

Reasoning on Domain Knowledge Level in Human-Computer Interaction

CHAOCHANG CHIU

*Department of Information Management,
Yuan-Ze Institute of Technology,
Chung-Li, Taiwan*

ANTHONY F. NORCIO

*Department of Information Systems,
University of Maryland, Baltimore County,
Baltimore, Maryland 21228*

and

CHI-I HSU

*Department of Information and Software Systems Engineering,
George Mason University,
Fairfax, Virginia 22030***ABSTRACT**

This paper proposes an innovative approach for dynamically analyzing a user's dialog behavior and inferring a user's domain knowledge level simultaneously that combines neural networks, fuzzy cognitive maps, and fuzzy production rules. Further, this approach supports more cooperative human-computer interaction through dialog adaptation. Furthermore, when the user's knowledge level and problem-solving capability are inferred more accurately, there is more assurance that the system's interaction strategy can match more closely to the user's style. This research implements a neural network for classifying a user's performance pattern using UNIX file security commands. Input and output information that relate to a fuzzy cognitive map and fuzzy production rules are explained.

1. INTRODUCTION

One of the most difficult problems associated with a successful knowledge-based support system is providing the user with an appropriate level of advice or help. There are several fundamental aspects involved in perceiving the user's domain knowledge as well as determining appropriate

NOTICE TO READER:
This material may be protected by
copyright law (Title 17 U.S. Code)

information presentation. As Kraak [14] notes, the behavior of a novice with only a little system knowledge is quite different from an expert's behavior. For example, novice users are more prone to operation errors and usually require more frequent access to guided help than experts. Due to these differences between the various levels of knowledge, users behave differently in their problem-solving capabilities. Although users of different skill levels typically adopt different navigation strategies, they may also adopt similar strategies. However, users that are within the same skill level category are supposed to indicate a pattern that overlaps the patterns of other user categories.

Traditional methodologies such as rule-based production systems are inappropriate in real-time inferencing of a user's dialog behavior. These drawbacks include the following aspects:

- *Uncertainty Reasoning*: One of the most important criteria for evaluating expert systems is the correctness of its reasoning techniques [10]. The most frequently used uncertainty reasoning inference in traditional rule-based systems are based upon plausible assumptions. It is not clear that these techniques provide sufficient richness to handle the degree of uncertainty that is usually present [5].
- *System Overhead*: A major concern with traditional expert systems for modeling a user dynamically is that they are computationally expensive [9]. Thus a more efficient inference approach is required in a complicated dynamic environment.
- *Rule-Base Maintenance*: Maintenance of an expert system is critical in assuring the effectiveness of a complicated rule base.
- *Problem Representation*: Rule-based systems are not appropriate for modeling problems that are not well structured. Instead, other alternatives that are able to model the relationship between input and output information automatically are more appropriate.

This paper investigates how neural networks can overcome these drawbacks in modeling a user's knowledge level in an unstructured human-computer dialog activity. Also, neural networks and fuzzy-logic-based techniques, such as fuzzy cognitive maps (FCMs) [13] and fuzzy compositional rules of inference [21], are investigated for deducing a user's knowledge level by observing the interaction sequence. A methodology is proposed that infers a user's knowledge of a particular domain.

An implementation is under development that identifies a user's grade of membership for each knowledge level. The remainder of this paper discusses and explains the details of this approach.

2. METHODOLOGY

Using the following scheme, a fuzzy cognitive map, a neural network, and the compositional rule of inference can deduce a user's knowledge level.

- *Fuzzy Cognitive Map*: A fuzzy cognitive map (FCM) is one type of cognitive map, which consists of variables linked with fuzzy values that allow fuzzy mathematics operations [12]. A FCM is usually used for problems that involve a causal relationship between process entities and inherent "fuzziness" reasoning. The methodology explored here uses a FCM for inferring user implicit knowledge based upon the causal links between different concepts.
 - *Neural Networks*: Several studies suggest using neural networks for detecting human-computer dialog behavior. Ye and Salvendy [20] apply the back-propagation (BP) paradigm in an adaptive menu-based interface in the UNIX environment. Also, Fix [8] uses a BP network for modeling the performance of different human operators who control simulated cars on a circular racetrack. The network successfully learned to mimic the operator's performance. Maren [15] suggests the potential of using neural networks in enhancing man-machine systems. She indicates their usage in control, diagnostic, and monitoring of adaptive user interfaces. Bergeron et al. [1] use a Madaline Perceptron (MP) [2] for modeling the interactive behavior between students and a computer-based instructional system. Furthermore, Eberts uses the BP method for determining the efficiency of UNIX commands that provide advice in intelligent help systems [7]. Neural networks are generally useful in a data intensive pattern recognition task.
 - *Compositional Rule of Inference*: Based on the output generated by neural pattern recognition, a user can be classified into different predefined categories with respective grades of membership. In order to collapse these fuzzy values into a single crisp value, a set of linguistic production rules (based on the compositional rule of inference) is used to refine this direct observation. Finally, this crisp value signifies a user and influences the subsequent system interaction strategies.
- Newell [17] indicates "problem spaces imply that ranges of possible behaviors are to be expected.... The same subject on repeated occasions will exhibit a range of behavior, even though working in the same space."

