data modifications can be carried out using the WILA workbench, [17].

- **Detect instances of a pattern in data**
- **Build or refine a model of a domain**

A: Select the most appropriate from two or more options

Q: How many options (classes) are there?
A: 6

Q: Are the classes mutually exclusive?

- **YES**
- **NO**

A: YES

Q: Could you provide examples from which an ML algorithm could learn?

- **YES**
- **NO**

A: YES

Q: Could you classify examples artificially generated by the system?

- **YES**
- **NO**

A: YES

Q: Can your data be noisy or inaccurate?

- **YES**
- **NO**

A: YES

Q: How many examples could you provide?
A: 200

Q: Can the examples be represented by attribute-value descriptions?

- **YES**
- **NO**

A: YES

Q: Do all the examples use the same set of attributes?

- **YES**
- **NO**

A: YES

Q: What type of attributes do you have?

- **Mostly numeric**
- **Mostly symbolic**
- **Mostly binary (Boolean)**
- **A mixture of different types**

A: A mixture of different types

Q: Is an expert available to evaluate interactively hypotheses formulated by a learning system?

- **YES**
- **NO**

A: YES

Q: Can you provide background knowledge to assist learning?

- **YES**
- **NO**

A: NO

Q: What kind of output would you like?

- **A decision tree**
- **A set of IF THEN rules**
- **A recognition function**

A: A set of IF-THEN rules

Q: Do you need the resulting rules (or tree or function) to completely cover the example space?

- **YES**
- **NO**

A: YES

CERTAINTIES after PRE-RUN session (*displayed on the workstation*)

- APT : -100%
- CIGOL : -91%
- CN2 : 84%
- DMP : -44%
- KBG : 25%
- LASH : -68%
- MAKEY : -100%
- MOBAL : -100%
- NEWID : 60%
- SICLA : -100%

II. CHOICE OF ML ALGORITHM AND SETTING OF ALGORITHM'S PARAMETERS.

At this stage the user could have chosen to execute any of the algorithms given above. Also, as introduced in Figure 4 he could have chosen to collect more information about the task, **or** *to have changed some of his earlier answers.*

This user, in fact, selected the CN2 tool, when one additional question was asked:

Q: Do you wish to monitor the algorithm run?

- **YES**
- **NO**

A: YES

The run with the CN2 tool then occurs and several record files are produced. Some of these are read by the Consultant after querying the user. The post-run phase then is entered.

III. POST-RUN PHASE.

Q: What is the status of the latest algorithm run in this session?

- **Completed**

- **Interrupted by the user**

A: Completed

Q: The elapsed time (35.69 secs) is:

- **Too long**
- **Satisfactory**

A: Satisfactory

Q: The number of rules (28 rules) is:

- **Too many**
- **Satisfactory**
- **Too few**

A: Too Many

Q: The average significance of the rules (9.43) is:

- **Low**
- **Satisfactory**
- **High**

A: Low

Q: The complexity of the learned result in general is:

- **Too complex**
- **Satisfactory**
- **Too simple**

A: Satisfactory

Q: The completeness of the learned result in general is:

- **Low, crucial knowledge has not been produced**
- **Satisfactory**
- **Quite complete**

A: Satisfactory

Q: The comprehensibility of the learned result is:

- **Difficult to understand**
- **Able to be understood**
- **Very easily understood**

A: Able to be Understood

Q: Does the expert want to include minority rules?

- **YES**
- **NO**

A: NO

Q: The accuracy of the learned result on the testing set (58.9 %) is:

- **Too high**
- **Satisfactory**
- **Too low**

A: Too Low

PARAMETER SUGGESTIONS (*post-run phase*)

The resulting parameter recommendations are:

- 0.72 evidence to increase Significance Threshold to 10.0
- 0.6 evidence to increase Star Size to 6
- 0.6 evidence to decrease Star Size to 4
- 0.5 evidence to set Maximum Class Forcing to TRUE

As shown above, two conflicting recommendations are suggested for the Star-Size. This is because the certainties are identical for each recommendation. One can resolve the conflict if one knows the reasons supporting each of the recommendations. Hence the user asked for an explanation for the star-size; based on this explanation, and given that the user in this session was mostly concerned about accuracy, he chose to increase the star-size to 6. CN2 was then rerun with the updated parameters. The new parameter settings suggested by Consultant-2 did in fact alter the performance of the algorithm, and a rule base resulted which had fewer rules and higher accuracy. The value obtained were in fact 20 rules with 71% accuracy.

