1996

A neural-net approach to user expertise modelling

Siu Cheong Leung
University of Wollongong

A NEURAL-NET APPROACH TO USER EXPERTISE MODELLING

A thesis submitted in partial fulfilment of the requirements for

the award of the degree

MASTER OF SCIENCE (HONOURS)

from

UNIVERSITY OF WOLLONGONG

by

Siu Cheong LEUNG,

BSc.(HONS), CUHK*, Grad.Dip. Edu, CUHK*

* Chinese University of Hong Kong

Computer Science Department

1996

STATEMENT OF ORIGINALITY

I hereby declare that I am the sole author of this thesis and it has not been submitted to any other university.

I also declare that the material presented within is my own work, except where duly acknowledged, and that I am not aware of any similar work either prior to this thesis, or currently being pursued.

Siu Cheong LEUNG, June 1996.

ABSTRACT

A neural network approach to low-level user modelling is described, in the context of text editing tasks using the Jove editor. Knowledge of a user's expertise is extracted automatically, based on their interaction with Jove over a two-week period. A multi-layered perceptron (MLP) classifier which uses rprop, or resilient backpropagation" learning and incorporate input data fuzzification is developed to classify users into one of five expertise levels. Classification into correct level is achieved in around 80% of cases, with misclassification being restricted to adjacent classes. The neuro-fuzzy system is seen to outperform not only the binary classifier of Beale (1989), but also production rule and inductive expert systems developed especially for comparison purposes in this study.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Mr. John Fulcher, for his continuous guidance, encouragement and support in the completion of my thesis. I am especially impressed by the coordination work he has done for me. I would like to thank Professor Jennifer Seberry and Mr. Joshua Fan for their invaluable suggestions in the early stage of my thesis. Further, I would like to thank Mr. Michael Milway who developed the keystroke monitor hardware for the experiment, and Mr. David Wilson and Mr. Adam Barclay who set up the testing environment for my research. Finally I must thank those who participated in the actual typing experiment during the implementation of the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. INTRODUCTION

Interactive system development involves the design of a user interface. A good user interface can help users accomplish their intended tasks, enable them to understand the system better and also to enhance the user's capability [Tyler91]. Conventional applications assume a single model for all types of user. Lately, some researchers admit the fact that more than one user model exists. Accordingly, they let users select the level of expertise, then provide the appropriate interface. Contemporary user interface designs emphasize self-adaptability to different types of users so as to increase their productivity. Advanced techniques from the artificial intelligence field, in particular autonomous agents, are used as a personal assistant who is collaborating with the user in the working environment [Maes94]. The assistant becomes gradually more effective as it learns the user's interests, habits and preferences. This automatic adaptability is based on user modelling -- classification of users into different user models according to their interaction with the system. A user model should include information about a user's characteristics such as his/her cognitive abilities and weaknesses. Using this classification, the system plans the best way to interact with the user. The results of user tests on using adaptive interface and first generation autonomous agents are encouraging [Maes94].

User modelling is an emerging focus of attention in the human-computer interaction field [Coutaz92]. Past user modelling techniques are largely conventional based. An

artificial neural network (ANN) based classification technique for user modelling is studied in this research. This approach is different from traditional ones and it is expected that some of the difficulties that the former experienced can be overcome using it.

2. THE PROBLEM

2.1. MOTIVATION FOR THIS STUDY

An adaptive interface can adjust itself to the user's behaviour and characteristics. It can speed up task accomplishment for expert users and can help to lower error rates for the novice user. A successful intelligent interface must get access to many different sources of knowledge, namely the knowledge of the user, the user's task goal, the tools, the domain task itself and the interaction modalities [Rissland84]. It can be seen that a good understanding of the user and his/her task goal are vital. However, extraction of such knowledge from inside the user is not trivial.

Manual methods exist for this extraction -- for example, asking the user questions about their ability and task objectives when (s)he enters the system [Rich79]. These methods rely exclusively on user inputs that may be inaccurate or unreliable. On the other hand, some automatic ways are available, such as production rule systems [Bovair90], frame-based systems [Finin83] and inductive inferencing techniques [Durkin94]. Each of them has their merits and demerits -- common problems being either effectiveness or efficiency.

An ANN approach provides promise in solving these problems in an effective and efficient way.

2.2. THEORETICAL FRAMEWORK FOR THE PROPOSED STUDY

2.2.1. User modelling

Human Computer Interaction (HCI) designers use a cognitive psychology definition of the user model to aid their tasks, but this is not what we are interested in. Here, instead, we refer to the user model as the "Embedded user model", which is incorporated in a running system and used to increase automatic adaptability [Coutaz92]. This user model supports the end user's task, not the HCI designer's task. The component of the user model to be focused on in this study is the expertise level of the system user.

The basic assumption of user modelling is that a person has patterns of behaviour that will persist over a period of time, and that are repeated either consciously or unconsciously. In human-computer interaction, such behavioural patterns between the user and the system include the interaction preference, the timing, the sequence and the error rate. The pattern of interaction reflects the strength/weakness, goal/plan, preference/attitude and belief of the user about the world.

2.2.2. Artificial neural networks

An artificial neural network is composed of layers of non-linear processing elements joined together by variable interconnecting weights. The network is trained by presenting training epochs (a set of training examples) to it. Here the training exemplar comes from the abstraction of the usage trace of a selected system, into bit pattern form which the neural network is able to read. The interconnecting weights are adjusted after each epoch according to the learning algorithm. The training continues until the error between desired and actual outputs falls within a tolerance

limit or after a predetermined time. The knowledge of the user is learnt and stored by a network in the weighted synapses and threshold logic units [Beale90]. The trained network contains the contribution of different user interactions to the classification (here it is the level of expertise). Outputs can be in either Boolean form or fuzzy form, representing the category of user in terms of expertise level in using the system.

The ANN learns automatically -- it does not require explicit rules. Moreover, it can learn non-linear functions with many inputs and outputs. The ANN has strong knowledge acquisition capability. If the training set is representative, the trained network can generalise from specific examples to principles, allowing the ANN to recognise and classify unlearnt patterns. The parallel nature of the ANN makes it noise tolerant. It can work with incomplete or noisy inputs, and guarantees an output. The parallel nature also allows the ANN to respond fast to inputs. Training of the ANN may take a long time, but once trained, the network can produce the correct output in a very short response time.

2.2.3. Fuzzy logic

Fuzzy logic is a branch of logic that uses membership in sets rather than a strict true/false membership [Zadeh89]. It is primarily concerned with quantifying and reasoning about vague or fuzzy terms that appear in our natural language [Durkin94]. During the past decade, use of fuzzy logic has been applied to many task domains.

Accurate task and user modelling are critical for constructing intelligent human-computer interfaces that can adapt to individual users. However, there are inherent difficulties in dealing with inexactness that are always associated with individual

as production rules, there is still a need to select the main interactions to track in the extraction and abstraction of a user trace, as well as a need to decide the representation of interactions. An improper selection and abstraction of information can hide significant features from the neural network and cause misclassification.

2.5. DEFINITION OF TERMS

Usage Trace. Usage trace refers to the user interaction log on the selected system.

Abstraction of Usage Trace. The raw usage trace must be preprocessed before presentation to the input of the ANN. This is termed abstraction. The abstracted usage trace is a bit pattern.

difference and human problem solving strategies. Fuzzy logic provides a way to represent such ambiguity. [Norcio91] pointed out that a fuzzy logic based representation can provide a better description of the user expertise level. We can expect that such representation can better model the novice-to-expert shifts.

2.3. STATEMENT OF THE PROBLEM

The purpose of this study is to investigate the feasibility of knowledge acquisition in user modelling by means of a neural network approach.

The elements to be investigated in the study are:

- To determine the feasibility of the ANNs in classifying users into their levels of expertise.
- To determine the best performance network parameters in the chosen learning paradigm for such an application.
- To contrast the ANN approach against a production rule approach
- To contrast the ANN approach against an inductive system approach (which is an alternative automatic learning paradigm)

2.4. LIMITATIONS OF THE STUDY

Unlike other methodologies, an ANN approach stores the 'rules' and 'knowledge' in the interconnection weights of the links. If there is any criticism of this approach, it is that this is a "black box" approach to user modelling. No explicit rule is evident to explain the reasoning of the user model and so it is difficult to analyze ANNs.

Lastly, although the neural network approach makes fewer assumptions on the weighting of each parameter of the user model than other traditional systems, such

3. REVIEW OF THE LITERATURE

3.1. OVERVIEW OF THE THEORY AND RESEARCH LITERATURE

A study of user modelling crosses the human-computer interaction and artificial intelligence fields. There are a number of approaches for developing a user model. An early and direct technique is the classification into stereotypes [Rich79]. In this approach the user is classified into a category after responding to a series of questions. One of the problems with this approach is the inaccurate responses, either due to users' misinterpretation or reluctance to give information about themselves. Moreover, it also introduces an extra load onto the users. Other traditional approaches to user modelling involve an explicit, large *knowledge base* that stores details of possible user characteristics and a *rule-based system* that infers the user classification [Beale89]. Unfortunately, knowledge acquisition by rule-based systems is tedious and problematic. It requires intensive interaction between the knowledge engineer and the domain expert, either to elicit problem-solving procedures or to observe how the expert solves problems. Furthermore there are too many parameters in human-computer interaction, and it is difficult to define the rules. Because of the complexity of the problem, some parameters are even unknown to the designer. Thus it is possible to miss important parameters in defining rules.

Machine learning methods provide better ways of knowledge acquisition and updating. These methods attempt to automate the knowledge acquisition process. Numerous discussions of these methods are available in the artificial intelligence

literature. One of these methods is inductive learning in which the rules are extracted from a set of training examples [Durkin94]. A well-known inductive inference technique, ID3, constructs a decision tree of attributes from training examples. The weighting of each attribute (decision factor) indicates its contribution in classifying training examples. Generalisation is achieved by repeated application of certain operations to the initial descriptions, such as dropping non-relevant factors and climbing up the decision tree. However, with a large number of criteria, this approach would become impractical in terms of search time [Deng94].

Another method is memory-based reasoning [Stanfill86] in which the input is compared with worked cases stored in memory to find the best matched ones. Its mechanism differs from that of inductive learning, in that memory-based reasoning operates directly on the data, whereas inductive learning uses rules as an intermediate structure. Each stored case consists of some predictor features and a goal feature. Firstly, the predictor features of the input are compared with those of stored cases. A numerical "distance" and a numerical "weight" are computed for each predictor. Using these distances and weights, a "total distance" measure is computed. The total distance metric determines how closely the input matches the problems at hand. The records with the smallest total distance are then selected. This approach has problems when the input is not sufficiently similar to stored cases or is equally similar to several stored cases [Deng93].

The ANN approach proposed here is good at handling problems with incomplete information and noise. It is well suited to pattern recognition and classification problems. Many competing hypotheses can be tested simultaneously using massive parallel networks composed of nonlinear processing elements connected by links

with variable weights [Deng93]. ANNs use training examples as inputs so no explicit rule is required. A neural network's generalisation ability allows the system to recognise patterns not previously learnt. Due to its parallel nature, it is more noise tolerant. Once trained, the network responds fast so it is resource efficient.

3.1.1. Past work on user modelling

The study of user characteristics from keystrokes dates back to the keystroke level model of Card, Moran and Newell [1980]. This model takes into account the interkey pause and classifies the user interactions with the system into Keystroke, Mental and other attributes [Card80]. Such a model has been extensively studied in uniquely identifying individual users for the purpose of intruder detection [Brown93] [Newberry & Seberry89] [Newberry91] and the results are quite promising. For a given user and keyboard pair, the interkey time distribution is very consistent over time and over different English typed texts [Pisitkasem90].

The pattern of keyboard interaction can also serve as a tool to analyse a user's expertise level [Beale89] [Lane93] and task goal [Villegas94]. Expertise level and task goal are the two basic domains of a user model that can be used to develop an intelligent interface [Norcio91]. Knowing what the user is intending to do and how familiar he/she is in using the system, the system can predict the user's next action and provide appropriate help and an interface for the user to work with.

As a user model tends to be incomplete, uncertain, ambiguous, unstructured and unstable [Chiu91], the choice of feasible paradigm to tackle the classification of users is limited. An approach that can adapt to an environment of incomplete and noisy information and has superb learning ability is ideal for performing such a classification task. For the point of view of real time interactive system

development, a fast reactive approach is demanded. An ANN approach has a strong appeal to perform such a task.

3.1.2. Past work on user modelling by ANN approach

Modelling users with a neural architecture is a new research field. Chen and Norcio [Chen92] provide a more complete picture of the primitives of a neural network that are useful for user modelling:

(1) feed-forward network trained by the backpropagation algorithm is an appropriate pattern classifier paradigm;

(2) bi-directional associative memory (BAM) and Kohonen's LVQ are useful in recognising user-task context;

(3) adaptive resonance theory (ART) addresses the conflict in learning between adaptability and stability by shutting down learning when a large mismatch occurs between the input and the projected output.

The same authors also proposed in another paper a neural net approach to user modelling in the context of information retrieval [Chen91]. A prototype system, UM-net that generated descriptions of decision support system (DSS) tools, was presented. It modelled the user's domain experiences and inquiry interests and tailored the descriptions about the DSS software package provided to a user. UM-net was a modular design. The neural network module was a set of feed-forward networks trained by the backpropagation algorithm. It was further divided into three sub-modules (Figure 1), each performing a different task of pattern classification.

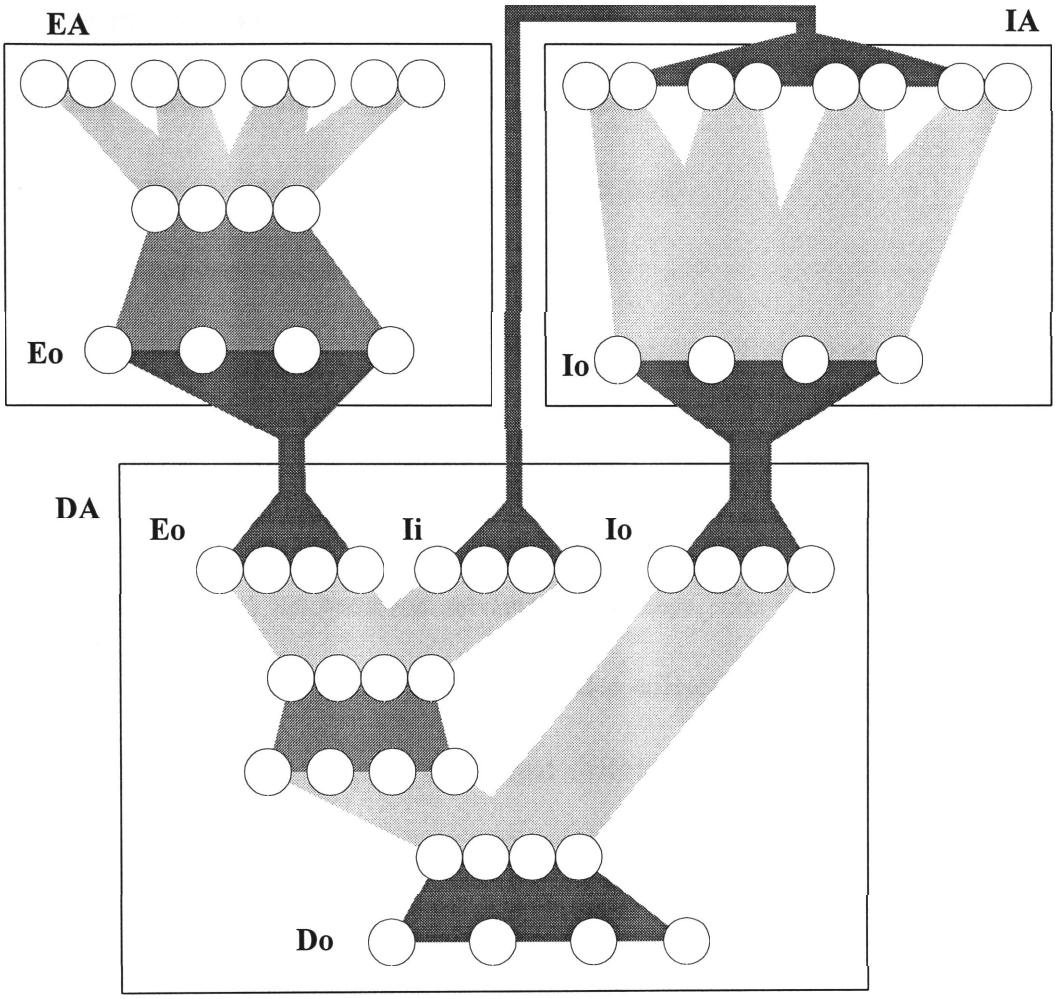


Figure 1. Schematic diagram of neural network module in UM-net

The Expertise Analysis (EA) module consisted of 22 input nodes that denoted the user's engagement in these categories: programming, database application and system operation, application of decision models, application of DSS software package and position description. There were 4 hidden nodes and 4 output nodes. The outputs indicated the user's level of expertise in four aspects: computer literacy, DSS application, management and capability of problem solving

The Interest Analysis (IA) module is a two-layered network, consisting of 20 inputs and 4 output units. The inputs of IA come from a user's inquiry, and contain 4 sets of input values: model-oriented inquiry, application-oriented inquiry, operation-oriented inquiry and commercial-oriented inquiry. The output indicated the user's implied intention, namely, to understand:

- technical features about system functionality
- application domains and relevant instances
- operational features about the software, and
- purchase information

The Description Analysis (DA) module is a five-layered network that included 12 inputs, 4 outputs and 3 hidden layers each comprising four processing elements. The 3 hidden layers are introduced to simulate the sequences of inference performed by a human intermediary in information advising activities. The inputs of DA come from three sources: outputs of both EA and IA, and a subset of inputs in IA, making it a cascading topology. The outputs described different aspects of the DSS software. This output is then used to look up suitable DSS software in the database.

Subdividing a neural network processing system into modules provides a more understandable view of the system. It also allows more flexible training -- the training process can be conducted for either a single module or the whole system. Another advantage is that each module can be utilized separately for different purposes. Note however that the user's domain knowledge in this system is not extracted from his/her interaction with the system.

[Villegas94] mentioned a multi-layered perceptron (MLP) neural network to identify *text-editing goals*. 16 graduate students using the text editor "vi" as their primary editor through their coursework and research were chosen as subjects. They were presented with 9 prepared memos with which to work. The keystroke operations and pauses between keystrokes were recorded and used as inputs to the MLP. The outputs were the six text-editing goals, namely:

(1) address memo: subject enters To:, From:, Date: and Subject:.

(2) review work: subject reviews "`vi`" operations made

(3) error correction: subject makes or corrects an error

(4) organise memo: extraneous lines are deleted; paragraph formed

(5) punctuate memo: subject adds comma, period and capitals

(6) enhance memo: subject adds words and phrases

The memo editing operators were grouped into 36 keystroke operators and 3 pause operators that denoted different lengths of pauses (long pause of over 10 seconds, medium pause of 5-10 seconds and short pause of less than 5 seconds). In order to capture the context of the current operator, a moving window of size 5 was used. Since the 39 operators could occur in any one of the 5 window slots, the net consisted of 195 input nodes. The hidden layer had 5 nodes, resulting in a 195-5-6 (input-hidden-output) MLP architecture, trained using the backpropagation learning algorithm. The overall recognition rate achieved was 96%.

An ANN approach to user classification according to *expertise level* was proposed by [Beale89] which uses *advanced distributed associated memory* (ADAM) [Chen94]. Beale modelled expertise characteristics ("expert" versus "novice" users) of a functional programming environment called *Glide* at the University of York. "Glide" was command-based, having a small command set of 24. Glide usage was unobstructively and automatically logged over a period of 3 months. The usage trace was captured and abstracted (i.e. transformed into a bit pattern) before passing to the ADAM neural network for classification.

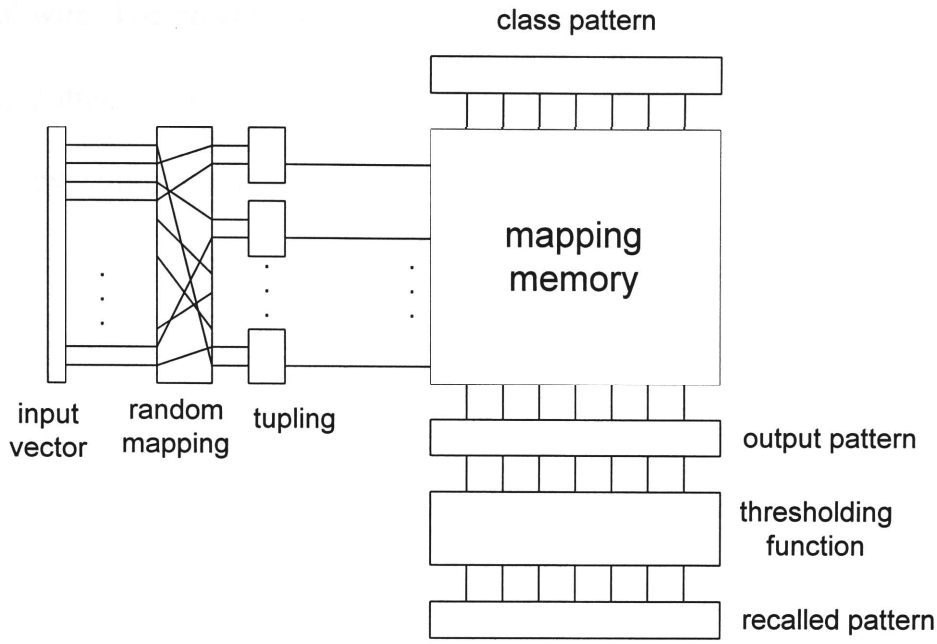


Figure 2. Schematic diagram of ADAM

The architecture of ADAM is shown in Figure 2. The usage trace was first abstracted to provide a trace that reflected the pertinent information within the interaction. Each element of the abstracted trace could be represented by an individual bit pattern, and so a whole trace could be represented by a sequence of these bit patterns.

The bit pattern was first passed through a constant random mapping function, and then into a tupling function. The tupling process involved selecting bits from the presented pattern and combining them through a binary logic function to produce an output that was essentially a sparse encoding of the state of the input bits sampled. The tupling function was used to provide a non-linear element in the processing of the pattern which enabled patterns that were not linearly separable to be successfully classified, and so accounted for much of the generalisation properties of the system.

In the *training* phase, each input example was presented along with a unique class bit pattern that contained *n* selected bits set to one. The class pattern appeared on the vertical wires, whilst the tupled input appeared on the horizontal wires. The memory matrix thus contained links where each active vertical wire crossed an active

horizontal wire. The process was repeated for the whole example set. In *recall* phase, the input pattern was presented as before, but this time the class pattern was calculated by summing the number of links in each column that are activated by the tupled input. These totals are then *n-point thresholded* to recover exactly the number of bits set in the original class pattern, *n*.

The experiment randomly chooses 8 training exemplars each from the expert and novice groups and classifies the remaining data set of 176 traces (no detail was provided as to how the pattern was extracted and abstracted, however). The average classification success rate reported was 71.2%. Beale's study relied on data that was collected over a 3 month period, but he did not take into account that the novice user may shift to "more advanced" over time. Nor did he detail how the usage trace was abstracted. Moreover, Beale's coarse classification of a user into 2 levels -- expert or novice -- was relatively easy to identify, but it might not be sufficient if we expect a finer granularity.

3.2. SUMMARY OF "WHAT IS KNOWN AND UNKNOWN" ABOUT THE RESEARCH TOPIC

There had been little research covering the application of neural networks to user modelling. Beales' and Villegas' results showed that ANNs are capable of generating a sufficiently powerful knowledge base by generalising from an example set. However, there had been little subsequent research to validate or improve upon their findings. The applicability of other ANN architectures is still open for exploration. In terms of interaction types, there has been little study on mixtures of keyboard and

mouse. Other aspects of user modelling, such as user preference and attitude, remain unexplored.

3.3. THE CONTRIBUTION THIS STUDY WILL MAKE TO THE LITERATURE

In this study, we propose a different neural network architecture from Beale's ADAM, namely the multi-layered perceptron (MLP), to classify users into categories according to expertise levels. Secondly, we evaluate which learning algorithm, from among standard backpropagation (SBP) and some of its variants, yields best performance and resulting best parameter set. Thirdly, we derive the optimum granularity. Fourthly, we want to demonstrate that the classification is possible using few but substantial usage traces, as compared to the 3-month logging used in the Glide system. Fifthly, we explore the appropriate features for classification according to expertise level. Finally, we demonstrate the improvement in performance over the production rule and inductive system approaches by adopting the neural network approach.