Thus the network is used to monitor the user's interaction continuously and to update a user's grade of membership values for each predefined knowledge level.

3. IMPLEMENTATION

This paper presents a methodology that infers a user's UNIX command knowledge level by considering multiple commands in a session. The UNIX operating system provides a powerful command set, but this power prevents novice users from quickly utilizing its strengths. Consequently, a beginner usually uses only very limited and simple commands in a particular task without any intention of exploring more powerful and efficient commands. Eberts [7] indicates that a neural network can infer the efficiency for every input of a single UNIX command. However, in a real world situation, the possibilities that a user may follow other navigation strategies need to be considered. For example, a user might try different types of commands in sequences, which are mutually unrelated. This might indicate the user is just trying (or browsing) the command rather than directly solving the task. Providing dynamic advice by judging only a single command seems insufficient. On the other hand, the command that a user types might implicitly reflect his mastery of using this command.

Desmarais and Pavel [6] apply a conditional probability formula that infers a user's global knowledge (i.e., background knowledge) with a single command via propagating the interdependency knowledge structure. Though this method can successfully assess a user's knowledge, it can also significantly increase the system overhead, especially in a real-time mode. Several aspects, as the authors point out, need further improvement, including (1) adding an AND/OR type of structure to deal solely with implication relations and (2) including other types of relations to indicate efficient or inefficient command usage.

Instead of using conditional probability to assess user knowledge, this research approach applies a FCM to infer the global knowledge by propagating the casual relationship among command nodes. FCMs provide a powerful technique for analyzing a casual relationship that involves uncertainty. Also, FCMs allow integration of expertise from various knowledge sources. A FCM can be easily constructed by either domain expert(s) or other technical sources. Because global knowledge can be generated before an interactive session begins, system overhead can be reduced to a minimum. As previously pointed out, a single command is not sufficient to assess a user's knowledge level. Therefore, more than one command is needed to determine the inferences during the reasoning process. In this

work, five commands in sequence are necessary to discern a user's command performance pattern.

Specifically, the five commands are continuously analyzed in sequence. The command sequence can be viewed as a reflection of the user's cognitive organization for accomplishing the task. Based upon Miller's classic work [16] on the magical number 7 ± 2 , it is reasonable to assume that the user organizes the problem-solving procedures with five to nine steps. Clearly, these steps are manifested by the sequence of commands. Consequently, a sequence of five commands provides a sufficient minimum for determining the user's performance stability. The detailed technical design of reasoning a user's knowledge level of UNIX command usage is discussed in the following text.

3.1. OUTPUT FROM FCM AS INPUT FOR THE NEURAL NETWORK

The input source of the neural network is the output of the FCM. The FCM generates vectors that are activated by a specific navigation behavior through an iterative propagation process. The resulting vector is called a *global knowledge vector* (GKV). User file security on UNIX systems is made available through a few commands and features that form the basis for a complete security system. Figure 1 depicts a relationship representation for the subset of UNIX's file security commands. These commands are a subset of commands listed in the Appendix. The circles in Figure 1 indicate the command nodes and the links with the arrows show the effects and the directions of the effect one node has on the other node. The values of the link range from -1 to 1 . Negative values indicate the most negative effect, 0 means there is no relationship between two nodes, and 1 indicates the most positive effect. Thus, -1 has stronger negative influence than -0.1 , and 1 has stronger positive influence than 0.1 . The effect of node 4 (pack) on node 6 (ls) is 0.9 . This means that typing "pack" implies that this event (ls) is possible to be activated with degree 0.9 under the assumption that if the command is typed then it is mastered. The -1 effect from "pack" on "vi" indicates that this event (vi) is illegal to be activated right after the event of "pack." Based on Figure 1, the corresponding FCM, as shown in Table 1, is encoded by referencing the UNIX user manual [11]. For example, the entry value of transition from "crypt" to "rm" is 0.9 . This is because, normally, a user removes the original copy of the file after encrypting it, leaving this user with only the encrypted version.