Each version of the Consultant was thoroughly evaluated by partners who had primary responsibility for applications (namely BAe, AAR, Siemens & FORTH). Consultant-2 was used by each of these institutions; in the case of BAe and AAR, Consultant-2 was used by domain specialists. In the case of FORTH, due to pressure of time, the Medical expert did not use Consultant-2 directly but with a Knowledge engineer (a member of the MLT team).

The feedback from both groups of users on C2 was generally positive, [8].

7 A further perspective on Consultant-2: Learning Biases

Once the target ML algorithm has been decided, then the remaining L-factors can be thought of as biases for the execution of the algorithm. (For a definition of bias see [18].)

Early work in ML has been content to choose a single goal for the learning process, such as minimising the number of rules to achieve coverage, minimising the time required for the performance system to solve a range of tasks using the inferred knowledge, etc. Recently, the field has reported the study of tradeoffs between conflicting goals such as achieved accuracy vs. time required for learning, [13]. The MLT Case Studies, have led us to believe, that in real world tasks, more than two criteria need to be considered simultaneously. Under these conditions learning can be thought of as essentially a multi-criteria optimisation task; this is the perspective we discussed in [14].

8 Conclusions and future directions

Although clearly helpful, ML is not a "panacea" for knowledge acquisition. The process of using ML to develop large knowledge-based applications involves substantial decision making and experimenting with a number of factors, such as the L-factors, that can influence the success of the process. Introducing methodologies and advice systems, such as Consultant, can substantially increase the tractability of the process, can result in better and faster development cycles, and can make the technology available to a much wider range of potential users. We have described *Pre- and Post-Processing*, a first, simple methodology to support the processing of four specific L-factors: the ML tool itself, the data description language, the data and the tool's parameters. We also discussed our attempt to build a partially automated support system for the Pre- and Post-Processing methodology (Consultant-2), and described its functionality, architecture and the knowledge embodied in it.

Further developments may take various directions. Firstly, development can be focused towards further automation and tighter coupling with the MLT tools. The ultimate goal of this approach would be to build a "self-experimenting" version of the Consultant that would be able to autonomously "play" with different MLT tools, data, description language and parameter settings on a given application with a view to achieving some pre-specified goal.

Another idea is to experiment with new ML algorithms rather than with new subject domains. Here, the enhanced Consultant would attempt to automate the knowledge acquisition task performed manually for Consultant-2, by experimenting with a number of different applications. This approach, would aim at using automatic methods to induce rules about the new ML algorithms. Preliminary studies of this kind have been reported in [1] and [2].

Finally, possibly the most challenging direction, would be the development of methodologies and systems to support processing of the more difficult L-factors that were omitted from Consultant-2, namely, the selection of the *Problem Solver* and the *Data Formalism*. It is clear that the nature of the target Problem Solver *should* affect the ML algorithm used. The task of controlling the learning of required knowledge within a general framework is not trivial; for a discussion see [7]. Re-using or sharing the knowledge once it has been inferred is another issue. Modifying the data formalism, on the other hand, has also been identified as an important processing action in such subtasks as problem reformulation, changing the data to fit different ML tools, and integrating knowledge resulting from different learning processes.

9 Acknowledgements

Knowledge for the various versions of Consultant has been provided by other members of the consortium: Alcatel Alsthom Recherche (F), British Aerospace (UK), Foundation of Research and Technology – Hellas (Gr), Gesellschaft für Mathematik und Datenverarbeitung (D), INRIA (F), ISoft (F), Siemens AG (D), The Turing Institute (UK), Universidade de Coimbra (P), Université de la Réunion (F) and Université de Paris-Sud (F). In particular we thank Robert Davidge and Rüdiger Oehlmann, Aberdeen University, who implemented Consultant-0, and Chris Moore, British Aerospace, who implemented the knowledge functions for Consultant-1 and Consultant-2 and the natural language translator for Consultant's knowledge base. The development of Consultant is supported by the CEC as part of the Esprit II *"Machine Learning Toolbox"* project, P2154. Finally, we thank anonymous reviewers for their helpful comments.