4. RESEARCH DESIGN

4.1. CHOICE OF THE TRACKED SYSTEM

The system chosen is the `Jove` editor. It is based on the original EMACS editor and user manual written at Massachusetts Institute of Technology by Richard Stallman [Payne]. It is a popular full-screen editor that runs under UNIX and is widely used at the University of Wollongong. The users use the editor either for programming or for preparing email messages. As it is so popular, it is not difficult to locate subjects over a wide range of expertise. Also, under the UNIX environment it is possible to start a background process to capture the keystrokes of a session to a file.

`Jove` provides a command-line interface, and has a command set of approximately 200 (Appendix A). The command line is accessible by typing "Esc X". `Jove` also provides abundant hot-key support (Appendix B). The hotkey format is classified into three families: Ctrl-[x] sequences, Esc-[x] sequences and Ctrl-X-[x] sequences. `Jove` does not support a mouse or a pointing device.

4.2. INSTRUCTIONS GIVEN TO THE SUBJECTS

In order to get a faithful reflection of the subjects' daily style of working with the `Jove` editor, the subjects are told that the objective of the research is to study the different style of users in completing an editing task. They should do what they are used to doing, and not deviate from their usual style or speed. They are not told the importance of inter-key timing so as to avoid conscious or unconscious bias. The

subjects come from students and staff of the University of Wollongong and are all voluntary. As this experiment involves human subjects, permission to proceed was sought and granted from the University's Human Research Ethics Committee.

4.3. APPROACH TO USAGE TRACE RECORDING

There are two alternative approaches to recording usage traces; one being software and the other hardware.

Software Approach

There are two alternatives in the software approach. One way is to rebuild `Jove` with a user trace module embedded in it. Such approach requires access to the source code of `Jove`, especially the input/output routine. This was not deemed feasible for this research project. Another way is to have a separate program intercepting all the activities of the `Jove` session, and to change the subjects' shells. A program is written to create another session for the subject to run `Jove` and logs all input/output of the `Jove` session in a file. Such a method has been used in the study of a keystroke-level model of Lotus 1-2-3™ running on a stand alone personal computer [Lane93]. The advantage of the software approach is the transparency of the data capture to the user. Moreover, for the study of `Jove` running under the UNIX network environment, there are no site- and time-restrictions on data capture. A disadvantage of this method in a network environment is that the inter-key timing is dependent on the interrupt of the host computer and results in varying response time to the user. It is thus less accurate.

Hardware Approach

Noting the inaccuracies of the software approach, the hardware approach is adopted in this study. The requirement is a device (a dedicated microcomputer) attached to the terminal that the user works with. It captures all keystrokes of the terminal transparently and sends these to the host. The subjects need to work on a dedicated terminal at a specific time to take the usage trace. The advantage of this approach is that the device can provide accurate timing independent of the interrupt mechanism of the host computer [Pisitkasem90].

4.4. THE EQUIPMENT AND SETTINGS

4.4.1. The terminal

The terminal used is a Domino data display terminal model DVT227. It is set to emulate a VT100 terminal with the following settings:

Speed=9600 baud, 8 bits, no parity, 1 stop bit, no local echo, **XOFF at 128, Limited Transmit**

The layout of the keyboard is shown in Figure 3. Most subjects are familiar with PC keyboards rather than the Domino keyboards. So the major differences between a standard PC 101-key keyboard and the Domino DVT227 keyboard need to be highlighted here. They have an impact on the accuracy of the experiment. On the DVT227 keyboard,

- The Esc key is situated at F11 (some of the hotkey combinations use Esc key)
- The ">" and "<" keys are at the bottom-left corner ([GotoTOF] and [GotoBOF] use this key)
- The CapLock key is on the right of the Ctrl key (it is easy to inadvertently press CapLock while trying to press the Ctrl key)

- The keys Ctrl-S and Ctrl-Q are used for flow control and are not available within the `Jove` editor; Ctrl-\ substitutes for Ctrl-S, however Ctrl-Q has no substitute. (Ctrl-S is used in [Search] and [FileSave]; Ctrl-Q is used in [Quote])

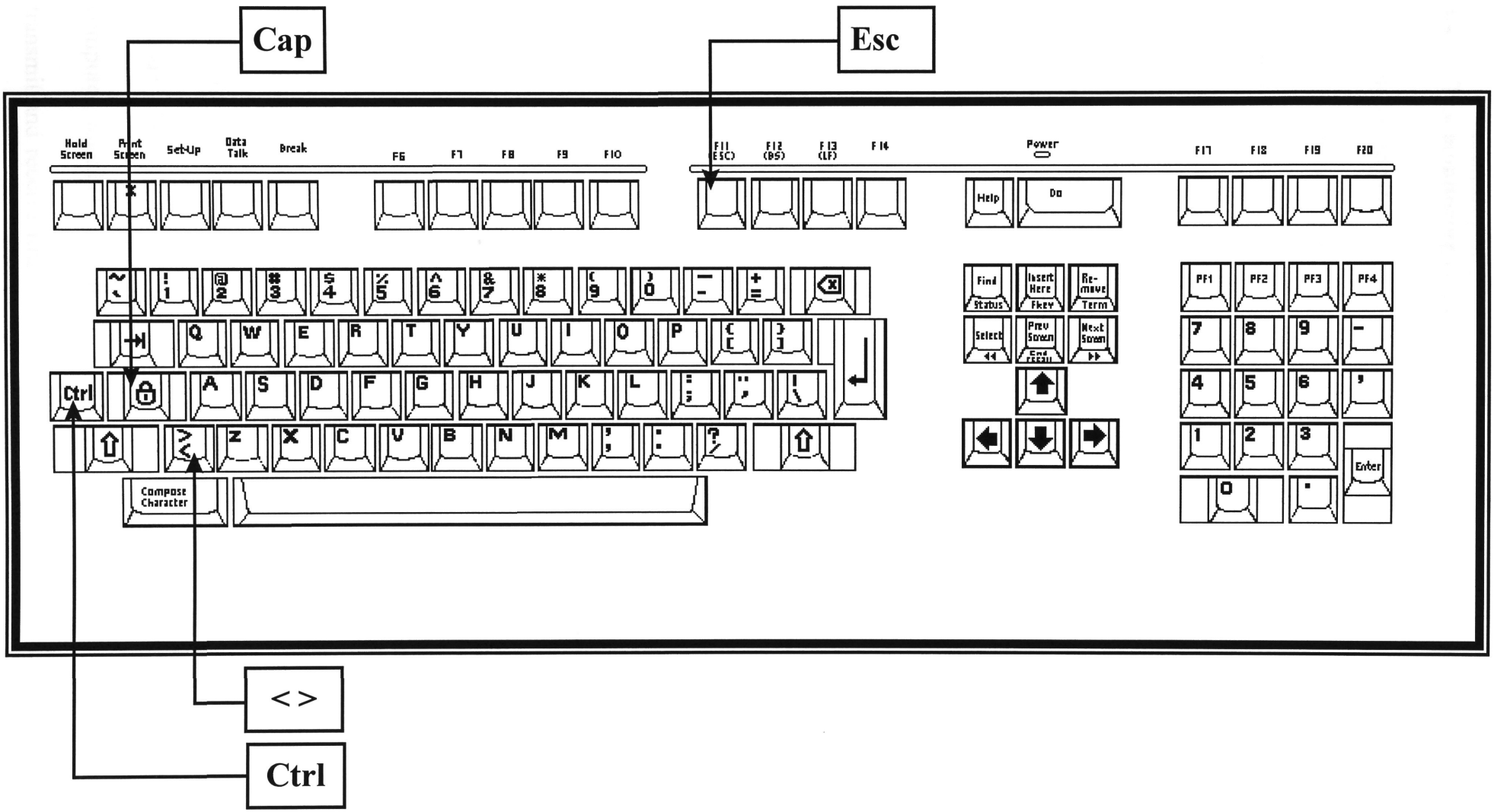


Figure 3. Keyboard layout for the Domino terminal

4.4.2. The monitoring equipment

The microcomputer-based monitor was constructed by Michael Milway at the Computer Science Department, University of Wollongong. It links directly with the transmit line of the terminal and provides accurate interkey timing *independent of* the interrupt services of the host computer.

The monitor is built from the Mini Board single-board computer [Martin94]. The hardware consists of a MC6811 based microcomputer with 256 bytes RAM and 2Kbytes EEPROM, one RS232 serial port and one programmable timer, running at 2MHz (Appendix C). The time interval is measured by the programmable timer that interrupts every millisecond. The input and output port speed of the monitor are set to be identical to the terminal's output port, i.e. 9600 baud.

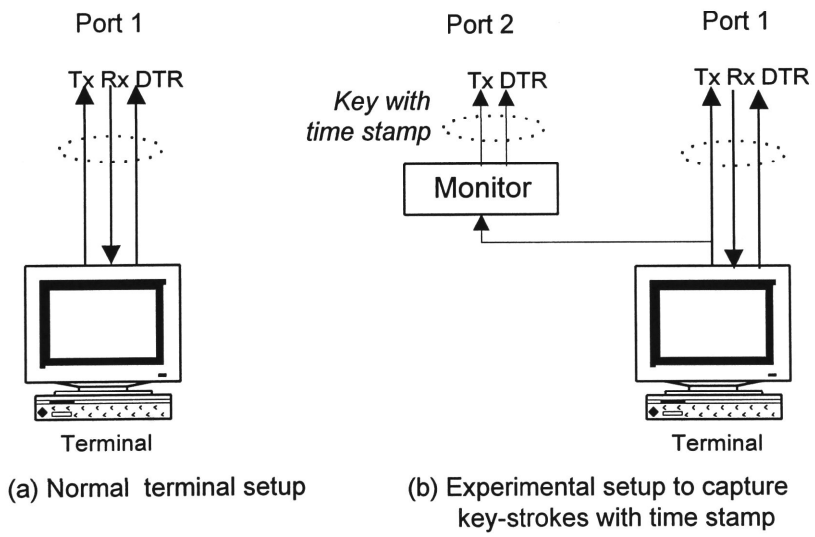


Figure 4. Configuration of monitoring device

A C program `keysnoop.c` (Appendix D) was written by Michael Milway and burnt into the EEPROM. The program uses 8 bytes of global variables and 611 bytes of code. The program receives data from the terminal port and sends this to the host computer, together with timing information and a synchronisation character. The transmit and receive lines of the terminal are not affected by this setup. The data

format is 1 byte of data, 2 bytes of 8-bit timing information plus a linefeed, making up to 4 bytes per record:

1	2	3	4
Character	Timing (binary)		[linefeed]

Another program `startlog.c` (Appendix E) is written by the author to poll the snoop port and copy the bytes coming in to a raw usage trace file. The timing information is converted to hexadecimal form before saving. The output data format is 1 character byte, a colon, 4 bytes of hexadecimal timing information and a linefeed, making up to 7 bytes per record:

1	2	3	4	5	6	7
Character	[colon]	Timing (hexadecimal)				[linefeed]

Subjects use the shell script `je2` (Appendix F) to start an editing session. `je2` was written to perform several tasks in addition to starting the standard `Jove` editing session. It starts a copy of the `startlog` program to poll the port before `Jove` is run. When `Jove` is exited, `je2` waits for a signal from the `startlog` program, indicating the complete closure of the log file. Then `je2` copies the log file to the appropriate storing directory according to the task performed.

4.4.3. Avoiding information loss due to un-synchronisation

The input data of the microcomputer was originally of the same format as the output data, i.e. 7 bytes per record. However, it was discovered in the pilot study that such a high rate of data transmission causes information loss. In the end, the data capture

programs, the Domino terminal settings and Jove settings were all modified to alleviate this problem

At the terminal, under Communication Set-Up [Domino]:

- Limited Transmit is set to *ON* in order to restrict the terminal transmit speed to be at most 180 characters per second regardless of baud rate, minimising the interrupt burden on the system. There is 1/180 = 5.55 milliseconds between successive character transmissions. Since the output port speed of the monitor is 9600 baud, corresponding to 1.04 milliseconds per character, we can pack 4 more characters in the output stream. This is sufficient for a 4-byte block but not the 7-byte block of the original design.
- Flow control is set to *XOFF at 128* to avoid display fall off synchronisation (the default buffer is 256 bytes long)

A new driver for `Jove`, *vt100_slow*, was created for this Domino terminal by adding the following line to the standard vt100 driver:

```
# standard vt100 stuff ...
#
# Line for Domino terminal below
set allow-^S-and-^Q off
```

This line informs `Jove` to ignore Ctrl-Q and Ctrl-S generated by the Domino terminal (they are used as XON|XOFF flow control).

4.5. USAGE TRACE ABSTRACTION

The above raw data is further processed to generate interactor objects to be stored with context information (edit/command/help mode), goal (the key/command), the

type of interaction (keystroke/command-line), the manipulated data and the time-stamp. This trace will be helpful in producing the abstraction (preprocessing) of inputs to the ANN. Such a framework would also be useful if we were to extend the usage trace to a graphical interface with mouse-click [Belage92], though this is beyond the scope of the present study.

This processing involves extracting meaningful keystroke sequences, taking usage statistics of certain commands/keystrokes, and the pause timing between important operators. The result is then encoded in bit pattern form. The raw trace is processed firstly in UNIX to produce a file containing records of interactor objects. This file is then further processed on a PC using several programs written in Microsoft Foxpro version 2.6. The FoxPro programming language belongs to the family of Xbase dialects, and is very close to that of dBASE IV. The reason for choosing FoxPro is because of the flexibility provided by the database system in filtering, searching and modifying the database structure. Moreover, the software is easy to access and the author is familiar with the Xbase dialects.

The abstraction procedure was continuously refined from the pilot study to the full scale study in order to obtain optimum results.

4.6. DESIGN OF THE ANN

From an ANN perspective, user modelling is regarded as a type of pattern recognition and classification problem. The input to the ANN is a representation of user characteristics obtained from the user-machine interaction. It is an abstraction of the usage trace of the selected system. The output of the ANN is the classification, that is, the appropriate expertise level in using the `Jove` editor.

Network simulator used

The Stuttgart Neural Network Simulator (SNNS) was used throughout this study. Version 3.2 of the software is installed in the Computer Science Department of the University of Wollongong, running under SUN OS Solaris 2.5, and Version 4.1 is installed in the author's 486-based PC running Linux Version 1.2.3. SNNS is a software simulator for neural networks on UNIX workstations developed at the Institute for Parallel and Distributed High Performance Systems [Zell95] at the University of Stuttgart in Germany. Information about this software is available at *http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html* and the source code is available from *ftp://ftp.informatik.uni-stuttgart.de*. The simulator provides a flexible environment to create and test various types of neural network architectures, and provides a lot of different learning and initialising functions to work with.

4.6.1. Multi-layered perceptron (MLP)

The Multi-layered Perceptron is a neural network architecture based on supervised learning. It consists of an input layer, an output layer and at least one hidden layer (Figure 5). Each input node is connected to each of the hidden nodes and each hidden node is connected to each of the output nodes. Nodes within the same layer are not connected. The learning rule of the MLP is called standard backpropagation. The weights of the network are initialised to small randomised values. Training patterns are then presented to the network one by one. The error is given by the difference between the desired output and the actual output. The error back-propagates from the output layer towards the input layer, and is used to adjust the weights of the network.

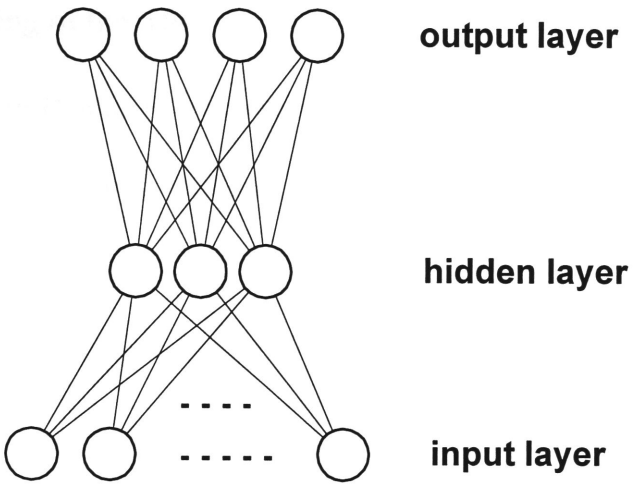


Figure 5. Multi-layered perceptron

Issues of MLP Design: There are a lot of design issues concerning MLPs, such as the number of layers, the number of hidden elements (nodes), the size of network parameters and the choice of weights. There is no definitive answer to these questions but rather they are determined by trial-and-error [Ferret93]. It should be pointed out, however, that a 3-layered MLP is enough to separate any number of classes [Beale90].

4.6.2. Kohonen's self-organised map (SOM)

Kohonen's Self-Organised Map algorithm is based on unsupervised learning. It constructs topology-preserving mappings of training data where the location of a unit carries semantic information. SOM consists of two layers of units: a one-dimensional input layer and a two-dimensional competitive output layer, organised as a 2D grid of units, as indicated in Figure 6. Each unit in the layer holds a weight vector, that resembles another different pattern after training. The learning algorithm for SOM accomplishes two things:

(1) clustering of the data

(2) spatial ordering of the map so that similar input patterns tend to produce a response in units that are close to each other in the grid.

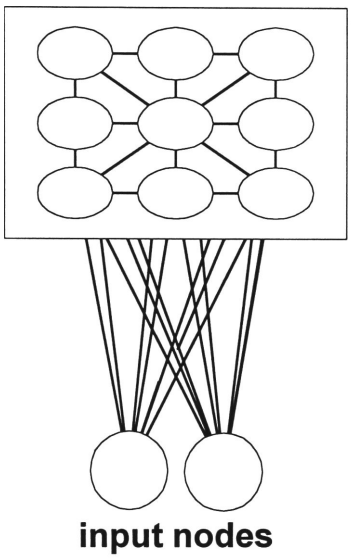


Figure 6. Kohonen's self-organised map

4.7. PATTERN RECOGNITION IN USER MODELLING

User modelling can be viewed as a pattern recognition and classification problem. The user's interaction with the system is the pattern and the "model" (expertise level) is the classification. The classifiers are knowledge (and frequency of use) of specific operators, the sequence of operators, efficiency in using these operators, speed and errors. The following concepts of image and speech recognition are also applicable to pattern recognition in user modelling.

(1) *Feature Nomination*

List the features to be extracted and how they are represented in the usage trace or intermediate output during preprocessing.

(2) *Pre-processing*

The purpose is to extract intended features -- like operators (hotkeys), pattern sequences, errors, timing – and hiding of inappropriate information.

(3) *Normalisation*

The purpose is to reduce the dimensionality of the input space, to reduce the complexity of the system or to remove data dependencies. For example, we take the overall hotkey timing as a RATIO normalised to the users' normal key timing. This removes the dependency of the figure on the user's typing speed.

(4) *Thresholding*

The purpose is to suppress background noise level or divide a continuous input into sub-ranges represented by discrete levels. For example, overall hotkey usage is represented by five nodes in the neural network, so four thresholds are used to divide the continuum of hotkey usage into five discrete levels.

(5) *Granularity Control*

Granularity control applies to both *Input Abstraction* and *Classification.*

Input Abstraction

If we have coarse grain abstraction (that is, we track the high level summary of various features), we may have insufficient information and only achieve low accuracy. If we have a fine grain (that is, we track operators down to the single keystrokes level) there will too much information, making the system complex and resulting in long training times.

Classification

If the grain of classification is not optimal, misclassification may often occur. If the classification is too fine, the classifiers have to be able to track small differences in pattern, making the system complex.

4.8. USER TASKS

All tasks are performed on a dedicated Domino text terminal, so subjects are forced to use a keyboard instead of a mouse to perform tasks. The subjects are given the same four tasks on two separate days about a week apart, so each subject is expected to generate two sets of data. Such an arrangement makes more data sets available and avoids possible error in performance due to memory effect. Some subjects could not arrange time for the second experiment, so their traces are used for testing only.

The experiment involves working with four memos, each of which has a different task goal. The order of the tasks is randomised every time to avoid error due to inheritance of order. The subjects are briefed on the difference between the experiment terminal and keyboard with the one they are used to. If they are not familiar with the keyboard they are given a memo of about 100 words for training.

The subjects are given written and verbal instructions to start. During the experiment, subjects can access the investigator for help but the investigator was not to interrupt while the subject was typing a memo.

Four memos were designed for four different tasks (Appendix G):

(1) creating a memo

(2) addressing a memo

(3) formatting a memo

(4) Enhancing and spell-checking a memo

Each of these tasks emphasises a different combination of keystroke and command set. The separate tasks serve as a way of masking the keystrokes for different task goals.

Creating a memo. Subjects should use more normal text keys. The memo is designed so they can use [Copy] and [Paste] hotkeys/commands to save some keystrokes in repeated typing. It is important to note that creation of a memo usually gives a longer inter-key pause in real life, when a subject is thinking while typing.

Addressing a memo. Subjects need to type only a few lines of To:, From:, Date: and Subject: and navigate to the next memo. They can speed up navigation by using [NextPage]/[PrevPage] hotkeys.

Formatting a memo. More editing keys (like [Tab], [Delete], [Linefeed]) are used.

Enhancing and spell-checking a memo. A wide variety of activities are involved, including error correction, text search and replace, text block move and delete. So it is possible to have a wide mix of features, including [Search], [Replace], [Cut], [Copy], [Paste], [ChangeCase], [ExchangeChar], [GotoEndOfLine], [GotoTOF], [GotoBOF] ... and so on.

4.9. PHASES OF STUDY

The study was conducted in 2 phases: the Pilot study and the Full-scale study. The pilot study was implemented in October and November 1995. The full-scale study was implemented in March and April 1996.

In the pilot study, only a limited number of subjects were involved. The objective of the Pilot Study was to evaluate the *feasibility* of a connectionist approach in classifying subjects into two categories -- expert and novice -- according to their level of expertise in using the `Jove` editor under UNIX. The by-product of the pilot study was a refined procedure of data capture and preprocessing which was subsequently used in the full scale study.

In the Full-scale study, more subjects were involved. The results are more precisely treated and analysed. The number of levels of classification is increased from 2 to 3, and then finally to 5.

5. PILOT STUDY

Eight subjects were selected. The major task was to classify them according to their level of expertise in using the `Jove` editor. They were to be classified into two groups -- "expert" and "novice" -- according to their level of use of `Jove`. The classification was performed through:

(1) Comparing usage trace file size and number of keystrokes used,

(2) Comparing the number of advanced features used,

(3) Comparing the efficiency in using advanced features, and

(4) Comparing the recorded/observed errors made in using advanced features

5.1. SELECTION OF SUBJECTS

Eight voluntary subjects were selected. The choice of study discipline at the University of Wollongong for both groups was mixed, with three from the Faculty of Commerce and five from the Faculty of Informatics. Some of them use `Jove` solely as an email editor; others also use it in the writing of programs. Even for the subjects in Informatics, the frequency of using `Jove` varies. One of them is a frequent user of the `vi` editor but a novice with `Jove`. Typing speed varied within the population.

There was one hunt-and-peck[1] (two-finger) typist, and one was capable of secretarial typing speeds. The rest were normal speed typists with varying rates of typing error. There were 31 training patterns and 40 testing patterns.

5.2. PREPROCESSING

The purpose of preprocessing is to (1) extract important features from the log, (2) to reduce the dimension of the input, and (3) to convert data into a readable format.

5.2.1. Preprocessing stage 0 - manual fix of un-synchronisation

Due to the speed limitation of the hardware (in the early stages of this project), information loss occurred. When the subject presses cursor keys continuously some keys could not be logged, which causes loss of synchronisation at some points in the log file. The reason for this is that a single cursor key at the terminal side generates *three* equivalent keystrokes (e.g. up-arrow produces Esc-[-A) to the terminal output, and the hardware cannot cope with such a high speed. Until this problem was solved, the user traces capture program was modified to mark these points. The log file was then manually inspected, and several macros were developed in Word for Windows to delete these and put the remainder of the trace in proper position. (The resulting error should not impact significantly on the accuracy of the result, as it happens only in a long continuous stream of cursor keys.) The raw user trace looks like this:

```
Key:Pause(hexadecimal)
--- -----
^[:0234
```

[1] A hunt-and-peck typist is an unskilled typist who generally uses at most four fingers, locating the position of each key individually each time it is pressed. There is no binding between fingers and keys, the typist using whatever finger happens to be the most convenient.

```
O:0006
B:0007
D:0067
a:0142
^M:FFFF
```

5.2.2. Preprocessing stage 1 -- converting raw data to interaction records

- Firstly, some important information is added to the end of the record. It includes the interaction mode (EDT/CMD/HLP/ESC/CTLX), the operator and the descriptor. A program that emulates the interpreter of the `Jove` editor to user input was written to do the job. The resulting trace is helpful in producing the abstraction of input to the ANN.
- Secondly, hexadecimal inter-key pause is converted to decimal format.
- Thirdly, some keys recorded in the log file present problems for further processing. These keys include tab(^I), linefeed(^M), and newline-and-tab(^J). The characters for these keys are usually regarded as delimiters between fields of a record, and cause a mistaken end-of-field or end-of-record. They are converted to other forms that do not interfere with the record handling.

Operators

Operators consist of the class of the operation and the subclass. For example, [Delete] is classified as a normal editing key for deletion, denoted as ***edit-del***; [Ctrl-K] (Delete-to-end-of-line) is classified as an advanced editing key for deletion, denoted as ***edita-del***.

Key	Keys	Pause	Mode	Operator	Descriptor
^[	^[OB	15	EDT	*navig*	cursor_dn
O	^[OB	7	EDT	*navig*	cursor_dn
B	^[OB	6	EDT	*navig*	cursor_dn
M	^M	1156	EDT	*edit-lf*	linefeed
D	D	1582	EDT	text	---
a	a	210	EDT	text	---
t	t	146	EDT	text	---
e	e	245	EDT	text	---

5.2.3. Preprocessing stage 2 -- stripping off unnecessary records

Raw data from the user trace is first preprocessed to strip off unnecessary records. Typically, a single cursor key generates the keystrokes from the terminal (e.g. up-arrow gives Esc-O-B) is converted to a single representation with the inter-key pause of O-B neglected. After Preprocessing 2, the trace file should look like this:

Key	Keys	Pause	Mode	Operator	Descriptor
B	^[OB	7	EDT	*navig*	cursor_dn
B	^[OB	7	EDT	*navig*	cursor_dn
M	^M	1156	EDT	*edit-lf*	linefeed
D	D	1582	EDT	text	---
a	a	210	EDT	text	---
t	t	146	EDT	text	---
e	e	245	EDT	text	---
:	:	897	EDT	*colon*	colon
		792	EDT	*space*	space
1	1	299	EDT	text	---
3	3	263	EDT	text	---
A	A	1596	EDT	text	---
^H	^H	446	EDT	*edit-del*	backsp

5.2.4. Preprocessing stage 3 -- mark cognitive pauses and hotkeys

- Interleaved in the keystrokes are different levels of pauses. A long pause usually indicates a mental operation rather than slowness in typing. These pauses should be excluded in the averaging of the inter-key timing for typing. The long pause is a hidden operator of the trace, which usually indicates a boundary between two tasks. It should be useful in identifying task goals [Villegas94], but in the study we simply mark those records with long (L) medium (M) and short (S) pauses. (Thresholds = 10sec, 5sec and 2sec, respectively)
- In `Jove` the hot-keys begin with Esc or Ctrl-X. Their operation and timing should be distinguished from normal text typing. We mark records preceded by these 2 keys for later use. (M denotes meta-key, i.e. Esc; C denotes Ctrl-X).

Key	Keys	Pause	Mode	Operator	Descriptor	Pause	Prev-Key
^X	^X	87	CTRLX	---	>>>		
S	S	123	EDT	*file-save*	save file		C
B	^[OB	7	EDT	*navig*	cursor_dn		
M	^M	1156	EDT	*edit-lf*	linefeed	L	
D	D	1582	EDT	text	---	L	
a	a	210	EDT	text	---		
t	t	146	EDT	text	---		
e	e	245	EDT	text	---		
:	:	897	EDT	*colon*	colon	M	
^[	^[	87	ESC	---	>>>		
G	G	233	EDT	*naviga*	goto-line		M
		792	EDT	*space*	space		
1	1	299	EDT	text	---		

5.2.5. Preprocessing stage 4 -- generate statistics

Preprocessing Stage 4 is a statistical gathering process. The output directly determines the input to the neural network.

- Firstly, the user trace is evaluated and statistics taken of counts and average timings of normal key-key, and hot-key operations.
- Secondly, the count of edit keys (normal and advanced), navigation keys (normal and advanced), and other functional hot-key classes (block marks, search and replace) are taken, along with the average and standard deviation.
- Thirdly, the size of continuous editing blocks and continuous navigation blocks are recorded.

5.2.6. Preprocessing stage 5 -- abstraction and dimension reduction

Preprocessing Stage 5 is the abstraction of the user trace into a neural network input pattern (binary strings). What we achieve here is to identify the key features of a user trace to be presented to the network, and to reduce the dimension of the inputs. The technique used in dimension reduction involves using ratios and normalisation, which improves the independence of variables. We firstly identify the features that are selected to present to the neural network, which include:

Task performed

Task 1	Task 2	Task 3	Task 4

Proportion of Advanced Editing Usage (3 bits representing 3 levels)

Ratio of (Number advanced editing keys) to (Number of normal and advanced editing keys)

0	0 - 0.05	0.05 - 1.0

Proportion of Advanced Navigation Usage (3 bits representing 3 levels)

Ratio of (number advanced navigation keys) to (number of navigation keys)

0.02	0 - 0.05	0.05 - 1.0

Common Advanced Feature Used (3 bits representing blocking, searching and replacing)

block cut and paste	text search	text replace

Good Planning of Editing

Ratio of (average editing block size)/(average navigation block size). A high ratio usually means a user works more efficiently – either the user is familiar with the editor or plans his/her work better.

0 - 0.15	0.15 - 1.5	1.5 - 5	5 - 15	> 15

Usage of Hot-key (5 bits representing 5 levels)

Ratio of (Number of hotkeys used)/(Total number of keys)

0 - 0.001	0.001 - 0.01	0.01 - 0.1	0.1 - 0.5	> 0.5

Latency in using Hot-key (5 bits representing 5 levels)

Ratio of (the mean hot-key pause)/(mean normal key pause)

0 - 1	1 - 2	2 - 4	4 - 7	> 7

Percentage Error in using Hot-keys (3 bits representing 3 levels)

Ratio of (Errors in using hot-keys)/(Number of hotkeys used)

0 - 0.1	0.1 - 0.25	> 0.25

The result is then encoded in bit pattern form, for example:

Task	Adv. Edit	Adv. Navig.	Adv. feature	EditSize /NavigSize	Hot-key Usage	Hot-key Timing	Hot-key Error
1 0 0 0	1 0 0	1 0 0	0 0 0	0 0 0 1 0	0 1 0 0 0	1 0 0 0 0	0 0 1

|<---------- 31 bits ---------->|

5.3. DATA ABSTRACTION IN THE PILOT STUDY

It should be noted that the 31 inputs to the neural network represent a very *high level abstraction* of the usage trace. Individual features are ignored but the figure of generalisation is taken, for example, instead of tracking an individual editing hotkey, only the overall proportion of advanced editing key usage is summarised, properly thresholded and represented by 3 inputs. By ignoring the detail of the remainder of the trace, the size of the inputs falls to a manageable dimension.

The representation of inputs in terms of ratio and proportion amounts to a means of *dimension reduction.* It also helps to *eliminate dependent variables* due to individual differences, such as the typing speed of the various subjects.

To represent inputs in terms of levels, it is critical to produce *appropriate thresholds* with strong differentiation power. This is done by careful examination of the results in Preprocessing Stage 4.

5.4. NETWORK ARCHITECTURE

A 31-3-1 MLP was constructed. The single output unit denotes the expertise level: 1 for expert and 0 for novice. The size of the hidden layer (here, three) is minimum yet sufficient for this problem. The update function is "Topological Order," while the initialisation function is "Randomised Weights." In order to distinguish this MLP

with the ongoing refinement of the network, we code this MLP-0, the architecture of which is shown in Figure 7.

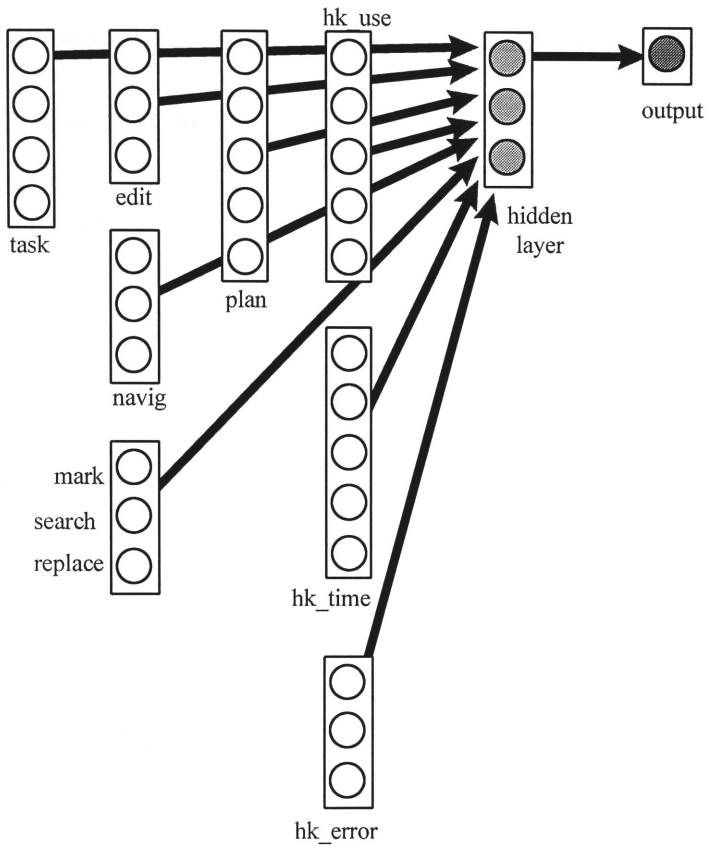


Figure 7. Architecture of MLP-0

Columns 1-4 are input layers
Task: one of the 4 tasks; Edit: the editing operators; Navig: the navigation operators;
Mark: text mark operators; Search: text search operators; Replace: text replace operators;
Plan: planning ability; hk_use: hotkey usage; hk_time: hotkey interkey timing; hk_error: hotkey usage errors
Column 5 is hidden layer
Column 6 is output layer: 1=expert, 0=novice

5.5. LEARNING ALGORITHM

The network is trained using one of three 3 different learning algorithms. The meaning of their respective learning parameters is as follows:

Standard backpropagation (SBP)

η: learning factor, $0 < \eta$; $d_{max:}$ maximum tolerated difference, 0 - 0.2

Backprop+momentum (BP+M)

η: learning factor, $0 < \eta$; μ: momentum factor, $0 < \mu < 1$

c: flat top elimination value, $0 < c < 0.25$; $d_{max:}$ as above

Quickprop (QP)

η: learning factor, $0.1 < \eta < 0.3$; μ: maximum growth factor, $1.75 < \mu < 2.25$

ν: weight decay factor, $0 < \nu < 0.001$; $d_{max:}$ as above

5.6. RESULTS OF THE PILOT STUDY

5.6.1. Convergence rate

The optimum parameters of the MLP network for the different learning algorithms are shown in Table 1. The acceptable error E is chosen to be 0.1, 0.01 and 0.001.

Algorithm	*Learning Parameters*			*Iterations to achieve error*		
				E = 0.1	*E = 0.01*	*E = 0.001*
SBP	$\eta = 3.0$			10	40	350
BP+M	$\eta = 1.5$	$\mu = 0.8$	$c = 0.1$	3	5	20
QP	$\eta = 0.2$	$\mu = 2.0$	$\nu = 0.0001$	8	12	15

Table 1. Comparison of learning algorithms as applied to user categorisation.

QP converges faster for small error but it is not very stable. BP+M is chosen as the best algorithm because it is stable and the rate of convergence is also good.

5.6.2. Accuracy of classification

There are 24 training data sets and 32 testing data sets. The training sets come from 6 subjects, each contributing 4 sets from the 4 different tasks. The testing sets consisted of 2 parts: the first 20 sets come from 5 of the 6 subjects producing the

training sets; the remaining 12 sets come from 2 subjects whose traces have never been presented to the network. One of the 2 subjects is regarded as a novice and the other as an expert. The result of the classification using the testing patterns is shown in Table 2.

	SBP	BP+M	QP
Correct Classification	29 (90.6%)	30 (93.8%)	30 (93.8%)
Incorrect Classification	3	2	2
TOTAL	32	32	32

Table 2. Overall successful rate of user categorisation using different learning algorithms

Out of the testing patterns, 12 are from subjects whose traces have not been presented to the network previously. Table 3 shows a comparison of the results from both seen and unseen subjects.

	SBP	BP+M	QP
Correct Classification (seen subjects)	17/20	18/20	18/20
Correct Classification (unseen subjects)	12/12	12/12	12/12

Table 3. Comparison of successful rate of seen and unseen subjects

The misclassifications in all three algorithms come from the same 2 patterns of a subject, with an additional one from another subject in SBP.

5.6.3. Accuracy in classifying unseen subjects

The accuracy in classifying traces of "u"unseen" subjects (those whose keystrokes have not been used in network training) were high in the pilot study. All of them are classified correctly. To explain this result, we note that unseen expert and novice subjects' behaviours are very prominent.

5.6.4. Relative effectiveness of training data set

The training sets and the testing sets are exchanged and re-tested. There are 2 incorrect classifications in SBP, 3 in both BP+M and QP. This represents a small drop in accuracy from 93.8% down to 87.5%, and indicates that the original training data set may have contained more representative patterns for classification.

5.7. SOURCES OF ERROR

(1) **Error due to keyboard used**

A lot of noise is generated in using the Domino keyboard (see Section 4.4.1), which has some important attribute differences from standard PC keyboards, with which subjects would be familiar, namely:

- Key arrangement -- the locations of the Esc, "<" and ">" keys on the Domino keyboard are very different from that on a standard PC keyboard. This introduces performance errors. (`Jove` uses the Esc as prefix for many hotkeys; "Esc <" and "Esc >" are fast navigation keys to go to the top and bottom of a file, respectively.)
- Key availability -- Ctrl-S and Ctrl-Q are reserved for the Domino keyboard for flow control and cannot be used. (`Jove` uses Ctrl-S as text-search hot-key and Ctrl-Q for quoting.) Although Ctrl-\ can replace Ctrl-S, the performance is erroneous
- Touch and feel -- the "feel" of the keyboard is not good, rendering typing less efficient than usual.

(2) **Error due to typing style**

The two incorrectly classified patterns come from the same subject who is a hunt-and-peck typist. Since there is no binding of keys and fingers [Newberry91] there may be a larger variation of typing timing that leads to errors in the result.

5.8. IMPLICATIONS FROM THE RESULTS OF THE PILOT STUDY

In the pilot study, a simple MLP was used to classify users of the `Jove` editor into 2 categories according to their level of expertise, with a successful classification rate of around 90% which was very encouraging. It indicated that, in the first place, neural networks are a *feasible* approach to the user modelling problem. Furthermore, the simple network architecture employed promises an economical and portable implementation. However, the subjects chosen in the pilot study belong to two extremes of a continuum – very proficient expert and very fresh novice, and we only do a simple partition of the population into 2 groups. If we proceed to include more users belonging from the middle of the continuum and classify the population into more levels of expertise, we should expect a drop in accuracy.

5.9. REMARK ON THE PILOT STUDY

A loss of information in the usage trace was discovered, which occurs when the subject holds down one of the four cursor keys to repeat. In VT100 terminals, cursors are coded with a 3-byte escape sequence. The monitor equipment was sending a 7-byte output for every character received. Thus it can be hard to keep up with the speed of terminal transmission in practice. Work was done by Michael Milway and David Wilson, technical support staff of Department of Computer

Science Department to resolve the problem. The monitor program was rewritten to send out only 4 bytes for every character received, and the terminal set to "limit transmit", restricting transmission to a maximum of 180 characters per second. Some other settings in the terminal and `Jove` were also performed (see Section 4.4).

For the Pilot Study this loss of information introduced a small amount of error. The usage traces need to be *manually* examined to cut away unreadable records.

6. FULL SCALE STUDY

6.1. SELECTION OF SUBJECTS

Twenty-two voluntary subjects were selected. The choice of discipline for both groups was mixed, with four from the Faculty of Commerce and the rest from the Faculty of Informatics at the University of Wollongong. Some of them use `Jove` solely as an email editor; others use it also in the writing of programs. Even for the subjects in Informatics, the frequency of using `Jove` varies. Three of them use the `vi` editor frequently but seldom use `Jove`. Their adaptation to the `Jove` editor differed greatly, depending on their differing backgrounds using the `vi` editor. Typing speed varied within the population: there were three hunt-and-peck typists, but also two capable of secretarial typing speeds. The rest were normal speed typists with varying rates of typing error. Initially, they were classified into three groups: "expert", "medium" and "novice", according to their performance in the use of `Jove`. This classification was performed through:

(1) Comparing usage trace file size and number of keystrokes used,

(2) Comparing the number of advanced features used,

(3) Comparing the efficiency in using advanced features, and

(4) Comparing the recorded/observed errors made in using advanced features

The inclusion of more second and third year Computer Science students as subjects enabled us to classify the population into a finer grain of expertise levels.

The subjects were given the same four tasks on two separate days a week apart, so each subject produced two sets of data. Since four of the subjects could not arrange times for the second session, their usage traces were used for testing only. So altogether there were 72 (18x4) training sets and 88 (22x4) testing sets.

6.2. PROBLEMS ENCOUNTERED IN THE CLASSIFICATION INTO THREE LEVELS OF EXPERTISE

We modified MLP-0 by changing the single output node to three nodes E, M and N, each representing one expertise level – expert, medium and novice. Only one of three nodes is turned on at one time during training. We named this 31-3-3 network MLP-1.

We attempted to apply the same procedure as in pilot study and use MLP-1 to classify the 22 subjects into 3 groups – expert, medium and novice. However, the network failed to converge, producing an error E=2.0 which is not acceptable. If we go back and partition the subjects into 2 groups – expert and novice – as before, we can obtain an accuracy of 80% to 90%.

The Kohonen's Self-Organised Map (SOM) network was used to ascertain if there exists any natural grouping similar to the 3 intended categories but the results were not encouraging.

The questions that needed to be answered were:

(1) Is the learning algorithm good enough?

(2) How can we better represent expert, medium and novice users as ranges in a continuum?

(3) How are we going to partition experts and medium users? Are the features tracked enough to do the job?

(4) Are the tasks substantial enough to separate? Should we separate the tasks in training?

6.3. MODIFICATIONS MADE TO IMPROVE ACCURACY

6.3.1. Using the RPROP learning algorithm

RPROP is found to produce faster and more stable convergence, though not better error (Figure 8).

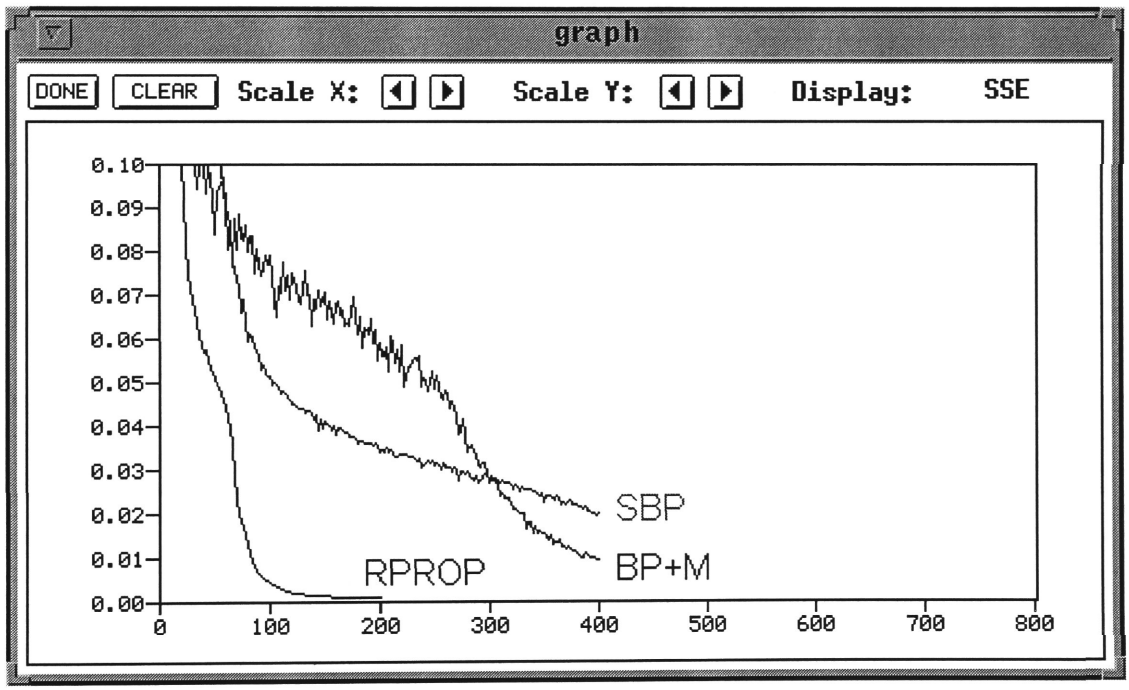


Figure 8. Comparison of RPROP with SBP and BP+M
x-axis: error; y-axis: number of training cycles

RPROP stands for "Resilient Backpropagation" and is a local adaptive learning scheme, performing supervised batch learning in MLPs. The basic principle of RPROP is to eliminate the harmful influence of the partial derivative on the weight step [Zell95]. As a consequence, only the sign of the derivative is considered to indicate the direction of the weight update. There are 3 parameters in SNNS to set:

$delta_0$: starting value of all $delta_{ij}$, (default is 0.1),

$delta_{max}$: upper limit for the update value $delta_{ij}$, (default is 50.0), and

α: weight decay parameter, a value of 4 corresponds to a ratio of weight decay term to output error of $1:10^4$

The choice of $delta_0$ is uncritical as it is adapted as learning proceeds. Convergence is usually insensitive to $delta_{max}$

6.3.2. Representing expertise levels in fuzzy form

Fuzzy representation was proved to give better acceptable error E for all tasks.

Fuzzification

Instead of setting either one of the three output nodes to 0 or 1, we use:

Class	***Output Nodes***		
	E	*M*	*N*
Expert	0.75	0.25	0
Medium	0.25	0.50	0.25
Novice	0	0.25	0.75

Table 4. Fuzzification of 3-level expertise

Defuzzification

We use the centre of gravity algorithm in defuzzification. It can be achieved using the following steps:

(1) Calculate Normalised Membership for E, M and N by:

NormMemb(X) = Membership(X) / Sum of Memberships

(2) Calculate the ClassValue

ClassValue = [NormMB(E)*3+NormMB(M)*2+NormMB(N)*1] / Σ(NormMB(x))

where Σ(NormMB(x)) =1, since normalised

(3) Assign Class by noting value of Class Value

Class = {Expert, if 2.5 < ClassValue;

Medium, if 1.5 <= ClassValue < 2.5;

Novice, if ClassValue < 1.5 }

6.3.3. Tracking individual hotkeys

MLP-1 include nodes that represent a high level summary of editing, navigation, blocking search and replace group of keys. Here we abandon these nodes (9 altogether) and add to the network nodes that track individual hotkeys. Nodes (36 altogether) are added to indicate if a certain hotkey has been used (by setting the node to 1). They include advanced navigation operators, editing operators, blocking operators, text search operators, text replace operators and two nodes for command mode access and help (Appendix H). The new 58-3-3 network is named MLP-2.

MLP-2 gives improved convergence, but misclassifications still occur frequently, mostly with *both* the expert and medium nodes turned on.

6.3.4. Separating tasks in training and testing

We would expect more complicated tasks like Task 1 (Creating a memo) and Task 4 (Enhancing and Spell-checking a memo) to give better results when trained separately. The patterns are presented separately in this case. The convergence is better, especially for Task 4 and Task 2 (Address a memo). Surprisingly Task 1 does not give good convergence. As we traced the source of error, we found that some expert and medium subjects did not use faster methods and operators that they know to accomplish the task. Rather, they just keep on typing blindly. This suggests the existence of inconsistency in the user model. With Task 4, the accuracy classification of internal tests is 100% with E=0.001. However, the accuracy of classification of external tests (unseen pattern) is only 60%. Compared with the result obtained in the pilot study (around 90%) this is not favourable.

In the subsequent refinements below, it is found that improvements continue with Task 4, but not with the other three tasks. Accordingly, the latter are removed from

the subsequent discussion. The major reason for a lack of improvement is most likely because the three tasks are not substantial enough to allow classifying into more than two levels of expertise.

6.3.5. Changing fuzzy output to represent five expertise levels

The misclassification using three levels (where the output nodes E and M are turned on simultaneously) has inspired the refinement by adding more expertise levels. It is expected that some subjects fall midway between expert and medium user. We introduced two more expertise levels: one between expert and medium, and another between medium and novice. The classification accuracy improved. Such representation also gives better acceptable error for all tasks. For Task 1, E=0.06; for Task 2, E=0.001; for Task 3, E=0.2 (worst); for Task 4, E=0.001 (best).