References

[1] D. Aha. Generalizing from Case Studies: A Case Study. In D. Sleeman and P. Edwards, editors, *Proceedings of the 9th International Conference in Machine Learning (ML92)*, pages 1–10. Morgan Kaufmann, 1992.

[2] P. Brazdil, P. Gama, and B. Henery. Analyzing Results of ML Using Second Order Learning. In Y. Kodratoff and P. Langley, editors, *Proceedings of the ECML93 Workshop on "Real World Applications of Machine Learning"*. 1993.

[3] B. Buchanan, D. Barstow, R. Betchel, J. Bennet, W. Clancey, C. Kulikowsky, T. Mitchell, and D. Waterman. Constructing an expert system. In F. Hayes-Roth, D. Waterman, and D. Lenat, editors, *Building Expert Systems*. Addison-Wesley, 1983.

[4] J.G. Carbonell, R.S. Michalski, and T.M. Mitchell. An overview of Machine Learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning*, volume 1, pages 3–23. Tioga Press, Palo Alto, CA, 1983.

[5] S. Craw, D. Sleeman, N. Graner, M. Rissakis, and S. Sharma. CONSULTANT: Providing advice for the Machine Learning Toolbox. In M. Bramer, editor, *1992 BCS Expert Systems Conference*, pages 5–23. Cambridge University Press, 1992.

[6] N. Graner, S. Sharma, D. Sleeman, M. Rissakis, C. Moore, and S. Craw. The Machine Learning Toolbox Consultant. *International Journal on AI Tools*, 2,3:307–328, 1993.

[7] N. Graner and D. Sleeman. MUSKRAT: a Multistrategy Knowledge Refinement and Acquisition Toolbox. In R. Michalski and G. Teccuci, editors, *Proceeding of the 2nd International workshop on Multistrategy Learning*, pages 107–119, 1993.

[8] Y. Kodratoff, D. Sleeman, M. Uszynski, C. Cause, and S. Craw. Building a Machine Learning Toolbox. In B. Le Pape and L. Steels, editors, *Enhancing the Knowledge Engineering Process - Contributions from ESPRIT*, pages 81–108. Elsevier, 1992.

[9] T. Mitchell. Generalization as Search. *Artificial Intelligence*, 12(2):203–226, 1982.

[10] K. Morik, K. Causse, and R. Boswell. A Common Knowledge Representation Integrating Learning Tools. In R. Michalski and G. Teccuci, editors, *Proceedings of the 1st International Workshop on Multi-Strategy Learning (MSL 91), West Virginia (USA)*. Harpers Ferry, November 1991.

[11] Y. Niquil. Guiding Example Acquisition by Generating Scenarios. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Conference on Machine Learning (ML 92)*, pages 348–354. Morgan Kaufmann Publishers, Inc., July 1992.

[12] Y. Niquil. Specifications of Example Postprocessing and Evaluation Modules, and their integration in MLT-2. Deliverable 7.4, Machine Learning Toolbox ESPRIT Project P2154, June 1992.

[13] F. Provost. A baseline taxonomy of bias selection policies. In *Proceedings of the ML92 workshop on Biases in Inductive Learning*, 1992.

[14] M. Rissakis and D. H. Sleeman. Consultant-2: Adjustment of learning bias for real world tasks. Unpublished report, Computing Science Dept., Univ. of Aberdeen, Scotland, 1993.

[15] D. Sleeman. Towards a Technology and a science of Machine Learning. *AI Communications*, 7(1):29–38, 1994.

[16] D. Sleeman, R. Oehlmann, and R. Davidge. Specification of Consultant-0 and a Comparison of the Several Learning Algorithms. Deliverable 5.1, Machine Learning Toolbox ESPRIT Project P2154, December 1989.

[17] B. Tsatsarakis and D. Sleeman. Supporting Pre-Processing and Post-Processing for Machine Learning Applications: A Workbench for ID3. *Knowledge Acquisition, 5*, pages 367–383, 1993.

[18] P. Utgoff. Shift of bias for inductive concept learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning*, volume 2, pages 107–148. Morgan Kaufmann, Los Altos, CA, 1986.

[19] B. Wielinga and A. Breuker. Models of expertise. In *Proceedings of the 7th European Conference on AI (ECAI'86)*, pages 306–318, 1986.