Fuzzification

Class	***Output Nodes***		
	E	*M*	*N*
Expert	0.75	0.25	0
Expert-Medium	0.50	0.50	0
Medium	0.25	0.50	0.25
Medium-Novice	0	0.50	0.50
Novice	0	0.25	0.75

Table 5. Fuzzification of 5-level expertise

Defuzzification

We use the centre of gravity algorithm in defuzzification. It can be achieved using the following steps:

(1) Calculate Normalised Membership for E, M and N by:

NormMemb(X) = Membership(X) / Sum of Memberships

(2) Calculate the ClassValue

ClassValue = [NormMB(E)*3+NormMB(M)*2+NormMB(N)*1] / Σ(NormMB(x))

where Σ(NormMB(x)) =1, since normalised

(3) Assign Class by noting value of Class Value

Class ={ Expert, if 4.5 < ClassValue;
Exp-Med, if 3.5 <= ClassValue < 4.5;
Medium, if 2.5 <= ClassValue < 3.5;
Med-Nov, if 1.5 <= ClassValue < 2.5;
Novice, if ClassValue < 1.5 }

6.3.6. Tracking hotkey timing

It was intended to enhance the granularity of classification by tapping the timing information for individual keys. However this proved not to be successful. Firstly, a lot of the hotkeys are of Ctrl-[x] sequences whose latency time is screened by the keyboard and so cannot be recorded by the monitor equipment. Only a few Esc-[x] sequences are left. "Esc >" (bottom of file) and "Esc <" (top of file) have large variances due to the unpleasant location of "<" and ">" on the keyboard. Surprisingly, "Esc f" (forward word) and "Esc b" (backward word) also have a large variance and could not be used. This leaves only "Esc P" (previous page), which does not help a lot.

6.3.7. Tracking efficiency patterns

The assumption "Experts tend to work more efficiently" is used mainly to classify *expert, exp-med* and *medium* users. All three classes of users may know how to use some specific hotkeys to accelerate navigation in the document. When they use any one of these features, a specific input node is turned on. However, such a single node cannot tell how well the subject is using this feature in real situations. We assume

that expert users tend to use a better mix of these hotkeys to accomplish their tasks, so in general a more expert-oriented subject tends to be more efficient than a less expert-oriented subject.

For example, there are several ways to reach a word near the end of line,

slowest way :	cursor right char-by-char from beginning of line until destination is reached
slow way :	cursor down one line, cursor left to cross line boundary, and cursor left a few characters
less fast way :	forward word-by-word several times, and cursor right a few characters;
fastest way :	jump to end-of-line, backward-word, and cursor left a few characters;

Remark:

(1) We do not imply that expert and efficiency is a one-to-one mapping!

(2) Even experts sometimes show inconsistency in their interaction efficiency.

The idea of an efficient navigation is that a long sequence of right-arrows, say, can be replaced by a [forward word], [end-of-line] or [end-of-line, backward word] depending on the length of the sequence. This applies to other navigation keys, as well.

Defining efficiency terms for cursor keys

We pay attention to situations where the cursor is pressed many times (over a certain threshold) and can be replaced by a hotkey to speed up the operation. The "actual number of cursor keys used" and "minimal number of cursor keys required" is used as a metric to represent the efficiency of that cursor key.

METRIC

Minimal keystrokes = Total no. of keystrokes actually used -
SUM (no. of over-threshold cursor blocks* threshold_size)

e.g. for cursor_left ,

Minimal keystrokes = Total no of keystrokes actually used -
(blocks over word_threshold * length of word_threshold +
blocks over line_threshold * length of line_threshold)

Thresholds selected for left and right arrows

Using the word count program "wc", the average word length of the test document is found to be below 10. The document page width is 70. A line threshold is about one word less than that. So

WORD_THRESHOLD : 10
LINE_THRESHOLD : 60

Thresholds selected for up and down arrows

The thresholds selected for up and down arrows is given by the page height of the 80x24 terminal.

PAGE_THRESHOLD : 24

Definition

Efficiency = best no. of keystrokes/ current no. of keystrokes

= 1 - (number of keys that can be reduced / total keys used)

which is always within the range 0-100%

The efficiency of each cursor key (Left/Right/Up/Down) is divided into levels from reading the performance of subjects practically:

Left/Right: (0-0.3, 0.3-0.5, 0.5-0.7, 0.7-0.9, 0.9-1.0)
Up/Down: (0-0.5, 0.5-0.7, 0.7-0.9, 0.9-1.0)

Each level is represented by one node, so altogether 18 nodes are added.

6.3.7.1. Define efficiency term for backspace

Applying a similar idea to the keys [delete] and [backspace], a long sequence of [delete char], say, can be replaced by a [delete word], [delete to end-of-line] or [block-mark-and-cut], depending on the length of the sequence.

METRIC

Recall,

WORD_THRESHOLD : 10
LINE_THRESHOLD : 60
PAGE_THRESHOLD : 24

Definition

Efficiency	**= best no. of keystrokes/ current no. of keystrokes**
	= 1 - (number of keys that can be reduced / total keys used)

From reading the performance of subjects practically, we decided that the [delete] key efficiency is not a good discriminator, and for [Backspace], four levels are enough to represent its efficiency. Each level is represented by one node, so altogether 4 nodes are added.

BS Efficiency: (0-0.5, 0.5-0.7, 0.7-0.9, 0.9-1.0)

6.3.7.2. Define efficiency term for deletion

A very inefficient way of deleting characters is the use of the combination [cursor_right]+[backspace]. It can be replaced by a single [delete char] key.

METRIC

Attention is paid to the pattern of "cursor right *x* times, then backspace *y* times".

A figure of y' = minimum(x,y) is taken for each occurrence of such a pattern, and these are summed to give the total number of backspaces that are accompanied by a right arrow, i.e.

SUM(y').

Another figure is the total number of backspaces actually used in the session., i.e.

SUM(y).

Ideally, SUM(y') should approach 0.

Definition

Efficiency in Deletion by Backspace
= 1 - (ratio of SUM (y') to SUM (y))

which is always in the range 0-100%. Note that an expert may use "another replacement method", like [Replace], [Query-replace], or [Cut] & [Paste], but it is difficult to devise metrics to include these. From reading the performance of subjects practically, we decide that five levels are sufficient to represent their efficiency. Each level is represented by one node, so altogether 5 nodes are added.

Deletion Efficiency: (0-0.3, 0.3-0.5, 0.5-0.7, 0.7-0.9, 0.9-1.0)

6.3.7.3. Change in network architecture

We modified MLP-2 (58-3-3) to include the efficiency pattern classifiers (altogether 27 nodes), so it becomes a 85-3-3 MLP. We name this network MLP-3

6.3.7.4. Results of applying efficiency pattern

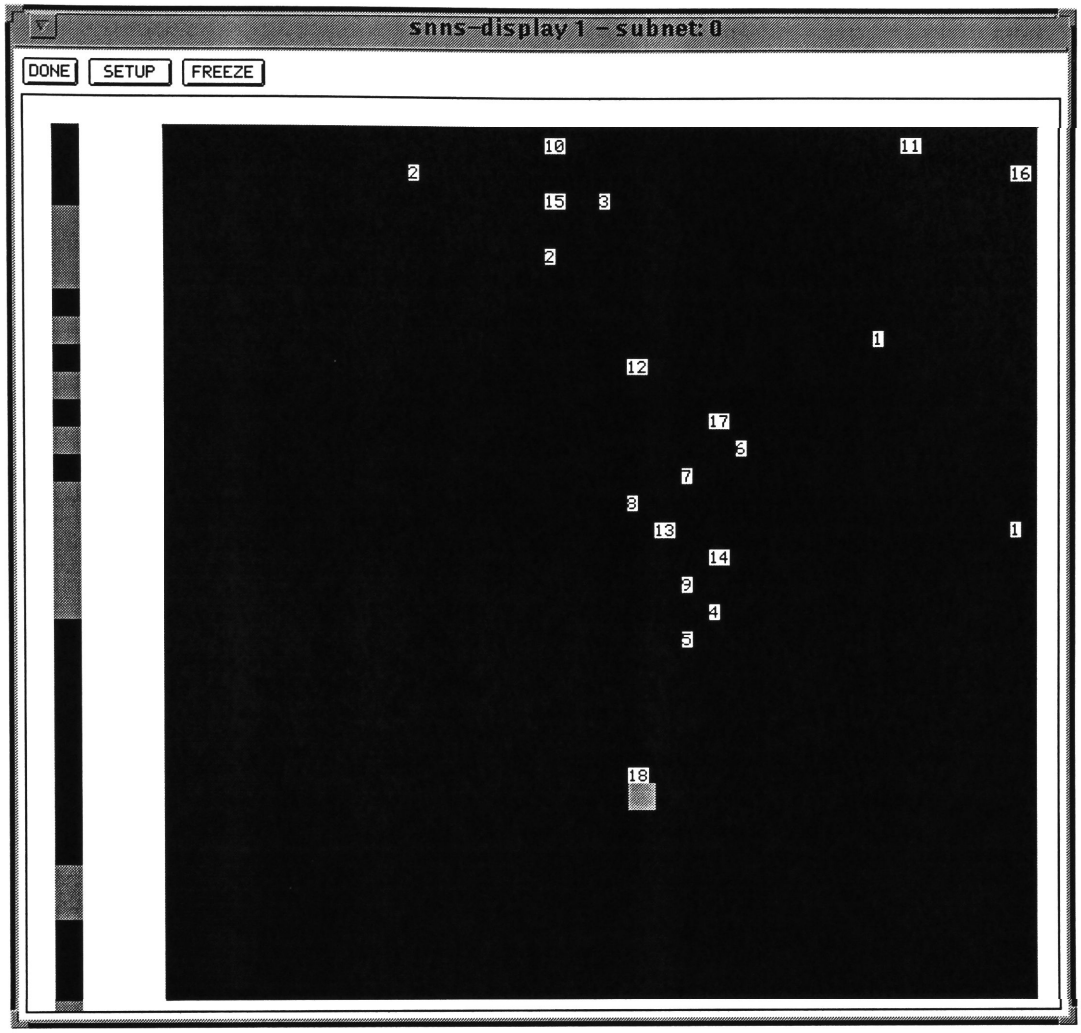
We measure the improvement of the efficiency pattern by testing the classification accuracy before and after applying the efficiency pattern classifiers. This shows that the classifier has significant improvement in the classification accuracy.

		Before	After
Test1			
	correctly identified	59%	70%
	within adjacent classes	95%	100%
Test2			
	correctly identified	67%	77%
	within adjacent classes	95%	95%

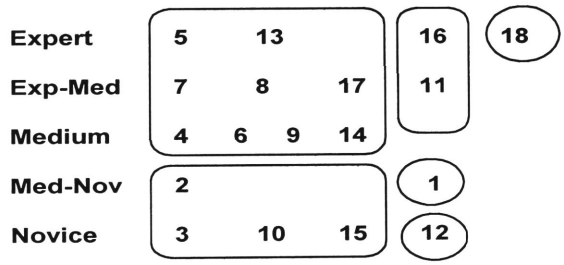
6.3.8. Enhancing coding of all level representations

6.3.8.1. The old coding method

We attempted to use SOM to visualise the natural grouping of the input patterns, but without success. The output of SOM shows that we are trying to input something that is difficult to classify. The expert, exp-medium and medium merely merge into a single group. A similar situation applies to Med-Nov and Novice (Figure 9).



(a) Graphical output of SOM in SNNS after training of 10 cycles. The nearby elements are supposed to have greater similarity.



(b) The grouping as depicted in (a) in a view corresponding to the expertise level. Circled elements are supposed to be in the same group.

Figure 9 Visualisation of natural group by 32x32 SOM network under old coding method

Assume a feature is represent by 4 levels. The old coding method uses the following to represent the different levels:

Level 1: 1 0 0 0
Level 2: 0 1 0 0
Level 3: 0 0 1 0
Level 4: 0 0 0 1

While this coding method works fine for MLPs, it is not a good representation for SOM. Consider the similarity of Level 2 to Levels 1, 3 and 4. The Hamming distance in each case is 2, meaning that there are 2 bit differences for each pair. If this single feature is put into SOM, it will be difficult to classify the levels by virtue of their similarity. Only patterns of the same level are grouped together, but the relative distances of different levels are not in proportion.

6.3.8.2. The new coding method

We want a coding method that can group patterns of the same level into one group, as well as pulling neighbouring levels nearer and pushing non-neighbouring levels further apart. The new coding system works like this: instead of turning a single node ON, all nodes below are turned on simultaneously. The 4-level feature shown above should be represented by:

Level 1: 1 0 0 0
Level 2: 1 1 0 0
Level 3: 1 1 1 0
Level 4: 1 1 1 1

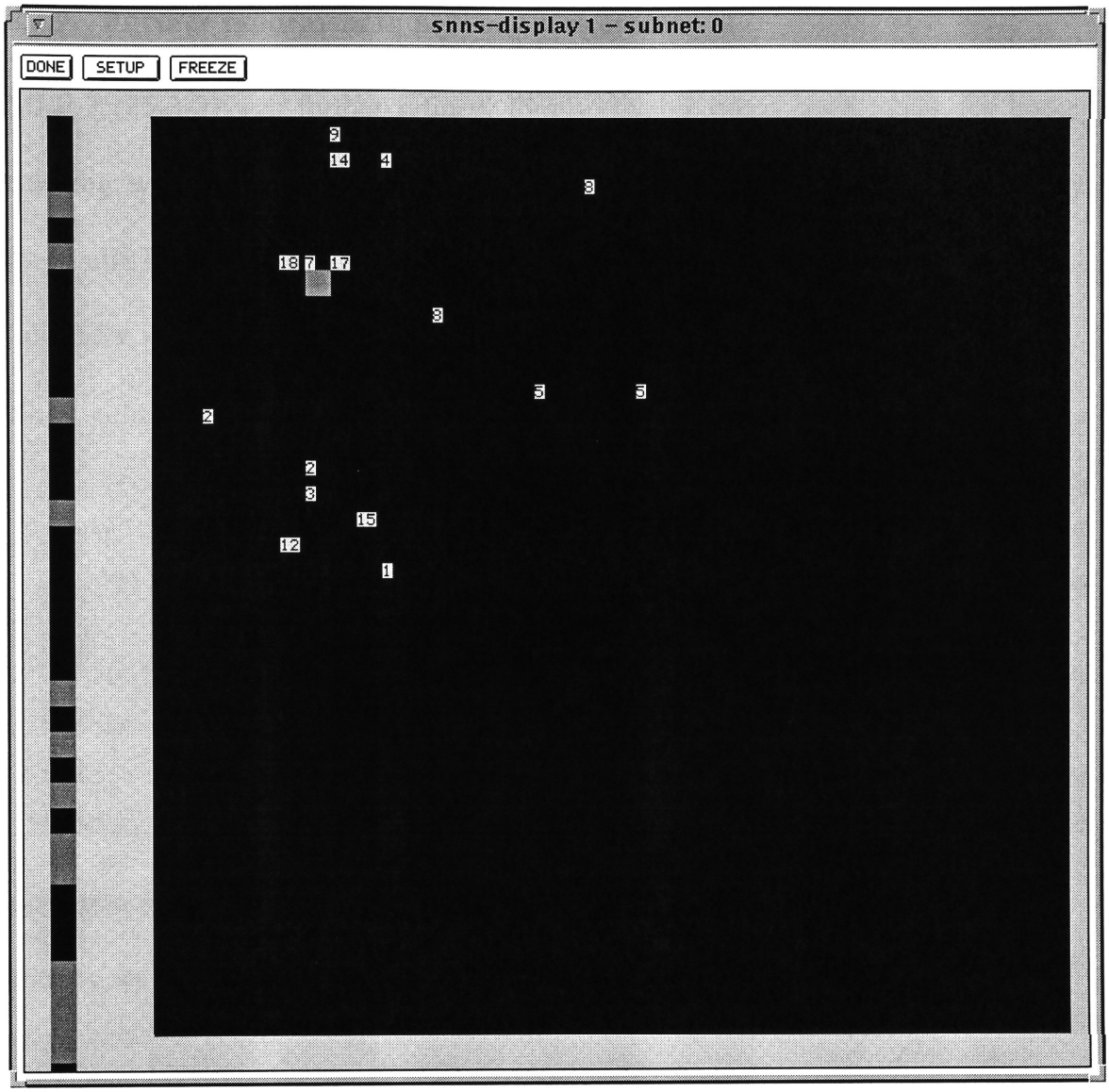
Consider the similarity of Level 2 to Levels 1, 3 and 4 again. The Hamming distances of Level 1>>2 and Level 3>>2 are 1 respectively, while that of Level 4>>2 is 2. If we put this into SOM, Level 1 and Level 3 patterns should be in closer proximity to Level 2 than to Level 4. Level 1 and Level 4, having a Hamming distance of 3 between them, should be pushed far apart.

6.3.8.3. Results of applying the new coding method

The new coding method gives a slight improvement to the accuracy of recognition using MLP.

		Before	After
Test1			
	correctly identified	70%	77%
	within adjacent classes	100%	95%
Test2			
	correctly identified	77%	77%
	within adjacent classes	95%	95%

The improvement of grouping in SOM is shown in Figure 10.



(a) Graphical output of SOM in SNNS after training. The nearby elements are supposed to have greater similarity.

Expert	**5**	**16**		**18**	
Exp-Med	**8**	**11**	**13**	**17**	**7**
Medium	**4**	**9**	**14**	**6**	
Med-Nov	**1**	**2**			
Novice	**3**	**12**	**15**	**10**	

(b) The grouping as depicted in (a) in a view corresponding to the expertise level. Circled elements are supposed to be in the same group.

Figure 10. Visualisation of natural group by 32x32 SOM network under new coding method

Comparing Figures 10(b) to Figure 9(b), we can see a prominent improvement in differentiation of "Expert" and "Medium" users.

6.3.9. Further refinement in network architecture

MLP-3 (85-3-3) was further refined. Firstly, twelve rarely used nodes for hotkey tracking were purged from the network to improve training speed and accuracy; Appendix I list the remaining hotkeys being tracked.

Secondly, the old links from individual operators to the hidden layer were removed.

Thirdly, one more hidden layer was added, containing:

(1) nodes for summarizing the effects of Individual navigation operators and Navigation Key Efficiency Patterns.
(2) nodes for summarizing the effects of Individual editing operators and Editing Key Efficiency Patterns.
(3) nodes for summarizing the effects of all individual operators
(4) nodes for summarizing the effects all efficiency patterns.

This added hidden layer (of 12 nodes) was connected to the next hidden layer when one more node is added.

Finally we have a (73-12-4-3) MLP, which we name MLP-4 (Figure 11).

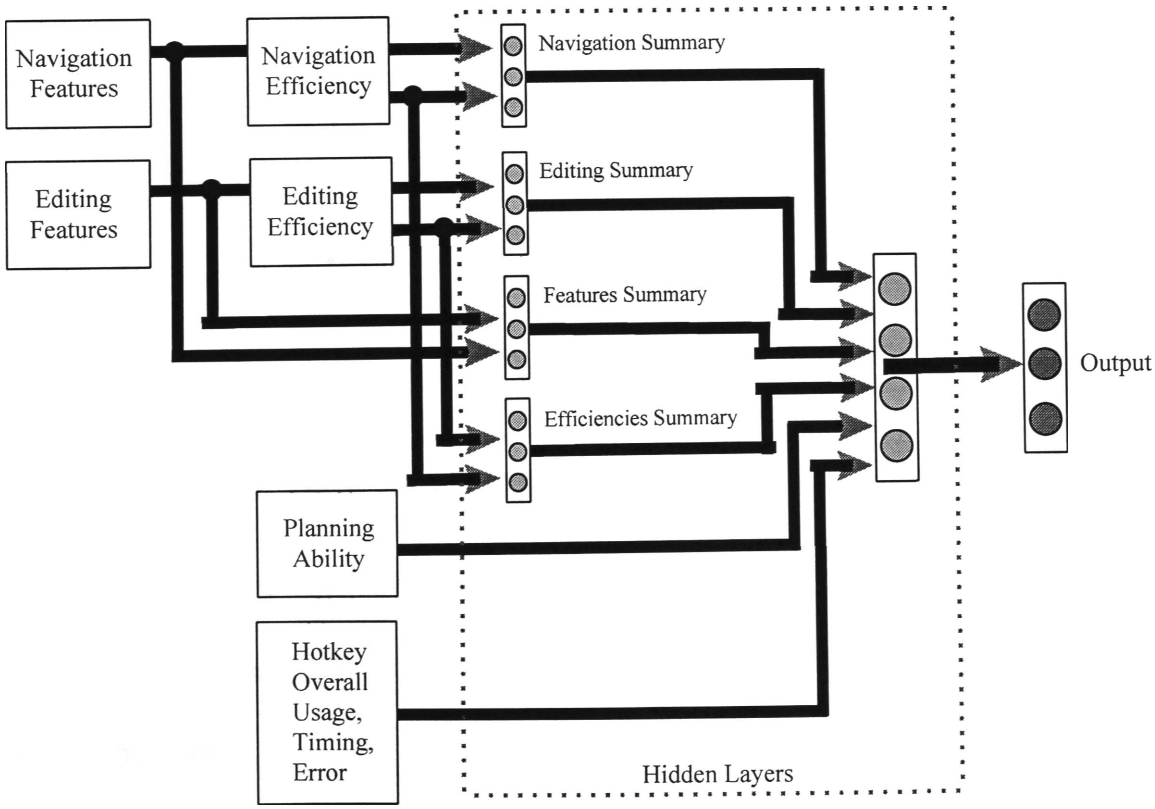


Figure 11. Schematic diagram of MLP-4

6.4. SUMMARY OF THE FINAL SETUP

- The network architecture is enhanced to a 73-12-4-3 network (MLP-4) which gives improved results.
- The learning algorithm RPROP is used for MLP training to improve the convergence rate and stability.
- A new input coding system is applied to improve grouping in SOM.
- Expertise Level is represented in fuzzy form to improve convergence.
- Expertise is represented in a finer grain of 5 Levels (it is shown that this representation gives better classification accuracy).
- Training is separated for different tasks and the most substantial task (Task 4) is used for ongoing tests.
- Features tracked in the final network:
  - Track 24 individual hotkeys
  - Track 6 Efficiency Patterns
  - Track the overall usage, overall timing and overall error of hotkeys
  - Track the planning ability of subject

The hotkey timing cannot be tracked because of noise introduced by the Domino keyboard.

6.5. PERFORMANCE OF THE FINAL SETUP

We trained this network with the RPROP learning algorithm and obtained in less than 100 cycles an acceptable error (E=0.001). The convergence rate was similar to the old network.

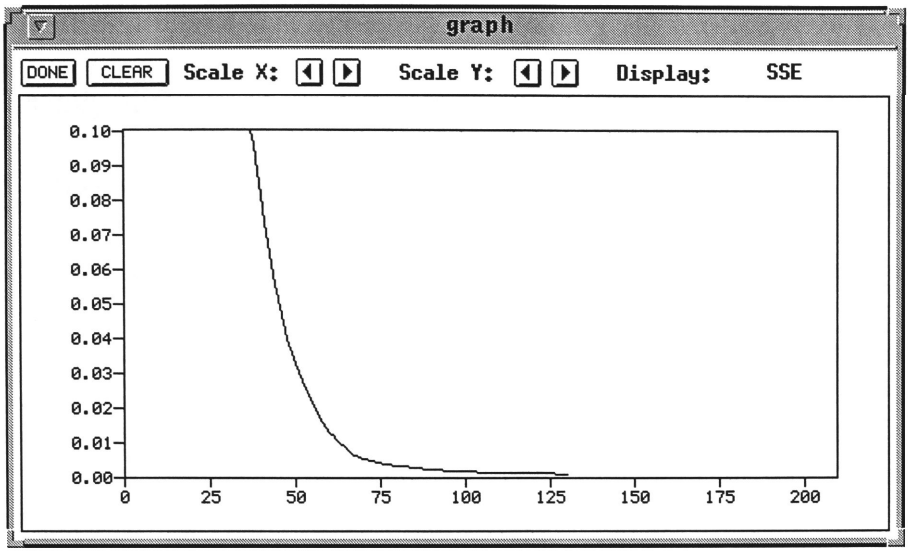


Figure 12. Convergence when applying RPROP learning algorithm to MPL-4
x-axis: error; y-axis: number of training cycles

We tested classification accuracy again and found that the performance was improved slightly. Furthermore, this network gives a better visualisation of the abstraction.

		Before	After
Test1			
	correctly identified	77%	82%
	within adjacent classes	95%	95%
Test2			
	correctly identified	77%	72%
	within adjacent classes	95%	100%
Average			
	correctly identified	77%	77%
	within adjacent classes	95%	97.5%

6.6. BALANCING TRAINING PATTERN CLASS TO IMPROVE ACCURACY

One important factor that affects the classification accuracy is *infrequent class*. The neural network dominantly trains the frequent patterns while leaving the information of infrequent patterns uncaptured. Existence of poorly trained patterns has considerable influence on the performance of the backpropagation model

[Cheung90]. It is obvious that enlarging the training set can improve the accuracy, but it is usually difficult to obtain a sufficient number of training examples for infrequent classes. Cheung proposed a modification to the BP algorithm by constructing a dynamic training set which changes according to the error. We are not using this more complicated algorithm in training, but instead borrow the idea of balancing the training set with equally sized classes. For example, consider a training set with the following class distribution:

```
Expert          3
MedExp          6
Medium          6
NovMed          2
Novice          5
```

Information of classes `Expert` and `NovMed` are not captured as much as other classes. We randomly replicate existing examples in the infrequent classes to make the class sizes balanced:

```
Expert          6
MedExp          6
Medium          6
NovMed          6
Novice          6
```

The replicated examples are only used for training and NOT counted towards classification accuracy in the testing. A slight improvement in classification accuracy is obtained as below:

		Before	After
Test1			
	correctly identified	82%	82%
	within adjacent classes	95%	100%
Test2			
	correctly identified	72%	77%
	within adjacent classes	100%	100%
Average			
	correctly identified	77%	77%
	within adjacent classes	97.5%	100%

6.7. THREE-LEVEL CLASSIFICATION REVISITED

With MLP-4, we revisited the 3-level classification by another set of pattern files.

		MLP-1	MLP-2 (track hotkeys)	MLP-4 (w/ all refinements)
Test1				
	correctly identified	(did not converge)	59%	80%
Test2				
	correctly identified	(did not converge)	67%	80%

In Test 1, all four errors occurred when misclassifying a Medium user as an Expert. In Test 2, three errors correspond to identifying a Medium user as an Expert, and one to identifying a Novice user as a Medium user.

7. BENCHMARKING

Neural networks have been successful in the pattern recognition of visual images [Beale90]. Its performance in the application to user modelling, however, has seldom been compared with other learning approaches. In this study, we implement versions of user expertise modelling using production rule system (or expert system) and inductive system approaches, in order to benchmark our neural network approach.

7.1. FUZZY PRODUCTION RULE SYSTEM

7.1.1. Problem in constructing a fuzzy production rule system for benchmarking

It would be unfair to benchmark the fine-tuned neuro-fuzzy system against a production rule system that is set up with little attention paid to detail. Firstly, we need to choose an equally powerful system for comparison. Secondly, we need to ensure that the set of production rules is optimised for better performance. Thirdly, we need to assign credits/weightings to each contributing factor. By doing so we ensure the production rule system is also fine-tuned. However, this is easier said than done.

The first difficulty is that the production rule system cannot learn the "rules" and weightings through supervised training as in neural networks. We have to define each rule manually. In a system of user interaction with so many attributes, the

dimension can be enormous. Recall that we have included several classes of inputs in the neural network: feature operators, efficiency operators, planning ability, hotkey usage, timing and error. Then what should be the dimension of the production rules? Can we write production rules using all these classes of inputs? Hopefully so, but in practice it is impossible to define rules manually involving all classes of input. In any case we have to reduce the dimension by neglecting some input classes. Of course this omission implies information loss and this sacrifice the accuracy to a certain degree. The second difficulty is the determination of the contribution of each factor to the classification. In the neural network, the contribution is in the form of the weight of the link between nodes and the weight is automatically learned during network training. Here we have to assign the credit subjectively.

If we start with writing our rules and determining contributions subjectively, we are likely to come up with a production system that is inferior to the neuro-fuzzy approach. This is not what we want. To take a more pragmatic approach, we assume that we possess some knowledge in writing appropriate production rules and assigning appropriate credits. We use the knowledge coming from our findings in the neuro-fuzzy approach to empower the production rule system. The following points are adopted in constructing a more powerful production system:

(1) *Choice of system.* We choose a production rule system that is capable of handling fuzzy facts and fuzzy logic. Such a choice aligns the capability of the production rule system with the neural network system in terms of fuzzy orientation.

(2) *Choice of Input.* We cater for rules with only 24 feature operators (Appendix I). The other 12 rarely used feature operators are purged to increase accuracy. This aligns with the neuro-fuzzy system in moving from MLP-2 to MLP-4. We will use both feature operators and efficiency operators.

(3) *Credit Assignment.* We assign weights to each class of input (feature range versus efficiency) and to each input according to hints provided by the weights of the neural network links. We recognise the importance of efficiency operator class and give it the same weighting as feature operator class, though the latter accounts for 24 inputs and the former 6.

However, since we do not have knowledge of the relationships between all input classes, we cannot include all classes of inputs in writing production rules. Moreover, we have to reduce the dimension of the input so that we can manage to write out production rules. Input classes like planning ability, hotkey usage and timing are taken out.

7.1.2. Approach

In designing the production rule system, we try to employ fuzzy representation and extend the use to both input and output. The pattern file for training the neural network is modified for testing (there are 40 patterns). The output is the five levels of expertise in using `Jove` editor.

The development environment

The system is developed under FuzzyCLIPS Version 6.02A for MS Windows. FuzzyCLIPS [NRC94] is an enhanced version of CLIPS 6.0 [Giarratano93] with fuzzy data representation. CLIPS stands for C Language Implementation Production

System and was developed by the Artificial Intelligence Section of the Johnson Space Centre of NASA. CLIPS is a forward-chaining rule-based system based on a pattern-matching algorithm. It provides the following beneficial features for our study:

- Mix of crisp and fuzzy fact
- Choice of inference method, defuzzification method, threshold of certainty factor
- Support of modifier (extremely, very, not...)
- Support of file I/O
- Support of Plotting of fuzzy value
- Availability in Windows and UNIX platforms

The listing of the CLIPS program can be found in Appendix J.

7.1.4. Fuzzification of input and output

The range of advanced features used (Feature-Range Operator) and the efficiency of using advanced features (Efficiency Operator) are correlated to give different levels of expertise. Other inputs (planning ability, hotkey usage/timing/errors) cannot be included as we have difficulties in coding the relationship in rules.

Feature-range operator

Definition: The number of advanced feature operators used, normalised by the maximum of an expert user, to a range of 0-4, triangle width 0.7 (allowing proper overlapping). The membership function of Feature-Range Operator is shown in Figure 13.

Weighting of feature-range operators

The 24 feature operators are assigned equal strength for simplicity.

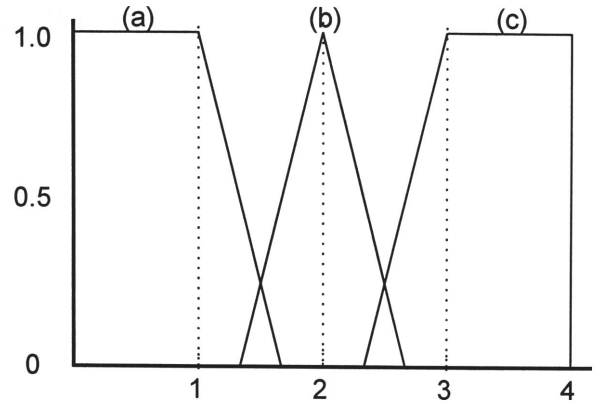


Figure 13. Membership function for feature-range operator
Linguistic Value: (a) Limited, (b)Medium, (c) Broad

Efficiency operator

Definition: Weighted Average of Cursor-Efficiency (cursor left/right/up/down operators), and Edit-Efficiency (backspace and delete character operators), normalised to a range of 0-4, triangle width 0.7 (allowing proper overlapping). The membership function of Efficiency Operator is shown in Figure 14.

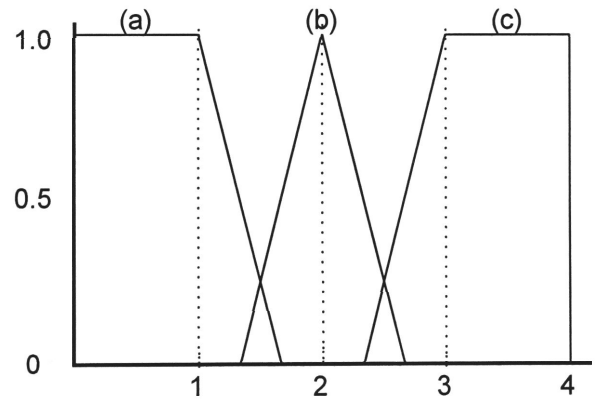


Figure14. Membership function for efficiency operator
Linguistic Value: (a)Low, (b)Medium, (c)High

Weighting of efficiency operators

There are 6 efficiency operators: (cursor_up, cursor_down, cursor_left, cursor_right, backspace, deletion). We found that some efficiency operators are more crucial to identify expertise level than others, we assign a higher weighting to *efficiency_deletion, then *efficiency_backspace and lastly the cursor usage

efficiency operators. This can be read from the listing of the CLIPS program in Appendix J.

Expertise level

Mapped by production rule from fuzzy sets of Feature-Range Operator and Efficiency Operators into a range of 0-6, width 0.7 (allowing proper overlapping). The membership function of Expertise Level is shown in Figure 15.

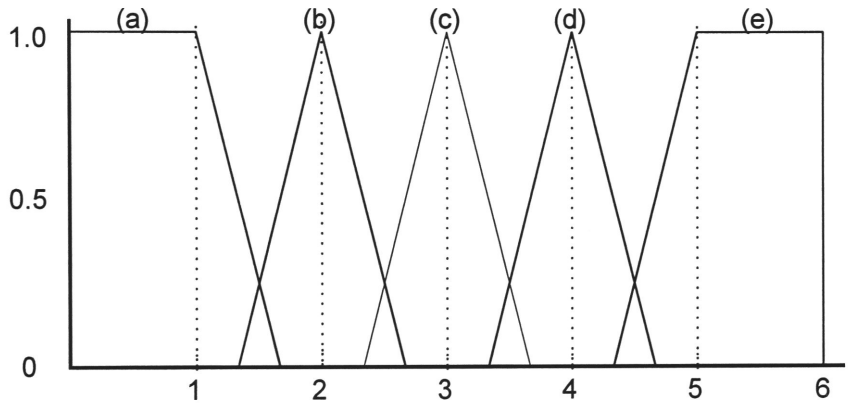


Figure 15. Membership function for expertise level
Linguistic Value: (a)Novice, (b)NovMed, (c)Medium, (d)MedExp, (e)Expert

Choice of class width

The width of class and pattern are *carefully chosen.*

1. The distance between the mean of each class is 1.0 and a class width of 1.4 is chosen. This distance allows a proper overlap of 0.4 (i.e., 28.6% of class width) between adjacent classes and zero overlap for others.
2. Since the user behaviour tends to oscillate within a certain range instead of exactly at a specific point, the values of feature-range and efficiency operators are fuzzified by a triangle instead of a straight line, as in Figure 16. The width of the triangle is 0.7, which is half of the class width. This is a compromise between neighbourhood contribution and class focusing.

For such an overlap, 2 rules may be fired for each operator, which amounts to a maximum of 4 rules (2 x 2) fired for each pattern.

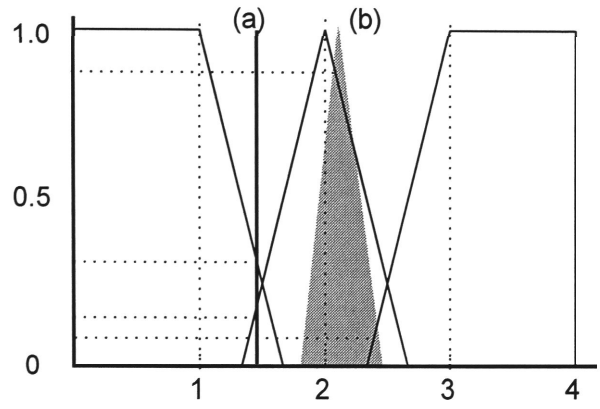


Figure 16. Fuzzification of operators in the study
(a) usual representation (b) representation taking into account the variance

7.1.5. Production rules

The production rules map the permutations of Feature-Range and Efficiency Operators (3x3) into 5 Expertise Levels (Novice, Novmed, Medium, MedExp, Expert). A schematic diagram of the mapping is shown in Figure 17 and the rules are shown in Figure 18.

Efficiency Operator	***Feature-Range Operator***		
	Broad	*Medium*	*Limited*
High	**Expert**	**MedExp**	**Medium**
Medium	**MedExp**	**Medium**	**NovMed**
Low	**NovMed**	**Novice**	**Novice**

Figure 17. Schematic diagram showing how production rules maps inputs to output

```
;;; Production Rules
;;; - mapping Feature-Range and Efficiency Operator into Expertise Levels

(defrule broad-high
   (feature-range broad)
   (efficiency high)
=>
   (assert (expertise expert))
)

(defrule broad-medium
   (feature-range broad)
   (efficiency medium)
=>
   (assert (expertise medexp))
)

(defrule broad-low
   (feature-range broad)
   (efficiency low)
=>
   (assert (expertise medium))
)

(defrule medium-high
   (feature-range medium)
   (efficiency high)
=>
   (assert (expertise medexp))
)

(defrule medium-medium
   (feature-range medium)
   (efficiency medium)
=>
   (assert (expertise medium))
)

(defrule medium-low
   (feature-range medium)
   (efficiency low)
=>
   (assert (expertise novmed))
)

(defrule limited-high
   (feature-range limited)
   (efficiency high)
=>
   (assert (expertise novmed))
)

(defrule limited-medium
   (feature-range limited)
   (efficiency medium)
=>
   (assert (expertise novice))
)

(defrule limited-low
   (feature-range limited)
   (efficiency low)
=>
   (assert (expertise novice))
```

Figure 18. Production rules of fuzzy expert system

Defuzzification

The fuzzy expertise level is defuzzified by the *Centre Of Gravity* (COG) algorithm into a crisp representation (5 levels) using the following classification:

```
Class ={ Expert,      if 4.5 < ClassValue;
         Exp-Med,     if 3.5 <= ClassValue < 4.5;
         Medium,      if 2.5 <= ClassValue < 3.5;
         Med-Nov,     if 1.5 <= ClassValue < 2.5;
         Novice,      if ClassValue < 1.5      }
```

7.1.7. Results

The *Max-Prod Inference* is used, as it preserves more information than Max-Min Inference.

40 patterns were used to test the fuzzy production rule system. The construction of these patterns had benefited from the findings in the neuro-fuzzy approach in removing irrelevant inputs and assigning credits to various contributing factors.

The result is shown in Table 6. The classification accuracy of the fuzzy production rule system was 52.5% compared with 79.5% with the neuro-fuzzy system.

Best Result Obtained	***Fuzzy Production Rule System***	***Neuro-Fuzzy System***
Correctly identified	52.5%	79.5%
within adjacent classes	97.5%	100%
within 2-classes apart	100%	100%

Table 6. Performance of fuzzy expert system versus neuro-fuzzy system

7.2. INDUCTIVE SYSTEM

7.2.1. Problem in constructing an inductive system

Since the inductive system can learn automatically, we do not have the problem in selecting appropriate rules and assigning reasonable credits to contributing factors as we did in the production rule system. We can include all classes of inputs and let the

inductive system determine the rules and the contributions. However we still take advantage of the findings in the neuro-fuzzy approach in purging some rarely used inputs to reduce the dimension of the input. One point worthy of note is that there is no fuzzy counterpart for the inductive system.

7.2.2. Approach

Rules are extracted from the set of training examples. The extracted rules are tested against the training examples and testing examples. The pattern file for training the neural network is modified to be used for testing (there are 40 patterns). The output is the five levels of expertise in using `Jove` editor.

7.2.3. The development environment

The system used is C4.5 running under UNIX. C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on [Quinlan93]. It extracts rules automatically from a set of training examples and is a kind of supervised learning.

7.2.4. Definition of categories

Categories must be established beforehand. They are put in the file <Task>.Names.

The order is NOT important. The format is:

Line 1:	Probable Outputs
Lines 2 onwards:	Input Categories: Range of Values

Figure 19 lists the content of the <Task>.Names file.

```
expert, medexp, medium, novmed, novice.

name:                          ignore.

edit_delete-current:           0, 1.
edit_delete-end-of-line:       0, 1.
edit_delete-word:              0, 1.
edit_case-lower:               0, 1.
edit_case-upper:               0, 1.
edit_case-proper:              0, 1.
edit_exchange:                 0, 1.
navigate_begin-of-line:        0, 1.
navigate_end-of-line:          0, 1.
navigate_top-of-file:                  0, 1.
navigate_end-of-file:                  0, 1.
navigate_goto-line:            0, 1.
navigate_next-page:                    0, 1.
navigate_prev-page:                    0, 1.
navigate_next-word:                    0, 1.
navigate_prev-word:                    0, 1.
mark_set:                      0, 1.
mark_cut:                      0, 1.
mark_yank:                     0, 1.
mark_yank-pop:                 0, 1.
search_forward:                0, 1.
search_reverse:                0, 1.
search_replace:                0, 1.
search_query-replace:          0, 1.

*edit-blk_navigate-blk_ratio: 1,2,3,4,5.
*hotkey_use:                  1,2,3,4,5.
*hotkey_time:                 1,2,3,4,5.
*hotkey_error:                1,2,3.
*efficiency_cursor-left:      1,2,3,4,5.
*efficiency_cursor-right:     1,2,3,4,5.
*efficiency_cursor-up:        1,2,3,4.
*efficiency_cursor-down:      1,2,3,4.
*efficiency_backspace:        1,2,3,4.
*efficiency_delete-action:    1,2,3,4,5.
```

Figure 19. Content of <task>.names file for C4.5

7.2.5. Preparation of training and testing examples

Training Sets must be stored in the data file <Task>.Data

The format is:

field_value(1), field_value(2), ..., field_value(n), output

Figure 20 lists the content of <Task>.Data file.

```
CH14, 0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0, 0,0,0,0, 0,0,0,0, 2, 3, 1, 3, 2, 2, 4, 1, 2, 5, novice
DW14, 0,1,0,0,0,0,0, 0,0,0,0,1,0,0,1,0, 0,0,1,0, 0,0,1,0, 2, 4, 2, 1, 2, 2, 4, 1, 3, 3, medium
FA14, 0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,0,0, 0,0,0,0, 2, 3, 2, 1, 2, 3, 4, 1, 1, 4, novice
GR14, 0,0,0,0,0,0,0, 0,1,0,0,0,0,0,0,0, 0,0,1,0, 0,0,0,0, 1, 4, 2, 2, 2, 2, 4, 1, 4, 3, medium
HI14, 0,1,0,0,0,0,0, 1,1,0,0,1,1,1,0,0, 0,1,1,0, 1,0,1,0, 2, 4, 2, 1, 3, 5, 4, 2, 3, 5, expert
HK24, 0,1,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,1,0, 0,0,0,0, 1, 4, 3, 1, 3, 3, 4, 2, 4, 3, medium
JN24, 0,1,0,0,0,0,0, 0,1,0,0,1,0,0,0,0, 0,0,1,0, 1,0,1,0, 2, 4, 2, 1, 2, 3, 4, 1, 4, 4, medexp
KC24, 0,1,0,0,0,0,0, 0,0,0,0,1,0,0,0,0, 0,0,1,0, 0,0,1,0, 1, 4, 2, 1, 2, 2, 4, 1, 4, 3, medexp
KV24, 0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,1,0, 0,0,0,0, 1, 4, 2, 2, 1, 1, 3, 1, 4, 3, medium
LI14, 0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,0,0, 0,0,0,0, 2, 2, 4, 1, 2, 2, 4, 1, 1, 3, novice
LS24, 1,1,0,0,0,0,0, 1,1,0,0,0,0,0,1,1, 0,0,0,0, 0,0,0,0, 2, 5, 1, 1, 4, 3, 4, 3, 4, 4, medexp
MD24, 1,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,0,0, 0,0,0,0, 2, 4, 3, 1, 2, 2, 4, 1, 2, 4, novice
MG14, 1,1,0,0,0,0,0, 0,1,0,0,0,0,1,0,0, 0,1,1,0, 1,1,1,0, 1, 4, 2, 2, 2, 2, 4, 1, 3, 4, medexp
PT14, 0,1,0,0,0,0,0, 1,0,0,0,0,0,0,0,0, 0,0,1,0, 1,0,1,0, 1, 4, 3, 1, 2, 2, 4, 1, 3, 3, medium
RB14, 0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0, 0,0,0,0, 0,0,0,0, 2, 3, 3, 1, 2, 2, 4, 1, 1, 4, novice
SC24, 1,0,0,1,1,1,1, 1,1,1,1,0,0,0,1,1, 0,1,1,0, 0,0,1,0, 2, 5, 2, 1, 5, 5, 4, 4, 4, 5, expert
WS14, 1,1,0,0,0,0,0, 0,0,0,0,0,1,0,0,0, 0,0,1,0, 0,1,0,0, 1, 4, 2, 1, 3, 3, 4, 2, 4, 4, medexp
XA24, 1,1,1,1,1,1,1, 1,1,1,0,0,0,0,1,1, 0,1,1,0, 0,0,1,0, 2, 5, 2, 1, 5, 5, 4, 4, 4, 4, expert
```

Figure 20. Content of <task>.data file for C4.5

Testing Sets must be stored in the data file <Task>.Test in exactly the same format as the <Task>.Data.

7.2.6. Results

40 patterns are available. In Test1, 18 are used as training sets and 22 are used as testing sets. In Test2, the two sets are swapped. The output screen (of Test2) is shown in Figure 21. C4.5 tries to build a decision tree by extracting information

from training data. It also tries to prune the decision tree to minimise the number of nodes (and hence the number of production rules).

```
C4.5 [release 5] decision tree generator       Mon Jun  3 23:08:03 1996
---------------------------------------

    Options:
      File stem <task1>
      Trees evaluated on unseen cases
Read 22 cases (35 attributes) from task1.data

Decision Tree:

*efficiency_backspace = 1: novice (5.0)
*efficiency_backspace = 2: novice (1.0)
*efficiency_backspace = 3: medium (1.0)
*efficiency_backspace = 4:
|   *hotkey_use = 1: medexp (0.0)
|   *hotkey_use = 2: medexp (0.0)
|   *hotkey_use = 3: medexp (0.0)
|   *hotkey_use = 4:
|   |   mark_cut = 0:
|   |   |   *efficiency_cursor-down = 1: medium (3.0)
|   |   |   *efficiency_cursor-down = 2:
|   |   |   |   *efficiency_cursor-right = 1: medium (1.0)
|   |   |   |   *efficiency_cursor-right = 2: medexp (2.0)
|   |   |   |   *efficiency_cursor-right = 3: medexp (3.0)
|   |   |   |   *efficiency_cursor-right = 4: medexp (0.0)
|   |   |   |   *efficiency_cursor-right = 5: medexp (0.0)
|   |   |   *efficiency_cursor-down = 3: medexp (0.0)
|   |   |   *efficiency_cursor-down = 4: medexp (0.0)
|   |   mark_cut = 1: expert (3.0/2.0)
|   *hotkey_use = 5: expert (3.0)


Simplified Decision Tree:

*efficiency_backspace = 1: novice (5.0/1.2)
*efficiency_backspace = 2: novice (1.0/0.8)
*efficiency_backspace = 3: medium (1.0/0.8)
*efficiency_backspace = 4:
|   mark_cut = 1: expert (5.0/3.2)
|   mark_cut = 0:
|   |   *efficiency_cursor-down = 1: medium (3.0/1.1)
|   |   *efficiency_cursor-down = 2: medexp (6.0/2.3)
|   |   *efficiency_cursor-down = 3: expert (1.0/0.8)
|   |   *efficiency_cursor-down = 4: medexp (0.0)

Tree saved

Evaluation on training data (22 items):

        Before Pruning           After Pruning
        ----------------   ---------------------------
        Size      Errors   Size      Errors   Estimate
          21    2( 9.1%)     11    3(13.6%)    (46.1%)   <<

Evaluation on test data (18 items):

        Before Pruning           After Pruning
        ----------------   ---------------------------
        Size      Errors   Size      Errors   Estimate
          21    5(27.8%)     11    5(27.8%)    (46.1%)   <<


         (a)   (b)   (c)   (d)   (e)  <-classified as
        ----  ----  ----  ----  ----
           3           3              (a): class expert
                                      (b): class medexp
                 2     5              (c): class medium
                                      (d): class novmed
                                 5    (e): class novice
```

Figure 21. Decision tree generated by C4.5

If we use the rule building facility, `c4.5rules`, to generate the rules, we get a set of rules as shown in Figure 22.

```
Rule 8:
       *hotkey_use = 5
       ->  class expert  [63.0%]

Rule 2:
       mark_cut = 0
       *hotkey_use = 4
       *efficiency_cursor-down = 1
       ->  class medium  [70.7%]

Rule 1:
       *efficiency_backspace = 1
       ->  class novice  [75.8%]

Rule 4:
       mark_cut = 0
       *efficiency_cursor-down = 2
       *efficiency_backspace = 4
       ->  class medexp  [61.2%]

Default class: medexp

Evaluation on training data (22 items):
Rule   Size   Error   Used   Wrong              Advantage
----   ----   -----   ----   -----              ---------
   8      1   37.0%      3       0 (0.0%)         3 (3|0)  expert
   2      3   29.3%      4       0 (0.0%)         4 (4|0)  medium
   1      1   24.2%      5       0 (0.0%)         5 (5|0)  novice
   4      3   38.8%      6       1 (16.7%)        0 (0|0)  medexp
Tested 22, errors 4 (18.2%)   <<

         (a)   (b)   (c)   (d)   (e)  <-classified as
        ----  ----  ----  ----  ----
           3     1                     (a): class expert
                 6                     (b): class medexp
                 1     4               (c): class medium
                 1                     (d): class novmed
                 1                5    (e): class novice

Evaluation on test data (18 items):
Rule   Size   Error   Used   Wrong              Advantage
----   ----   -----   ----   -----              ---------
   8      1   37.0%      3       0 (0.0%)         3 (3|0)  expert
   2      3   29.3%      7       2 (28.6%)        5 (5|0)  medium
   1      1   24.2%      3       0 (0.0%)         3 (3|0)  novice
   4      3   38.8%      2       2 (100.0%)       0 (0|0)  medexp
Tested 18, errors 7 (38.9%)   <<

         (a)   (b)   (c)   (d)   (e)  <-classified as
        ----  ----  ----  ----  ----
           3     2     1               (a): class expert
                                       (b): class medexp
                 2     5               (c): class medium
                                       (d): class novmed
                 1     1          3    (e): class novice
```

Figure 22. Rules extracted by C4.5

A comparison of the Classification Accuracy of Inductive System versus Neuro-Fuzzy System is shown in Table 7. The classification accuracy of the inductive system was 61.1%, compared with 79.5% of the neuro-fuzzy system.

Result Obtained	***Inductive System***	***Neuro-Fuzzy System***
Internal Test	81.8%	100%
External Test	61.1%	79.5%

Table 7. Performance of inductive system versus neuro-fuzzy System

8. DISCUSSION

8.1. KEYBOARD DYNAMICS STUDY FOR USER MODELLING AND USER IDENTIFICATION

Keyboard dynamics have been widely used in the area of computer security for user identification and intruder detection. It is interesting to compare the similarities and differences of the approaches used in user modelling in this study.

In both areas, the underlying assumption is that a user's interaction with the system has a pattern that persists for a period of time. Such patterns will be repeated by the user in the future. In the case of user identification, the focus is to identify the choice of command [Newberry & Seberry 89], or the inter-key timing of different key-pairs [Pisitkasem90] and locate the representative feature (the fingerprint) that helps identify the individual user. In user modelling of expertise level, we are looking for features that represent the user's knowledge of the system, such as the operators used, the efficiency and the error in working with them. The persistence lifetime of user's characteristics should be a little longer than their expertise level of a specific system. This is because choice of command and typing style are some kind of preference/habit of the user that is not readily changed. On the other hand, users will migrate toward "expert" as they learn more and more about the system, so this shift can be rapid. In both cases, individual differences and inconsistency exist, which usually make the classification more difficult. To deal with this problem, it is common in both areas of study to use normalisation to remove individual difference.

If the user characteristics are captured, such as the expertise level and stored in some kind of profile, temporal irregularities will not impact on the identification.

8.2. A BOTTOM-UP APPROACH IN HUMAN COMPUTER INTERACTION

In this study, we start with the operators used by a subject and map them to some possible knowledge about the user (expertise here). Similarly, we can map the operators to higher cognitive goals of the user. This represents a bottom-up approach [Eberts91], as compared with the top-down approaches in human-computer interaction research like the GOMS model [Card80] and production rules systems [Bovair90].

If we know the user's expertise level and his/her task goal, we can predict the user's action and recognise the errors made. The user can then be given the appropriate interface and help advice with which to work. Such an approach can be enhanced if the system can provide more contextual information. For example, in the study of efficiency patterns, we have made an assumption on the thresholds for word and line length, since we have no information about the actual boundary of the word and line as the user types. If the system can provide such information to a pool, the accuracy of classification could be significantly improved. In practice, such expertise level and task goal identifiers could be two agents that cooperate with the system module. Such architecture is becoming more the trend [Maes94]. The identification of expertise level is treated in this study. The identification of text-editing goal using neural networks has been studied in [Villegas94]. In both Villegas's and the present study, the operators are represented in a higher level form rather than by low level keystrokes. By doing so, the operators are more generalised and can be readily

mapped to other system. The studied system (Jove) includes only command-line and keyboard actions. However the same technique can be applied to modelling users for *multi-modal* interactions, for example, mouse actions. An expertise model agent can work with a task model agent on such information provided by the system to give advice to the user and predict their action. Lastly, we note that grouping operators into navigation and editing operators provides considerable insights.

8.3. GENERAL COMMENTS ON FUZZY LOGIC REPRESENTATION, THE PRODUCTION RULE, THE INDUCTIVE AND THE NEURAL NETWORK APPROACHES

8.3.1. Fuzzy logic representation

From this study, we can see that fuzzy logic has given us both a better representation of a continuum of expertise level and better classification accuracy. Fuzzy expert systems allow firing of multiple rules for a vague concept that has a different membership in various ranges. Furthermore, it also better accounts for expertise migration of a user.

8.3.2. Production rule approach

The advantage of the production rule approach (expert system) is that the relationships are represented by explicit rules that are easy to understand.

The disadvantages of the production rule approach in user modelling are, firstly that it involves more subjective determination of the contribution for each input. Such determination is difficult and unreliable. A related problem is conflict resolution – when more than one rule is matched, which rule has priority in firing? The problem

becomes more severe when the complexity of the problem grows -- more contribution factors of variables need to be determined. Secondly, it is less accurate (52.5% correct identification, compared with the neural network approach -- 79.5%). Thirdly, the rules are difficult to modify. Fourthly, the response time is long – the search goes through the rules one by one and fires matched ones until no more rules need to be fired.

8.3.3. Inductive approach

The inductive approach also has explicit rules. It is easy to understand as with the production rule approach, but has the advantage that the rules are extracted from the examples *directly*. It is fast and poses no problem in assigning credits to different attributes. It has a higher accuracy than the production rule approach.

However it also has its downside. Firstly, the extracted rules may not be meaningful. Secondly, it still has difficulty in dealing with complex problems. The computation complexity grows tremendously when the number of variables grows. Thirdly, it cannot achieve a good accuracy in internal test and has lower classification accuracy in external test, when compared with the neural network approach (on average 61.1% correct identification, compared with the neural network's 79.5%).

8.3.4. Neural network approach

It has been verified that the neural network approach has some benefits over the production rule and inductive approaches: it learns automatically (in contrast with the production rule approach), and with better learning ability than the inductive approach. It has a finer granularity of classification and better classification in coarse and fine-grained classification. It is more flexible to more complex problems and

thus is more scalable. Once trained, the neural network responds very fast. This is important in an adaptive interface interacting with the user in real-time. The neural network is capable of handling incomplete and noisy inputs and guarantees an output.

The downside of the neural network approach is that there are no explicit rules that specify the relationships of inputs and output. For the specific neural network architecture in this study (MLP), adding new features requires *re*training of the whole network.

8.4. CREDIT ASSIGNMENT PROBLEM

The 924credit assignment problem is the problem of assigning *credits* or *blames* for overall outcomes to each of the internal decisions made by a learning system and which contributed to that outcome [Minsky61]. In the task of user expertise modelling we are facing such a problem, namely how to assign contributions to each of the inputs (such as operators and efficiency patterns) of the system.

This problem leads to an inability to define rules for all components in the fuzzy expert system. The more components involved, the more complicated the relationships are, and the more difficult it is to write the rules. Such a problem manifests in all learning systems that cannot learn automatically.

As inductive system learns automatically by extracting rules from training examples, it can identify the decisive factors from among the multiple inputs. However, as we have seen in this study, an inductive system tends to neglect less important factors,

which may be significant in certain cases. The rules generated by an inductive system can only give a coarse assignment of credits and blames. That is to say, explicit symbolic rules, which is the beauty of expert and inductive systems, are not an effective mechanism to deal with credit assignment problem for multi-dimensional inputs.

A system, that can learn automatically and can fine tune assignment of credits to multiple inputs, is desirable for the user modelling task. The multi-layered perceptron used in this study has such a capability. Firstly, its learning is automatic. The weighting of each input's contribution to the output is adjusted automatically by the learning algorithm according to the back-propagated error and stored in the links. Secondly, the hidden neurons in the MLP architecture provide a place to integrate contributions from all inputs. Both the input's activation and the weights are represented as *numeric* values, so its contribution is readily combined with other inputs by summation at the neuron to generate a total contribution (Figure 24). The transfer function at the hidden neuron controls the threshold level at which the output is fired.

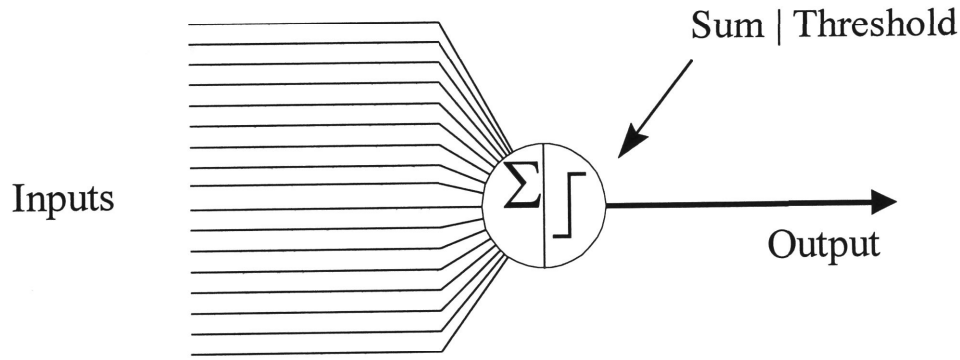


Figure 24. A hidden neuron in multi-layered perception

We can write it in the similar form of a rule:

```
IF ( contribution of input-1
       + contribution of input-2
       + …
       + contribution of input-N )

       is greater than THRESHOLD

THEN
       fire output.
```

The hidden neuron can be seen as a powerful representation of the rule. It is a simple and effective mechanism and here accounts for the better performance of the MLP over both the fuzzy expert and inductive systems.

9. CONCLUSION AND FUTURE WORK

9.1. CONTRIBUTION OF THE STUDY

We have verified that it is possible to use an artificial neural network approach to tackle the user modelling problem. Our multi-layered perceptron model is different from [Beale89]'s ADAM, but still employs a supervised learning paradigm. Another similarity is that Beale's and our approach track subjects' command use, help requests and errors. The 3-level classification accuracy we have obtained (80%) is better than Beale's 2-level classification (71.2%) and the 5-level classification accuracy we have obtained approximates that of 3-level (79.5% in correct class, and 100% within adjacent classes).

Unlike Beale's study, where data was abstracted from 3-month usage traces of the system, we extracted information from a small but substantial task designed specifically for the purpose of this study. It can be seen that a small but substantial task is good enough to give a picture of the user's expertise in using the system. However, a longer sampling time definitely gives an even better result in terms of accuracy and granularity. Some inconsistency was observed in subjects' behaviour due to emotional or some other reason. An expert user may fail to act effectively and efficiently sometimes. Chiu pointed out that such inconsistency in user behaviour is part of the nature of the user model [Chiu91]. It should be noted that the accuracy of MLP-4 in 3-level classification (80%) is close to that obtained with 5-level classification (79.5%). This means that there are a number of subjects who fall

midway between the medium user and expert user continuum. In other words, their behaviour "oscillates" within an interval of expertise, thus echoing the uncertainty of the user model. A longer sampling period can help to isolate irregularities and concentrate on the most representative behaviours. In this study, we do not observe subjects using very many help and command mode functions. If sampling time was made larger, this information would become available and could be used to classify subjects better and with finer granularity. However, we must point out that users can learn by doing. Thus there exists an indefinite migration of expertise among users that needs to be taken into consideration in this type of research. However, the longer the sampling period, the more noise is introduced.

The features we track in the final network setup MLP-4 include 24 individual hotkeys, 6 efficiency patterns, the overall hotkey usage, overall timing, overall error and the subject's planning ability. The hotkey timing cannot be tracked because of noise introduced by the keyboard. The variety of features that we track is more extensive than Beale's. It is worth mentioning that inclusion of efficiency patterns produces a better accuracy and results in a finer granularity of expertise level (Beale's work did not include efficiency patterns).

Our grain size in the division of expertise is 5-levels compared with Beale's 2-levels, which is achieved using a fuzzy representation of output (expertise level). Such representation gives better convergence, higher accuracy and makes better use of the 3 output nodes. Besides it also better accounts for the migration of expertise level.

The success of MLP-4 can be attributed to the importance of preprocessing the data, choice of data representation and selection of network architecture. Preprocessing selects significant features and hides unnecessary details. The representation of level

features in ratio and proportion greatly reduces the input dimension and keeps the network to a manageable size. The coding of the level features in contiguous 1's greatly improves the clustering of input patterns when presented to Kohonen's self-organised map.

9.2. BEST LEARNING ALGORITHM AND LEARNING PARAMETERS

The best learning algorithm found was RPROP with the following parameters in SNNS [Zell95] :

$delta_0$:	0.1 (default),
$delta_{max}$:	50.0 (default), and
α:	6 (corresponds to a ratio of weight decay term to output error of $1:10^6$)

9.3. IMPROVEMENTS THROUGH PROGRESSIVE REFINEMENTS

The accuracy of classification using the neuro-fuzzy system could be improved through the following progressive refinements:

(1) representing expertise level in fuzzy form,

(2) tracking individual operators/hotkeys,

(3) tracking efficiency pattern,

(4) using a new coding method, and

(5) refining the network architecture.

We have achieved improvements in classification accuracy, as indicated in Table 8.

		MLP1	*MLP2*	*MLP3*		*MLP4*	
		Initial	*Fuzzy O/P, Tracking Hotkeys*	*Tracking Efficiency Patterns*	*New Coding Method*	*Enhanced Architecture*	*Training Set Balancing*
Test 1	*Correctly identified*	did not	59%	70%	77%	82%	82%
	within adjacent classes	converge	95%	100%	95%	95%	100%
Test 2	*correctly identified*	did not	67%	77%	77%	72%	77%
	within adjacent classes	converge	95%	95%	95%	100%	100%
Average	*correctly identified*	did not	62%	74%	77%	77%	79.5%
	within adjacent classes	converge	95%	97.5%	95%	97.5%	100%

Table 9. Improvements in classification accuracy through progressive refinements

If we use MLP4 to classify users into 3 expertise levels, the classification accuracy is 80%.

9.4. INTELLIGENT APPROACHES TO DYNAMIC USER MODELLING

In this study we have benchmarked the performance of the neuro-fuzzy system against both fuzzy expert and inductive systems. A comparison of these results is shown in Table 9.

		Neuro-Fuzzy System	***Fuzzy Expert System***	***Inductive System***
Internal Test		100%	-----	81.8%
External Test	*correctly identified*	79.5%	52.5%	61.1%
	within adjacent classes	97.5%	95.0%	-----

Table 8. Performance of three different intelligent approaches to dynamic user modelling

9.4.1. Classification accuracy of automatic learning versus manual learning

We can see that automatic learning methods (inductive system and neuro-fuzzy system) yield better results compared with the fuzzy expert system in identifying user in the correct class. This is probably due to inability to write explicit rules for a

system that we do not possess enough knowledge of the antecedent-consequence pairs in a multi-dimensional input system like user modelling.

Another point worth to note is the fuzzy production rule system performs very well within adjacent classes (95% accuracy). This is very impressive. If we accept classification error within adjacent classes, the fuzzy production rule system is better, since it is more economical. Two questions then arise:

(1) Can we further improve the accuracy of the production rule system so that it approximates or outperforms the neuro-fuzzy system?

(2) Is the Fuzzy Production Rule System superior to the ANN approach?

To answer question (1), we can say that, of course, we can still further fine-tune the production rule system by adjusting the weighting of each operator or including more input classes. However, tuning the weighting of each operator still relies on information from the neural network or from the inductive system. That is to say, do we believe that a manually adjusted system can outperform an automatic learning system, provided that the neural network or the inductive system has been trained by representative exemplars? Furthermore, how can we write more complicated rules with more input classes? Even if the answers to these questions are positive, we still need to bear in mind that the production rule system is not very flexible. For example, modification of production rules is tedious, whereas retraining the neural network is quite straight-forward.

To answer question (2), we must bear in mind that the Fuzzy Production Rule System here has been empowered by a proper choice of inputs and weight adjustments and this knowledge indeed comes from the empirical findings of the fuzzy neural network. It is hard to acquire such knowledge from other reliable

sources. The most fundamental question is, "how do we acquire knowledge for a system that we have little knowledge, or a system that has too many varying attributes?" User modelling is an example of such a system. The "superior" fuzzy production rule system here has not answered this question. Its superb performance relies on another artificial intelligence mechanism (here neural network) to perform knowledge acquisition. Strictly speaking it has been transformed into a hybrid system.

This superior hybrid system would have these features:

- is production rule based
- has an automatic learning module (be it a MLP neural network or an inductive system) for knowledge acquisition
- represents input in fuzzy form and incorporates a fuzzy inference engine

9.4.2. Learning ability of the two automatic learning paradigms

There are two automatic learning paradigms discussed in this study: the neuro-fuzzy (neural network with fuzzy outputs) and inductive approaches. In the neural network, the learnt input-output relationships are stored as *weights* of the synapse linking the nodes while in inductive system they are derived as *explicit rules* that can be used by the expert system. The internal test classification accuracy tells us that the neuro-fuzzy system (100%) learns better than the inductive system (81.8%). Such difference in learning ability is also reflected in the external test classification accuracy (neural network system's 79.5% versus inductive system's 61.1%).

9.4.3. Granularity of classification

Although the Fuzzy Expert System is less accurate in correctly identifying the subject's exact class (52.5%), it has a good classification within adjacent classes (95%) which is comparable with the neuro-fuzzy system (97.5%). This suggests that the fuzzy expert system is fairly powerful with coarse classification. The neuro-fuzzy system, however, is more accurate even with finer classification.

9.4.4. How well can a fuzzy expert system perform?

The inductive system extracts the rules from training examples. The rules extracted should be independent of error caused by human judgement. If the training examples in the study are sufficiently representative, the classification accuracy of 61.1% should provide a hint to the upper bound of classification accuracy that a rule-based system can achieve (at present it is 52.5%).

9.4.5. Agreement of the extracted rules in the inductive system with the other systems

The inductive system searches for the underlying structure in the training examples that provides a basis for hypothesising the relationships between the variables governing the process. Hence the decision tree can give us a picture of the rule structure hierarchy to the problem. This "visible" structure is useful in checking the validity of the assumptions we have made in developing the neuro-fuzzy and the fuzzy expert systems. Let us look at the decision tree generated by C4.5 from the training examples in Figure 23.

```
*efficiency_backspace = 1: novice (5.0)
*efficiency_backspace = 2: novice (1.0)
*efficiency_backspace = 3: medium (1.0)
*efficiency_backspace = 4:
|   *hotkey_use = 1: medexp (0.0)
|   *hotkey_use = 2: medexp (0.0)
|   *hotkey_use = 3: medexp (0.0)
|   *hotkey_use = 4:
|   |   mark_cut = 0:
|   |   |   *efficiency_cursor-down = 1: medium (3.0)
|   |   |   *efficiency_cursor-down = 2:
|   |   |   |   *efficiency_cursor-right = 1: medium (1.0)
|   |   |   |   *efficiency_cursor-right = 2: medexp (2.0)
|   |   |   |   *efficiency_cursor-right = 3: medexp (3.0)
|   |   |   |   *efficiency_cursor-right = 4: medexp (0.0)
|   |   |   |   *efficiency_cursor-right = 5: medexp (0.0)
|   |   |   *efficiency_cursor-down = 3: medexp (0.0)
|   |   |   *efficiency_cursor-down = 4: medexp (0.0)
|   |   mark_cut = 1: expert (3.0/2.0)
|   *hotkey_use = 5: expert (3.0)
```

Figure . A portion of the decision tree extracted by C4.5

The decision tree suggests that the efficiency of using [backspace] is the primary classifier. A low efficiency in using [backspace] (*efficiency_backspace = 1 or 2) indicates a Novice expertise level; a medium efficiency (*efficiency_backspace = 3) indicates a Medium expertise level; a high efficiency (*efficiency_backspace = 4) indicates a range of expertise level: from Medium to Expert, depending on whether the subject uses a lot of hotkeys (the secondary classifier *hotkey_use). If he/she uses few hotkeys (*hotkey_use = 1, 2 or 3) then he/she is a MedExp. On the other hand, if he/she uses a lot of hotkeys (*hotkey_use = 5) then he/she is an Expert. If he/she uses a reasonable number of hotkeys (*hotkey_use = 4) then we need to take into account whether the subject knows the "mark_cut" operator or not. It the subject know this operator (mark_cut=1), then he/she is an expert, else we need to look at other factors (the *efficiency_cursor_down and the *efficiency_cursor_right). We can see that (1) the higher efficiencies in using operators, (2) a more intense use of hotkeys, and (3) the knowledge of using certain

more advanced operators all imply a higher level of expertise in using the Jove editor. This agrees with the assumption made in the refinement of the neural network system; namely that individual operators and some efficiency patterns are important in pattern recognition.

The level of efficiency patterns indicates that a lower efficiency maps a level nearer to Novice level and vice versa. This agrees with the production rules of the fuzzy expert system.

9.5. FUTURE WORKS

In this study, it has been shown that using MLPs and data abstracted from a small but substantial document can help identify the expertise level of a user. Accordingly, future research should follow up the following questions:

(1) how can the accuracy can be improved – by a longer and/or more substantial text?

(2) can a context information provider that sits inside the system significantly improve the performance?

(3) what kind of network architecture best suits the problem?

(4) how can we extend such research to a system of multi-modal inputs?

Follow-up research should clarify the above questions, thus enhancing and generalising our preliminary findings.

REFERENCES

[Beale89] Beale, R., Finlay, J., Austin, J. & Harrison, M. (1989). "User Modelling by Classification: a Neural-based Approach." In New Developments in Neural Computing. J.G. Taylor and C.L.T. Mannion(Eds.) Bristol, UK and New York: A. Hilger.

[Beale90] Beale, R. & Jackson, T. (1990). Neural Computing: An Introduction. London: Institute of Physics Publishing.

[Belage92] Belage, T. & Spenke, M (1992) "The GINA Interaction Recorder." Engineering for Human-Computer Interaction. J. Lason and C. Unger (Eds.) The Netherlands: Elsevier Science Publishes B.V.

[Bovair90] Bovair, S., Kieras, D.E. and Polson, p.G. (1990) "The Acquisition and Performance of Text-editing Skill: A Cognitive Complexity Analysis." Human Computer Interaction, Vol.5, pp.1-48.

[Brown93] Brown, M. and Rogers, S.J. (1993) "User Identification via Keystroke Characteristics of Typed Names using Neural Networks." International Journal of Man-Machine Studies Vol. 39, pp. 999-1014.

[Card80] Card, S.K., Moran, T.P. and Newell, A. (1980) "The Keystroke-Level Model for User Performance Time with Interactive Systems". Communications of the ACM Vol. 23, No.7, pp.396-410.

[Chen91] Chen, Q. and Norcio, A.F. (1991) "A Neural Network Approach for User Modelling." Proceedings of the IEEE 1991 International Conference on Systems, Man and Cybernetics, Charlottesville, VA, USA, Vol.2, pp. 1429-1434.

[Chen92] Chen, Q. and Norcio, A.F. (1992) "Modelling Users with Neural Architectures". Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD, USA, Vol. 1, pp.I-547-551.

[Chen94] Chen, Q. and Norcio, A.F. (1994) "Stereotyping Users and Tasks with Associative Memories". Proceedings of the 1994 IEEE International Conference on Neural Networks, Orlando, FL, USA, Vol. 2, pp. I-1169-74.

[Cheung90] Cheung, K.M., Lustig, I. And Kornhauser, A.L. (1990) "Relative Effectiveness of Training Set Patterns for Back Propagation" Proceedings of the International Joint Conference on Neural Networks, San Diego, CA, USA, Vol.1, pp.673-8.

[Chiu91] Chiu, C., Norcio, A.F. & Petrucci, K.E. (1991) "Using Neural Networks and Expert Systems to Model Users in an Object-Oriented Environment". Proceedings of the IEEE 1991 International Conference on System, Man and Cybernetics, Charlottesville, VA, USA, Vol.3, pp. 1943-1948.

[Coutaz92] Coutaz, J. (1992) "Critical Issues: User Modelling" in Engineering for Human-Computer Interaction. J.Lason and C. Unger (Eds.) The Netherlands: Elsevier Science Publishes B.V.

[Deng93] Deng, P.S. (1993) "Automating Knowledge Acquisition and Refinement for Decision Support: A Connectionist Inductive Inference Model". Decision Science Vol.24, No.2, Mar/Apr 1993, pp. 371-393.

[Domino] Domino. Data Display Terminal Operating Manual model/DVT-227+ Domino Computers Pty Ltd, Australia.

[Durkin94] Durkin, J. (1994) Expert System Design and Development. New York: Macmillan Publishing Company.

[Eberts91] Eberts, R. (1991) "Knowledge Acquisition Using Neural Networks for Intelligent Interface Design" Proceedings of the 1991 IEEE International Conference on Systems, Man and Cybernetics, Charlottesville, VA, USA Vol.2, pp.1331-5.

[Ferret93] Ferret, E. (1993) "Improving the Neural Network Testing Process". In Adaptive Intelligent Systems. Society for Worldwide Interbank Financial & Telecomm (Ed.) Amsterdam and New York: Elsevier Science Publishes B.V.

[Finin83] Finin, T.W. (1983) "Providing Help and Advice in Task Oriented System". Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, Vol. 1, p. 176-178.

[Giarratano93] Giarratano, J.C. (1993) CLIPS Version 6.0 User Guide, Software Technology Branch, Lyndon B. Johnson Space Centre, NASA, USA.

[Lane93] Lane, Napie, Batsell and Naman (1993) "Keystroke-Level Model". Human Computer Interaction Vol.8, No.2, pp.186-92.

[Maes94] Maes, P. (1994) "Agents that Reduce Work and Information Overload" Communications of the ACM Vol. 37, No.7, pp.811-821.

[Martin94] Martin, F. (1994) The Mini Board 2.0 Technical Reference, Massachusetts Institute of Technology, CA, USA.

[Minsky61] Minsky, M.L. (1961) "Steps towards Artificial Intelligence". Proceedings IRE, Vol.49, pp.8-30.

[Newberry91] Newberry, M. (1991) "Active Intruder Detection: Some Aspects of Computer Security and User Authentication", Ph.D. thesis, University of New South Wales, Australia.

[Newberry & Seberry89] Newberry, M. and Seberry, J. (1989) "User Unique Identification", Proceedings of the 12th Australian Computer Science Conference, Australian Computer Science Communications, Vol.11, No.1, pp. 163-71. University of Wollongong.

[Norcio91] Norcio, A.F. (1991) "Adaptive Interfaces: Modelling Tasks and Users". Proceedings of the IEEE 1991 International Conference on Systems, Man and Cybernetics, Charlottesville, VA, USA, Vol.2, pp. 1099-1103.

[NRC94] NRC (1994) FuzzyCLIPS Version 6.02A User Guide. Knowledge Systems Laboratory, Institute for Information Technology, National Research Council, Canada

[Payne] Payne, Jove Manual for Unix Users.
WWW fftp://ftp.cs.toronto.edu/pub/moraes/jove/jove.ps

[Pisitkasem90] Pisitkasem, S. (1990) "A Study of Keyboard Dynamics". Preprint No. 90/03. Department of Computer Science, University of Wollongong.

[Quinlan93] Quinlan, J.R. (1993) C4.5: Programs for Machine Learning. Morgan Kauffman,

[Rich79] Rich, E. (1979) "User modelling via stereotypes". Cognitive Science Vol. 3, pp. 329-354.

[Rissland84] Rissland, E.L.(1984) "Ingredients of Intelligent User Interfaces". International Journal of Man-Machine Studies Vol. 21, No.4, pp. 377-388.

[Stanfill86] Stanfill, C. & Waltz, D. (1986) "Toward memory-based reasoning". Communications of the ACM Vol. 29, No.12, pp.1213-1228.

[Tyler91] Tyler, S.W., Schlossberg, J.L., Gargan, R.A., Cook, L.K. & Sullivan, J.W. (1991) "An Intelligent Interface Architecture for Adaptive Interaction" in Intelligent User Interface. James W. S. (Ed.). New York: ACM Press; Massachusetts: Addison-Wesley .

[Villegas94] Villegas, L. and Eberts, R.E. (1994) "A neural network tool for identifying text-editing goals". International Journal of Man-Machine Studies Vol. 40, pp. 813-833.

[Zadeh89] Zadeh, L.A., (1989) "Knowledge Representation in Fuzzy Logic" IEEE Transactions on Knowledge and Data Engineering Vol. 1, Issue 1, pp.89-100.

[Zell95] Zell, A. et.al. (1995) Stuttgart Neural Network Simulator User Manual Version 4.1, Institute for Parallel and Distributed High Performance Systems, University of Stuttgart, Germany.

APPENDICES

APPENDIX A: JOVE EDITOR COMMANDS

add-lisp-special
append-region
apropos
auto-execute-command
auto-execute-macro
auto-fill-mode
auto-indent-mode
backward-character
backward-list
backward-paragraph
backward-s-expression
backward-sentence
backward-up-list
backward-word
begin-kbd-macro
beginning-of-file
beginning-of-line
beginning-of-window
bind-keymap-to-key
bind-macro-to-key
bind-macro-to-word-abbrev
bind-to-key
buffer-position
c-mode
case-character-capitalize
case-region-lower
case-region-upper
case-word-capitalize
case-word-lower
case-word-upper
cd
character-to-octal-insert
clear-and-redraw
compile-it
continue-process
copy-region
current-error
date
define-global-word-abbrev
define-macro
define-mode-word-abbrev
delete-blank-lines
delete-buffer
delete-current-window
delete-next-character
delete-other-windows
delete-previous-character
delete-white-space
describe-bindings
describe-command
describe-key
describe-variable
digit
digit-0
digit-1
digit-2
digit-3
digit-4
digit-5
digit-6
digit-7
digit-8
digit-9
digit-minus
dirs
down-list
dstop-process
edit-word-abbrevs
end-kbd-macro
end-of-file
end-of-line
end-of-window
eof-process
erase-buffer
exchange-point-and-mark
execute-kbd-macro
execute-macro
execute-named-command
exit-Jove
fill-comment
fill-paragraph
fill-region
filter-region
find-file
find-tag
find-tag-at-point
first-non-blank
forward-character
forward-list
forward-paragraph
forward-s-expression
forward-sentence
forward-word
fundamental-mode
gather-numeric-argument
goto-line

goto-window-with-buffer
grind-s-expr
grow-window
handle-tab
i-search-forward
i-search-reverse
i-shell-command
insert-file
interrupt-process
kill-next-word
kill-previous-word
kill-process
kill-region
kill-s-expression
kill-some-buffers
kill-to-beginning-of-sentence
kill-to-end-of-line
kill-to-end-of-sentence
left-margin-here
lisp-mode
list-buffers
list-processes
local-bind-keymap-to-key
local-bind-macro-to-key
local-bind-to-key
make-buffer-unmodified
make-macro-interactive
name-kbd-macro
newline
newline-and-backup
newline-and-indent
next-error
next-line
next-page
next-window
number-lines-in-window
over-write-mode
page-next-window
paren-flash
parse-errors
parse-spelling-errors-in-buffer
pause-Jove
pop-mark
popd
previous-error
previous-line
previous-page
previous-window
print
process-bind-keymap-to-key
process-bind-macro-to-key
process-bind-to-key
process-dbx-output
process-newline
process-send-data-no-return
push-shell
pushd
pushlibd
pwd
query-replace-string
quit-process
quoted-insert
read-only-mode
read-word-abbrev-file
recursive-edit
redraw-display
rename-buffer
replace-in-region
replace-string
right-margin-here
save-file
scroll-down
scroll-left
scroll-right
scroll-up
search-forward
search-forward-nd
search-reverse
search-reverse-nd
select-buffer
select-buffer-1
select-buffer-10
select-buffer-2
select-buffer-3
select-buffer-4
select-buffer-5
select-buffer-6
select-buffer-7
select-buffer-8
select-buffer-9
self-insert
set
set-mark
shell
shell-command
shell-command-no-buffer
shell-command-to-buffer
shell-command-with-typeout
shift-region-left
shift-region-right
show-match-mode
shrink-window
source
spell-buffer
split-current-window
start-remembering
stop-process
stop-remembering
string-length
suspend-Jove
text-mode
transpose-characters
transpose-lines
unbound

version
visible-spaces-in-window
visit-file
window-find
word-abbrev-mode
write-file
write-macros-to-file
write-modified-files
write-region
write-word-abbrev-file
xj-mouse-copy-cut
xj-mouse-line
xj-mouse-mark
xj-mouse-point
xj-mouse-word
xj-mouse-yank
xt-mouse-mark
xt-mouse-point
xt-mouse-up
yank
yank-pop

APPENDIX B: JOVE EDITOR HOTKEYS FOR VT100

Up Down Left Right		PF1	PF2	PF3	PF4
(as expected)		Left WORD	Right WORD	Beg of LINE	End of LINE
	with ESC --->	Split WINDOW	Switch WINDOW	Kill Other W	Enlarge WINDOW
Numeric pad:		**7** YANK	**8** COPY Region	**9** KILL Region	**-** Page DOWN
		4 SPELL	**5** FILL Para	**6** Fill-m Toggle	**,** Page UP
		1 List BUFFERs	**2** Select BUFFER	**3** Find FILE	**ENTER** | | SHELL
		--- 0 --- COMMAND		**.** OvrWT Toggle	| |
Numeric pad,	***with ESC --->***	**7 (IC)** Set MARK	**----**	**9 (DC)** Exch . & m	**-----**
		----	**----**	**----**	**-----**
		1 (IL) Insert LINE	**----**	**3 (DL)** Delete LINE	| | |
		---------		**-----**	|

LIST OF KEY BINDINGS (ones in parentheses are already bound to keys).

	CTRL	^X & CTRL	^X & LETTER	ESC & LETTER	ESC & CTRL
A	(begin-line)			back-sent	
B	(back-char)	(list-buffers)	(select-buffer)	(back-word)	back-s-expression
C		exit-Jove		case-capitalize	
D	delt-next-char		delt-curr-wind	kill-next-word	down-list
E	(end-line)	compile-it	exec-kbd-macro	forw-sent	
F	(forw-char)	(find-file)		(forw-word)	forw-s-expression
G	ABORT COMMAND			goto-line	
H	(delt-prev-char)				
I	(handle-tab)	insert-file		macro-interacti	
J	(newline-&-inden)			(fill-paragraph)	
K	kill-end-line		delt-buffer	kill-end-sent	kill-s-expression
L	redraw-display			case-lower	clear-&-redraw
M	(newline)	write-mod-files		first-non-blank	
N	(next-line)	next-error	(next-wind)		forw-list
O	(newline-&-back)	delt-blank-lin	(prev-wind)		
P	(prev-line)	prev-error	(prev-wind)		back-list
Q	quoted-insert			query-repl-str	
R	search-reverse	visit-file		repl-string	

```
S  search-forw     save-file       save-file       pause-Jove
T  transp-chars    transp-lines    find-tag
U  *4-numeric-arg                                  case-upper      back-up-list
V (next-page)      visit-file                     (prev-page)     page-next-wind
W (kill-region)    write-file                     (copy-region)
X                 (exch-.-&-mark)                 (exec-command)
Y (yank)                                           yank-pop
Z  scroll-up                                       scroll-down
```

CTRL		**ESC & CHARACTER**	
^\	search-forw	ESC ,	begin-wind
^~	quoted-insert	ESC -	digit-minus
^@	(set-mark)	ESC .	end-wind
		ESC [0-9]	digit
	^X & CTRL	ESC <	begin-file
		ESC >	end-file
^X ^\	save-file	ESC ?	describe-command
		ESC \	delt-white-space
	^X & CHARACTER	ESC]	forw-paragraph
		ESC ~	make-buffer-unmod
^X !	shell-command	ESC DEL	kill-prev-word
^X (	begin-kbd-macro	DEL	delt-prev-char
^X)	end-kbd-macro		
^X 1	(delt-other-winds)		
^X 2	(split-curr-wind)		
^X 4	wind-find		
^X ?	describe-key		
^X ^	(grow-wind) ^X DEL	kill-begin-sent	

In addition to the bindings shown, the following ESC combinations also work:

```
        ESC L9          search-reverse
        ESC DEL         kill-previous-word
```

APPENDIX C: SPECIFICATION OF MICROCOMPUTER

CATEGORY	*DESCRIPTION*	*DATA*
Microcomputer	68HC811E2FN	256 bytes RAM
	(Motorola 6811 series)	2048 bytes EEPROM
External Memory	none	-----
Digital I/O	bi-directional I/O	8 lines
Timer Pot	programmable timers	4 / 5 inputs
	hardware counters	3 / 4 inputs
Power	supply voltage	5.6 - 36 volts
	current drawn	80mA
Communications	1 x RS232	1 x RJ11 jack
	1 x Motorola serial interface	2 x RJ11 jacks

APPENDIX D: PROGRAM LOADED IN MICROCOMPUTER

```
/*-------------------------------------------------------------------------
 PROGRAM keysnoop.c
 -------------------------------------------------------------------------
 This program pools the terminal line every millisecond for character and
  copies the character to the output port with 2 timing information bytes (8-bits)
  plus a linefeed.

  The counter is incremented each pool until it reaches FFFF(hex) where it stays.
 The counter is reset to zero after each transmission.

  written by: Michael Milway 1995
-------------------------------------------------------------------------*/
#include <68hc11.h>
#include <intrpt.h>
#include "..\io.h"
#define SS 0x80

/* prototypes */
void sysinit(void);
void puts0(char *);
char getc0(void);
void putc0(char);
void putdigit(char);
void putword(int);

/* Global variables */
near volatile unsigned int time; /* System time */
near volatile unsigned int interval; /* time since last keypress */
const char digit[16] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

void main()
{
char ch;
unsigned int gap;
   sysinit(); /* set up serial I/O */
   puts0("\r\nKeysnoop v1.1.4/12/1995.\n\r");
   for (;;)
   {
            ch = getc0();
            gap = interval;
            interval = 0;
            putc0(ch);              /* echo the character */
            /* putc0(':'); */
            /* putword(gap); */
            putc0(gap>>8);
            putc0(gap);                                    /* send time as 2 8bit binary numbers */
            putc0('\n');
            }
}
/* initialise system requirements */
void sysinit(void)
{
/* Initialise timer varaibles */
   time = 0;
   interval = 0xffff; /* longest possible interval */
/* Initialise SCI to 9600 baud */
   BAUD = 0x30; /* 9600 baud */
   SCCR1 = 0x00; /* 8 bit, no parity */
   SCCR2 = 0x0c; /* Tx, Rx enabled, no interrupts */
   PACTL = 0x80; /* DTR output bit */
   PORTA = 0x00; /* DTR = 0 = asserted */
/* Initiasise timer 4 for 1msec interrupts */
            TCTL1 = 0x04;  /* Toggle OC4 pin on interrupt, toggles at 500Khz */
            TFLG1 = 0x10;  /* Enable TO4 interrupt */
            TMSK1 = 0x10;  /* Clear flag */

            ei(); /* Enable processor interrupts */
}
```

```
/* Send a string to the 6811 SCI */
void puts0(s)
char *s;
{
   while(*s)
            putc0(*s++);
}

/* Send a char to the 6811 SCI */
void putc0(x)
char x;
{
   while(!(SCSR & 0x80)); /* wait for TDRE set */
   SCDR = x;
}

/* get a character from the 6811 SCI */
char getc0(void)
{
   while(!(SCSR & 0x20)); /* wait for RDRF set */
   return (SCDR);
}

/* put a hex digit */
void putdigit(c)
char c;
{
            putc0(digit[c&0x0f]);
}

/* put a 16 bit word as a 4 digit hex number */
/* used by old version of program only */
void putword(x)
int x;
{
            putdigit(x>>12);
            putdigit(x>>8);
            putdigit(x>>4);
            putdigit(x);
}

/* Interrupt handler for real time 1ms interrupt */
void interrupt timer(void)
{
ROM_VECTOR(TOC4_VEC, timer);
   TOC4 += 2000; /* Bump timer compare value by 2000 cycles = 1ms */
   TFLG1 = 0x10;  /* Clear the interrupt flag */
   time++;       /* Increment system time */
   if (interval != 0xffff)
                        interval++;    /* Increment current interval count */
}
```

APPENDIX E: UNIX PROGRAM TAPPING THE PORT

```
/*-------------------------------------------------------------------------
  PROGRAM startlog.c
 -------------------------------------------------------------------------
 This program pools port 7023 of csci-ts1 server. It accepts input data of
 format:  [char] [2-bit timing information] [linefeed]
 from the microcomputer monitor and copies to a log file converted data of format:
                [char] : [4-bit hexadecimal timing information] [linfeed]

 History:  mostly ripped off from Peter Gray's annex.c
           modified by SC Leung
-------------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <signal.h>
#include <string.h>

#define BARF                    1
#define OK                      0
#define LOGFILE     "log"
#define SIG_WAIT    "sig_wait"
#define SIG_GO      "sig_go"
#define SIG_STOP    "sig_stop"
#define SIG_CLOSED          "sig_closed"

#define RECORD_SIZE         4
#define BUFFER_SIZE         10

int
main(int argc, char **argv)
{
        char        *annex = "csci-ts3"; /* deliberately hard coded */
        int         port = 7023;                    /* really */
        struct      sockaddr_in sin;
        struct      hostent *hp;
        int         c, i, s=0, count=0, insync=1, debug=0;
unsigned char       buf[BUFFER_SIZE], buf0;
        FILE        *fp, *fp_signal;
extern  char        *optarg;
extern  int         optind, optopt;
        if (argc > 1) debug=1;
        for (i=0;;i++)
        {
                fprintf(stderr,"Attempt %d\n",i);
                if (i > 20)
                {
                        fprintf(stderr, "Unable to connect after 20 attempts.\n");
                        fp_signal=fopen(SIG_WAIT,"w");
                        fclose(fp_signal);
                        exit(BARF);
                }
                memset((char *) &sin, 0, sizeof(sin));
                if (s > 0) close(s);
                s = socket(AF_INET, SOCK_STREAM, 0);
                if (s < 0)
                {
                        fprintf(stderr,"Unable to get socket, abort.\n");
                        exit(BARF);
                }
                hp = gethostbyname(annex);
                if (hp == NULL)
                {
                        fprintf(stderr,"Unable to determine network address of ");
                        fprintf(stderr,"%s\n", annex);
                        exit(BARF);
```

```
                }
                sin.sin_family = hp->h_addrtype;
                memcpy((char *) &sin.sin_addr, (char *) hp->h_addr, hp->h_length);
                sin.sin_port = (unsigned short) port;
                if (connect(s, (struct sockaddr *) &sin, sizeof(sin)) < 0)
                {
                        sleep(5);
                }
                else break;
        }
        fprintf(stderr, "Connected. Session capture begin\n");
        fp_signal=fopen(SIG_GO,"w");                    /* signal capture is read */
        fclose(fp_signal);
        fp=fopen(LOGFILE,"w");
        fprintf(fp,"StdIn script begin\n");
        if (debug)
        {
                printf("DEBUG MODE ON\n");
                printf("StdIn script begin\n");
        }
        for(;;)
        {
                if ((i = read(s, buf, 1)) > 0)
                {
                        if ((fp_signal=fopen(SIG_STOP,"r"))!=NULL)
                        {
                          fprintf(fp,"StdIn script end\n");
                          printf("Session capture end\n");
                          if (debug) printf("StdIn script end\n");
                          fflush(NULL);
                          fclose(fp);
                          fclose(fp_signal);
                          fp_signal=fopen(SIG_CLOSED,"w");/*signal log file closed */
                          fclose(fp_signal);
                          exit(OK);
                        }
                        else
                        {
                          buf0=buf[0];
                          if (insync) {
                                                count++;
                        switch(count) {
                                                        case 1: fprintf(fp, "%c:", buf0);
                                                                if (debug) printf("%c:",buf0);
                                                                break;
                                                        case 2: case 3:
                                                                if (buf0<15) fprintf(fp, "0");
                                                                fprintf(fp, "%X", buf0);
                                                                if (debug) {
                                                                  if (buf0<15) printf("0");
                                        printf("%X", buf0);
                                                                }
                                                                break;
                                                        case 4: if (buf0==10) {
                                                                  count=0;
                                                                  if (debug) printf("\n");
                                                                  fprintf(fp,"\n");
                                                                }
                                                                else {
                                                                  insync=0;
                                                                  if (debug) printf("*****\n");
                                                                  fprintf(fp, "*****\n");
                                                                }
                                                                break;
                                                }
                          }
                          else /* not insync*/
                                  if (buf0==10) {
                                                count=0;
                                                insync=1;
                                                if (debug) printf("\n");
                                                fprintf(fp,"\n");
```

```
                                        }
                                    }
                        }
                        else if (i == 0) break;
                        else
                        {
                                    exit(BARF);
                        }
            }
            fclose(fp);
}
/* end of program */
```

APPENDIX F: UNIX SHELL SCRIPT FOR USER TASK

```
#-------------------------------------------------------------------------
#  SCRIPT          je2
#-------------------------------------------------------------------------
#  This shell script copies the required input file to the
#   testing directory, then starts a copy of the log program
#  in the background and loads Jove and input file.
#
#  Usage: je2   taskname
#  History:          written by SC Leung
#-------------------------------------------------------------------------
#!/bin/sh

# Testing of New AutoRead for new firmware
if [ $# = 0 ]
then
  echo ""
  echo "There must be at least one argument"
  echo ""
  exit
fi

# Check if test environment is properly set up
if [ "s$id" = "snobody" ]
then
  echo ""
  echo "Please ask investigator to set up testing environment for you"
  echo ""
  exit
fi
#
clear
echo ""
echo ""
echo "\nWelcome to the experiment, $id.\n\n"
#
cd
cd Task
rm $1
cp DOC/$1 .
chmod a=rw $1
rm sig* 2>err
strtlog3&

# wait till go signal or error signal is detected (files beginning with sig)
until [ -f sig* ]
do
  cd .
done

# wait till log file is opened
if [ -f sig_go ]
then
if [ "s$stamp" = "snothing" ]
  then
    stamp=2
  fi
  echo "This is the $stamp time experiment.\n"
je $1
  echo . > sig_stop
  echo Finishing... Press [Space Bar]
  echo ""

# wait until log file is closed
  until [ -f sig_closed ]
  do
    cd .
  done
  cp log SAVE/$1/$1.$id$stamp
```

```
  chmod u=rx SAVE/$1/$1.$id*
  ls -l log SAVE/$1/$1.$id$stamp
else
  echo . > sig_stop
fi

stamp=
id=
exit

# the end
```

APPENDIX G: USER TASKS

The experiment involves working with four memos, each of which has a differrent task goal.

Application of Neural Network to User modelling

Investigator: SC Leung

Job Sheet

*It is important to keep your **normal style** of typing during this experiment*

TASK 1 Creating a memo

1. The investigator or his helper should assist you login the UNIX account and get the required files for experiment. Make sure you do not delete any file in the test directory.
   To go to test directory, type:
   ```
   $testdir
   ```

2. Please open the file **task1** with the editor "**je2".** The command to open the file is:
   ```
   je2 task1
   ```

3. You should see this on the screen before `Jove` editor is being brought up.
   ```
   Attempt ##
   Connected. Session capture begin
   ```

4. Type in **a new memo** as shown below. Make use of a page width of 70 characters.

5. Save the file when you have finished. You should see this on the screen when you exit `Jove` editor:
   ```
   Session capture end
   ```
   Go on with other tasks as instructed by the investigator.

(Please Turn Over)

```
---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
Date: 13 August 1995

To:   Mr. Van Buren, Vice President
From: Jane Spirakos, Assistant Training Coordinator
Subj: Declining Volunteer Training Job

I am very pleased that you considered me to direct the training sessions
for this year's United Fund Drive volunteers.

While that would be a challenging task, I believe you will need someone
with more experience than I have. You see I have been working as a trainer
for just six months. Also, since I am a new employee at Altamira and am
still in training for the job, my manager might be reluctant to give me
release time.

However, I would be interested in assisting with the campaign in some
less-demanding capacity. Just let me know how I might help. On the
content of the seminar, I think they can be organised as follows:

Sess  Activities                                    By
----  ----------                                    --
1.1   Pioneer 1000: Company Objective Review   General Manager
1.2   Pioneer 1000: What have we achieved            General Manager
1.3   Pioneer 1000 -> Pioneer 2000
      What should we emphasize?                Director

2.1   Pioneer 2000: Marketing Trend                  Trainer 1
2.2   Pioneer 2000: Marketing Direction '96    Marketing Manager

3.1   Pioneer 2000: Strategy:Discussion        Trainer 2
3.2   Pioneer 2000: Strategy:Group Presentation      Trainer 2
3.3   Pioneer 2000: What's next?               General Manager

I think the organising committee and the guest trainers should come up
with a much better idea.

Again, thank you for your gracious consideration. I look forward to
hearing from you again soon.
```

End of Task 1

Application of Neural Network to User modelling

Investigator: SC Leung

Job Sheet

*It is important to keep your **normal style** of typing during this experiment*

TASK 2 Addressing a memo

1. The investigator or his helper should assist you login the UNIX account and get the required files for experiment. Make sure you do not delete any file in the test directory.
   To go to test directory, type:
   ```
   $testdir
   ```

2. Please open the file **task2** with the editor "**je2**". The command to open the file is:
   ```
   je2 task2
   ```

3. You should see this on the screen before Jove editor is being brought up.
   ```
   Attempt ##
   Connected. Session capture begin
   ```

4. Type in the **bolded addresses** to the memos as shown below. Make use of a page width of 70 characters.

5. Save the file when you have finished. You should see this on the screen when you exit Jove editor:
   ```
   Session capture end
   ```
   Go on with other tasks as instructed by the investigator.

```
---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
================================================================================
```

MEMO 1

Date: 13 August 1995

To: Mr. Van Buren, Vice President
From: Jane Spirakos, Assistant Training Coordinator
Subj: Declining Volunteer Training Job

I am very pleased that you considered me to direct the training sessions for this year's United Fund Drive volunteers.

While that would be a challenging task, I believe you will need someone with more experience than I have. You see I have been working as a trainer for just six months. Also, since I am a new employee at Altamira and am still in training for the job, my manager might be reluctant to give me release time.

However, I would be interested in assisting with the campaign in some less-demanding capacity. Just let me know how I might help.

Again, thank you for your gracious consideration. I look forward to hearing from you again soon.

```
---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
================================================================================
```

MEMO 2

To: Sharon Willus, Supervisor, Production Control
From: Tom Hankins, Department Manager, Production Control
Subj: Status of Tuition Reimbursement Request

Date: 4 October 1995

It is always gratifying when one of our employees wants to continue her education. The company encourages this action by reimbursing employees for the cost of books and tuition for courses that contribute directly to the employee's work.

While the company is generous in its reimbursement plan it cannot reimburse courses that are not directly related to the employee's job.

For that reason, the proposed course in Twentieth Century Drama does not meet the criteria for reimbursement.

If you wish to take a qualifying course in future, please submit another request.

Please call me if I can assist you further.

---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
==========

MEMO 3

TO: All Sales Representatives
FROM: Jim Martins
DATE: July 15, 1992
SUBJ: Sales Meeting

Please attend a sales meeting on Friday, July 18 at 3 p.m. in the regional manager's office. We will discuss the attached quarterly sales totals.

This meeting is important because we may have to reduce dealerships and retail units.

End of Task 2

Application of Neural Network to User modelling

Investigator: SC Leung

Job Sheet

*It is important to keep your **normal style** of typing during this experiment*

TASK 3 Formatting a memo

1. The investigator or his helper should assist you login the UNIX account and get the required files for experiment. Make sure you do not delete any file in the test directory.
   To go to test directory, type:
   ```
   $testdir
   ```

2. Please open the file **task3** with the editor "**je2**". The command to open the file is:
   ```
   je2 task3
   ```

3. You should see this on the screen before `Jove` editor is being brought up.
   ```
   Attempt ##
   Connected. Session capture begin
   ```

4. The text is not properly formatted. Please format it as the sample below, i.e. you need to:
   - Change the page width (from 60) to 70
   - Properly paragraph the text
   - Add indents to points. (You can choose space or tab or whatever.
   - Give proper spacing between sentences.

5. Save the file when you have finished. You should see this on the screen when you exit `Jove` editor:
   ```
   Session capture end
   ```
   Go on with other tasks as instructed by the investigator.

(Please Turn Over)

Formatted text of Task3:

```
---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890

Date: 13 August 1995

To:   Mr. John Freeman
From: Marine Evers, CPA
Subj: Explain increase of bill

Dear John,

We understand your concern, and we hope the following will answer your
questions regarding the increase in our bill:

  1. Last year we spent 12 hours(@ $80/hour) preparing two tax
     returns for you.

  2. This year we spent 15 hours (@ $100/hour) preparing four tax
     returns for you.

  3. This year we successfully represented you in an audit with
     the Internal Revenue Service (3 hours @ 100/hour).

  4. This year we produced monthly financial statements for you,
     whereas last year we produced quarterly financial statements.

Please call me if you would like to discuss this further. We value you
as a client.



Mariame Evers
ME/jdc
```

End of Task 3

Unformatted text of Task3 (for reference only):

```
---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890

Date:13 August 1995 To:Mr. John Freeman
From:Marine Evers, CPA     Subj:Explain increase of bill

Dear John, We understand your concern, and we hope the
following will answer your questions regarding the increase
in our bill: 1.Last year we spent 12 hours(@ $80/hour)
preparing two tax returns or you. 2.This year we spent 15
```

hours (@ $100/hour) preparing four tax returns for you.
3. This year we successfully represented you in an audit with the Internal Revenue Service (3 hours @ 100/hour).
4.
This year we produced monthly financial statements for you,whereas last year we produced quarterly financial statements.Please call me if you would like to discuss this further. We value you as a client.
Mariame Evers ME/jdc

*** End ***

Application of Neural Network to User modelling

Investigator: SC Leung

Job Sheet

*It is important to keep your **normal style** of typing during this experiment*

TASK 4 Correcting error of a memo

1. The investigator or his helper should assist you login the UNIX account and get the required files for experiment. Make sure you do not delete any file in the test directory.
   To go to test directory, type:
   ```
   $testdir
   ```

2. Please open the file **task4** with the editor "**je2".** The command to open the file is:
   ```
   je2 task4
   ```

3. You should see this on the screen before `Jove` editor is being brought up.
   ```
   Attempt ##
   Connected. Session capture begin
   ```

4. Correct the typo-errors as shown below. Two words "**organization**" and "**(un)favorable**" are intentionally unmarked but you still need to correct all of them to "**organisation**" and "**(un)favourable**". Adjust the paragraphing if required. Make use of a page width of 70 characters.

5. Save the file when you have finished. You should see this on the screen when you exit `Jove` editor:
   ```
   Session capture end
   ```
   Go on with other tasks as instructed by the investigator.

(Please Turn Over)

Corrected text:

```
---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
Date: 13 August 1995

To:   All staff
From: Ian Wilson, Retail Manager, ANZ Bank
Subj: Reason for using forms to convey unfavourable messages to those
      outside the organisation
1.
Financial institutions such as banks, mortgage lending companies, credit
unions, and other need to convey unfavourable news more often than other
industries. They write to customers to announce the denial of credit cards,
mortgage loans, and other types of credits. As part of the Truth in Lending
Act, financial institutions must ensure that consumer's rights are not
abused. Therefore, they often use forms or form letters that have been
carefully designed to comply with the law and to respond quickly.

2.
Often, the form follows the usual unfavourable news strategy; however, all
possible reasons for the declination are listed. The organisation's
employees can, then, quickly check off the appropriate reason and respond
promptly to the customer or client. If a form letter is used, it usually
encourages the applicant to apply again when his or her circumstances have
changed. Some examples of change in circumstances include length of
residence at the same address, an increase in salary, or length of time on
a job. ~~Two examples of forms used by a lending institution to convey
unfavorable are attached to this memo.~~

~~4. Two examples of forms used by a lending institution to convey
unfavorable are attached to this memo.~~

3.
Unfavourable news for customers and clients can take many forms: a
student may be notified that his checking account has insufficient
funds to cover a check that he wrote to the bookstore, or an entrepreneur
may be notified by the IRS that she owes a penalty for underpaying a tax
liability. Often unpleasant news involves one's personal or business
finances. At other times, unfavourable messages involve problems with
handling customer orders, declining requests, and refusing employment.


4.
Two examples of forms used by a lending institution to convey unfavorable
are attached to this memo.
(cont. in next memo)
```

End of Task 4

Uncorrected text (for reference only):

```
---------1---------2---------3---------4---------5---------6---------7
12345678901234567890123456789012345678901234567890123456789012345678901234567890
Date: 13 August 1995

To:   All staff
From: Ian Wilson, Retail Manager, ANZ Bank
Subj: Reason for using forms to convey unfavorable messages to those
      outside the organization
1.
Financial institutions such as banks, morrtgage lending company, credit
unions, and other need to convey unfavorable news moer often than other
industry. They write to customers to announce the denail of credit cards,
mortgage loans, and other types of credits. As part of the truth in lending
act, financial institutional must ensure that customer's rights are not
abused. Therefore, they often use forms or form letters that have been
carefully designed to comply with the law and to respond quickly.

2.
Often, teh form follows the usual unfavorable news strategy. However, all
possible reasons for the declination arelisted. The organizations'
employees can, then, quickly check off the appropriate reason and respond
promptly to the customer or client. If a form letter is used, it usually
encourages the applicant to apply again when his or her circumstances have
changed. Some e.g. of change in situations include length of residence at
teh same address, an increase in salary, or length of time on a job. Two
examples of forms used by a lending institution to convey unfavorable are
attached to this memo.

4. Two examples of forms used by a lending institution to convey
unfavorable are attached to this memo.

3.
Unfavorable news for customers and clients can take many forms: a student
may be notified that his checking account has insufficient funds to cove a
check that he wrote to the bookstore, or an enterpreneur may be notified by
the I.R.S. that she owes a penalty for underpaying a tax liability, Often
unpleasant news involves problems with handling customer orders, declining
requests, and refusing employment.
(cont. in next memo)
```

***End ***

APPENDIX H: HOTKEYS TRACKED IN MLP-2

CATEGORY	KEYSTROKES	DESCRIPTION	CODE	TIMING
NAVIGATION				
	Esc-<	top of file	n_tof	Y
	Esc->	end of file	n_eof	Y
	Esc-G	goto line	n_gto	Y
	Ctrl-V	next page	n_npag	-
	Esc-V	prev page	n_ppag	Y
	Ctrl-A	begin of line	n_bol	-
	Ctrl-E	end of line	n_eol	-
	Esc-F	forward word	n_nwrd	Y
	Esc-B	backward word	n_pwrd	Y
	Esc-A	begin of sent	n_bos	Y
	Esc-E	end of sent	n_eos	Y
	Esc-,	top of window	n_tow	Y
	Esc-.	end of window	n_eow	Y
	Esc-M	goto 1st nonblank	n_nblk	Y
	Esc-]	next paragraph	n_npar	Y
EDIT				
- Change Case				
	Esc-L	change to lower	e_lcas	Y
	Esc-U	change to upper	e_ucas	Y
	Esc-C	change proper case	e_pcas	Y
- Fast Deleting				
	Ctrl-D	delete cursor	e_dcur	-
	Esc-D	delete word	e_dwrd	Y
	Ctrl-K	delete end of line	e_deol	-
	Esc-K	delete end of sent	e_deos	Y
	Esc-\	delete white spaces	e_dwsp	Y
- Other				
	Ctrl-U	repeat character	e_rep	-
	Ctrl-T	exchange characters	e_xch	-
BLOCK MARKING				
	Esc-<space>	set mark	m_set	Y
	Ctrl-W	kill marked	m_cut	-
	Ctrl-Y	yank	m_ynk	-
	Esc-Y	yank pop	m_ykp	Y
	Ctrl-X Ctrl-X	exchange mark	m_xch	Y
SEARCH & REPLACE				
	Esc-Q	query-replace	s_qrpl	Y
	Esc-R	replace	s_rpl	Y
	Ctrl-\	search forward	s_fwd	-
	Ctrl-R	search reverse	s_rev	-
COMMAND-LINE				
	Esc-X	command line	c_cmd	Y
	Esc-X;help/apropos	help	c_hlp	-

APPENDIX I: HOTKEYS TRACKED IN MLP-4

CATEGORY	KEYSTROKES	DESCRIPTION	CODE	TIMING
NAVIGATION				
	Esc-<	top of file	n_tof	Y
	Esc->	end of file	n_eof	Y
	Esc-G	goto line	n_gto	Y
	Ctrl-V	next page	n_npag	-
	Esc-V	prev page	n_ppag	Y
	Ctrl-A	begin of line	n_bol	-
	Ctrl-E	end of line	n_eol	-
	Esc-F	forward word	n_nwrd	Y
	Esc-B	backward word	n_pwrd	Y
EDIT				
- Change Case				
	Esc-L	change to lower	e_lcas	Y
	Esc-U	change to upper	e_ucas	Y
	Esc-C	change proper case	e_pcas	Y
- Fast Deleting				
	Ctrl-D	delete cursor	e_dcur	-
	Esc-D	delete word	e_dwrd	Y
	Ctrl-K	delete end of line	e_deol	-
- Other				
	Ctrl-T	exchange characters	e_xch	-
BLOCK MARKING				
	Esc-<space>	set mark	m_set	Y
	Ctrl-W	kill marked	m_cut	-
	Ctrl-Y	yank	m_ynk	-
	Esc-Y	yank pop	m_ykp	Y
SEARCH & REPLACE				
	Esc-Q	query-replace	s_qrpl	Y
	Esc-R	replace	s_rpl	Y
	Ctrl-\	search forward	s_fwd	-
	Ctrl-R	search reverse	s_rev	-

APPENDIX J: LISTING OF CLIPS PROGRAM FOR PRODUCTION RULES

```
;;;*****************************************
;;;* GLOBAL VARIABLES
;;;*****************************************

;;; width of PI triangle
(defglobal ?*width-match* = 0.35)
;;; metrics
(defglobal ?*eff-cursor* = 0.0)
(defglobal ?*eff-delete* = 0.0)
(defglobal ?*eff-overall* = 0.0)
(defglobal ?*feature-class* = 0.0)
(defglobal ?*feature-score* = 0.0)
(defglobal ?*feature-overall* = 0.0)
(defglobal ?*continue* = 1)
;;; read-in fields from pattern file
(defglobal ?*name* = 0)
(defglobal ?*level* = 0)
(defglobal ?*e_dcur* = 0)
(defglobal ?*e_deol* = 0)
(defglobal ?*e_dwrd* = 0)
(defglobal ?*e_lcas* = 0)
(defglobal ?*e_ucas* = 0)
(defglobal ?*e_pcas* = 0)
(defglobal ?*e_xch* = 0)
(defglobal ?*n_bol* = 0)
(defglobal ?*n_eol* = 0)
(defglobal ?*n_tof* = 0)
(defglobal ?*n_eof* = 0)
(defglobal ?*n_gto* = 0)
(defglobal ?*n_npag* = 0)
(defglobal ?*n_ppag* = 0)
(defglobal ?*n_nwrd* = 0)
(defglobal ?*n_pwrd* = 0)
(defglobal ?*m_set* = 0)
(defglobal ?*m_cut* = 0)
(defglobal ?*m_ynk* = 0)
(defglobal ?*m_ykp* = 0)
(defglobal ?*s_fwd* = 0)
(defglobal ?*s_rev* = 0)
(defglobal ?*s_rpl* = 0)
(defglobal ?*s_qrpl* = 0)
(defglobal ?*ebnb* = 0)
(defglobal ?*hkuse* = 0)
(defglobal ?*hktime* = 0)
(defglobal ?*hkerr* = 0)
(defglobal ?*eff_lf* = 0)
(defglobal ?*eff_rt* = 0)
(defglobal ?*eff_up* = 0)
(defglobal ?*eff_dn* = 0)
(defglobal ?*eff_bs* = 0)
(defglobal ?*eff_del* = 0)
```

```
;;;**************
;;;* TEMPLATES *
;;;**************
 (deftemplate feature-range "range of feature attemptted"
  0 4 level
  ((limited (z 1  1.7))
   (medium (pi .7  2))
   (broad (s 2.3  3)))
)

(deftemplate efficiency
  0 4 level
  ((low (z 1  1.7))
   (medium (pi .7  2))
   (high (s 2.3  3)))
)

(deftemplate expertise "expertise level"
  0 6 level
  ((novice (z 1 1.7))
   (novmed (pi .7 2))
   (medium (pi .7 3))
   (medexp (pi .7 4))
   (expert (s 4.3 5)))
)

(defrule print_expert
  (declare (salience -200))
  ?f  <- (feature-range ?)
  ?ef <- (efficiency ?)
  ?e  <- (crisp expertise ?exp)
  ?p  <- (printing)
  ?s  <- (signal ?)
=>
  (format t "       Expertise(%s) is %3.1f %n%n" ?*level* ?exp)
  (format outfile "%s Expertise(%s) is %3.1f %n%n" ?*name* ?*level* ?exp)
  (retract ?p ?e ?ef ?f)
)

(defrule match_defuzzy
   (declare (salience -100))
   ?s <- (signal defuzzy)
   ?f <- (expertise ?)
  =>
   (bind ?cexp (moment-defuzzify ?f))
   (assert (crisp expertise ?cexp))
   (retract ?s ?f)
)

(defrule match_fuzzy
   (declare (salience -100))
   ?s <- (signal fuzzy)
   ?f <- (crisp expertise ?exp)
```

```
  =>
   (assert (expertise (pi 1 ?exp)))
   (retract ?f ?s)
)

(deffunction defuzzy ()
   (assert (signal defuzzy))
   (run)
)

(deffunction fuzzy ()
   (assert (signal fuzzy))
   (run)
)

(deffunction fuzzify (?fztemplate ?value ?delta)
   (bind ?low (get-u-from ?fztemplate))
   (bind ?hi  (get-u-to   ?fztemplate))
   (if (<= ?value ?low)
   then
     (assert-string (format nil "(%s (%g 1.0) (%g 0.0))"
        ?fztemplate ?low ?delta))
   else
     (if (>= ?value ?hi)
     then
       (assert-string (format nil "(%s (%g 0.0) (%g 1.0))"
              ?fztemplate (- ?hi ?delta) ?hi))
     else
       (assert-string (format nil "(%s (%g 0.0) (%g 1.0) (%g 0.0))"
              ?fztemplate (max ?low (- ?value ?delta)) ?value (min ?hi (+ ?value
?delta)) ))
     )
   )
)

(deffunction initialise ()
    (bind ?*eff-cursor* 0)
    (bind ?*eff-delete* 0)
    (bind ?*eff-overall* 0)
    (bind ?*feature-class* 0)
    (bind ?*feature-score* 0)
    (bind ?*feature-overall* 0)
)

(deffunction features ()
   (bind ?edit-score (+ ?*e_dcur* ?*e_deol* ?*e_dwrd* ?*e_lcas* ?*e_ucas* ?*e_pcas*
?*e_xch*))
   (bind ?navig-score (+ ?*n_bol* ?*n_eol* ?*n_tof* ?*n_eof* ?*n_gto* ?*n_npag*
?*n_ppag* ?*n_nwrd* ?*n_pwrd*))
   (bind ?mark-score (+ ?*m_set* ?*m_cut* ?*m_ynk* ?*m_ykp*))
   (bind ?srch-score (+ ?*s_fwd* ?*s_rev* ?*s_rpl* ?*s_qrpl*))
   (bind ?*feature-score* (+ ?edit-score ?navig-score ?mark-score ?srch-score))
   (bind ?*feature-overall*
       (+ (* ?*feature-score* 0.0625 2) 1) )
```

```
   (format t "%s feature-overall=%3.1f %n" ?*name* ?*feature-overall*)
   (format outfile "%s feature-overall=%3.1f %n" ?*name* ?*feature-overall*)
(assert (feature-range (pi ?*width-match*  ?*feature-overall*)))
)

(deffunction efficiencies ()
   (bind ?*eff-cursor* (/ (+ (/ ?*eff_lf* 5) (/ ?*eff_rt* 5) (/ ?*eff_up* 4) (/
?*eff_dn* 4)) 4))
   (bind ?*eff-delete* (/ (+ (* (/ ?*eff_bs* 4) .5) (* (/ ?*eff_del* 5) 1.5)) 2))
   (bind ?*eff-overall* (+ (* ?*eff-cursor* 0.67) (* ?*eff-delete* 1.33) 1))
   (format           t "          efficiency-overall=%3.1f (efficiency-cursor=%3.1f
efficiency-delete=%3.1f)%n" ?*eff-overall* ?*eff-cursor* ?*eff-delete*)
   (format  outfile  "            efficiency-overall=%3.1f  (efficiency-cursor=%3.1f
efficiency-delete=%3.1f)%n" ?*eff-overall* ?*eff-cursor* ?*eff-delete*)
   (assert (efficiency (pi ?*width-match*  ?*eff-overall*)))
)


(deffunction dowork ()
    (close)
    (open "in.dat" infile "r")
    (readline infile)         ; skip header
    (open "out.dat" outfile "w")
    (printout outfile "header of outfile" crlf crlf)

  (while (= ?*continue* 1)
    (initialise)
    (bind ?*name* (read infile))
    (if (eq ?*name* EOF)
    then
      (progn
        (close)
        (bind ?*continue* 0)
      )
    else       ;;; read features
      (progn
        (bind ?*level*  (read infile)) ; edit
        (bind ?*e_dcur* (read infile))
        (bind ?*e_deol* (read infile))
        (bind ?*e_dwrd* (read infile))
        (bind ?*e_lcas* (read infile))
        (bind ?*e_ucas* (read infile))
        (bind ?*e_pcas* (read infile))
        (bind ?*e_xch*  (read infile))
        (bind ?*n_bol*  (read infile)) ; navigatiion
        (bind ?*n_eol*  (read infile))
        (bind ?*n_tof*  (read infile))
        (bind ?*n_eof*  (read infile))
        (bind ?*n_gto*  (read infile))
        (bind ?*n_npag* (read infile))
        (bind ?*n_ppag* (read infile))
        (bind ?*n_nwrd* (read infile))
        (bind ?*n_pwrd* (read infile))
        (bind ?*m_set*  (read infile)) ; block
```

```
        (bind ?*m_cut*  (read infile))
        (bind ?*m_ynk*  (read infile))
        (bind ?*m_ykp*  (read infile))
        (bind ?*s_fwd*  (read infile)) ; search & replace
        (bind ?*s_rev*  (read infile))
        (bind ?*s_rpl*  (read infile))
        (bind ?*s_qrpl* (read infile))
        ;read efficiency
        (bind ?*ebnb* (read infile))
        (bind ?*hkuse* (read infile))
        (bind ?*hktime* (read infile))
        (bind ?*hkerr* (read infile))
        (bind ?*eff_lf* (read infile))
        (bind ?*eff_rt* (read infile))
        (bind ?*eff_up* (read infile))
        (bind ?*eff_dn* (read infile))
        (bind ?*eff_bs* (read infile))
        (bind ?*eff_del* (read infile))
        (features)                          ;;; evaluate features score
        (efficiencies)                      ;;; evaluate efficiencies score
        (defuzzy)                           ;;; defuzzify expertise
        (assert (printing))
        (run)
      ) ;progn
    ) ;if
  ) ;while
  (close)
)

(defrule broad-high
    (feature-range broad)
    (efficiency high)
=>
    (assert (expertise expert))
)

(defrule broad-medium
    (feature-range broad)
    (efficiency medium)
=>
    (assert (expertise medexp))
)

(defrule broad-low
    (feature-range broad)
    (efficiency low)
=>
    (assert (expertise medium))
)

(defrule medium-high
    (feature-range medium)
    (efficiency high)
=>
```

```
    (assert (expertise medexp))
)

(defrule medium-medium
    (feature-range medium)
    (efficiency medium)
=>
    (assert (expertise medium))
)

(defrule medium-low
    (feature-range medium)
    (efficiency low)
=>
    (assert (expertise novmed))
)

(defrule limited-high
    (feature-range limited)
    (efficiency high)
=>
    (assert (expertise novmed))
)

(defrule limited-medium
    (feature-range limited)
    (efficiency medium)
=>
    (assert (expertise novice))
)

(defrule limited-low
    (feature-range limited)
    (efficiency low)
=>
    (assert (expertise novice))
)

(deffunction app-on-init ()
    (unwatch globals)
    (unwatch facts)
    (unwatch activations)
    (dowork)
)
```

D.B. Aitchison
Bookbinder
122 Windang Rd.
Primbee 2502
Ph 42742229