Unlike viewing or editing the crypted files directly using the "vi" or "ed" command with $-x$ option, packed files must be first decrypted and

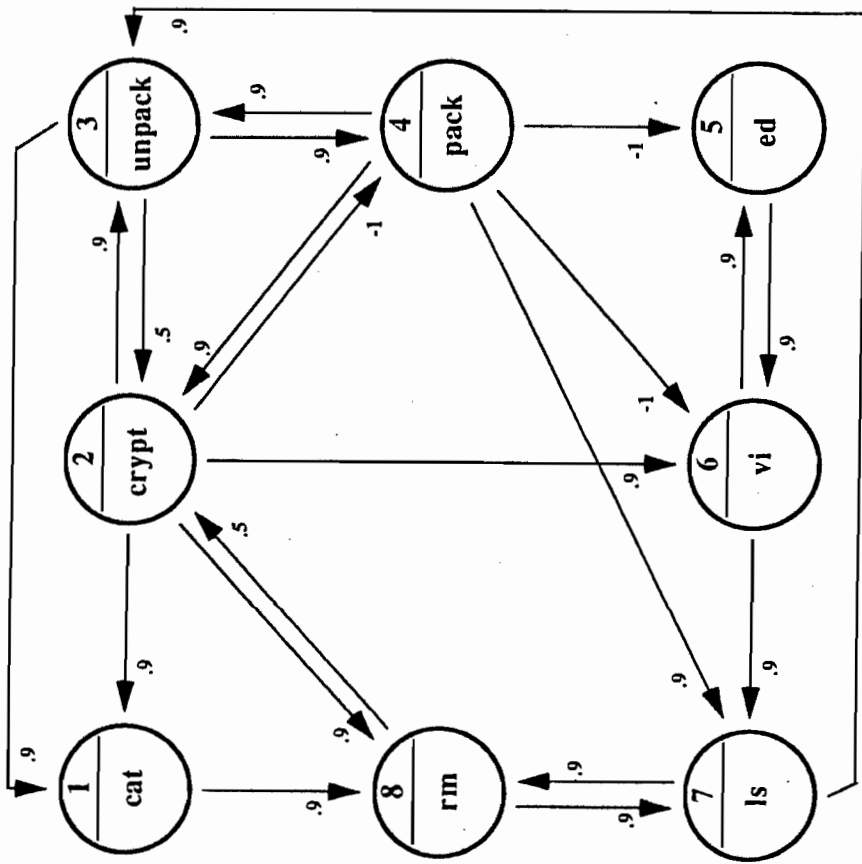


Fig. 1. UNIX file security commands.

TABLE 1
Example of FCM for UNIX File Security Commands

No.	1	2	3	4	5	6	7	8
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9
2	0.9	0.0	0.9	-1	0.0	0.9	0.0	0.9
3	0.9	0.5	0.0	0.9	0.0	0.0	0.0	0.0
4	0.0	0.9	0.9	0.0	-1	-1	0.9	0.0
5	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0
6	0.0	0.0	0.0	0.0	0.9	0.0	0.9	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9
8	0.9	0.5	0.0	0.0	0.0	0.0	0.9	0.0

unpacked to be edited. Thus, the entry values in the FCM for the connection from "pack" to "vi" and "ed" are assigned with -1 instead. All the entries with empty values imply that there are no straightforward or direct associations for the pairs of any two commands.

3.2. DYNAMIC PROPAGATION OF FCM

A FCM is similar to a neural network. It can be initiated by stimulus (activation) and allows vector-matrix multiplication continuously until reaching equilibrium. This work is not intended to explain the details of a FCMs internal inferencing process; further information can be found in [3], [13], [18].

The vector propagation process arrives at a stable state with a single propagated vector. However, this process may cycle and produce a series of stable propagated vectors. If this situation occurs, the system adopts the smallest value among the overall outcomes. For example, when any propagation from a specific node ends with a limit cycle, then all GKV's are integrated into single one by replacing 0 by either 1 or -1 and 1 by -1 if these values coexist. After the propagation equilibrium is achieved, for example, if the FCM is activated by node -4, then the resulting GVK is (1,1,1,1,-1,-1,1,1), which indicates that the user can go to all the nodes except node 5 and 6. Thus, every time a user inputs a new command, a corresponding entry in the resulting GVK is extracted.

All possible entries in GVK will be either -1, 0, or 1. For example, when a user types the *j*th command from *i*th command (the command that immediately follows the *i*th command), then the *j*th entry value is extracted from the resulting GVK. During the interactive session, every five commands the user types in sequence form a five-entry values vector (EVV). Each entry value is the link from each pair of events. Those five values need further preprocessing to form the input vector for the neural network in order to achieve better classification results.

3.3. NEURAL NETWORK INPUT ANALYSIS

Figure 2 shows how all possible performance patterns are classified into five different outcomes. Each performance group consists of several different patterns. The X axis indicates the sequence of commands and the Y axis indicates the relationship degree (i.e., EVV). Because there are many combinations for a single pattern, one pattern is used as the representative pattern for the others for the sake of convenience.

For example, there is a pattern (1,1,1,1,-1) in the Group 1. This pattern also represents all the following patterns as (1,1,1,-1,1), (1,1,-1,1,1), (1,-1,1,1,1), and (-1,1,1,1,1). Those possible combinations of patterns are all included in the training process. All the input, based on their special feature of performance patterns, is trained to map into one of the user's knowledge patterns, is trained to map

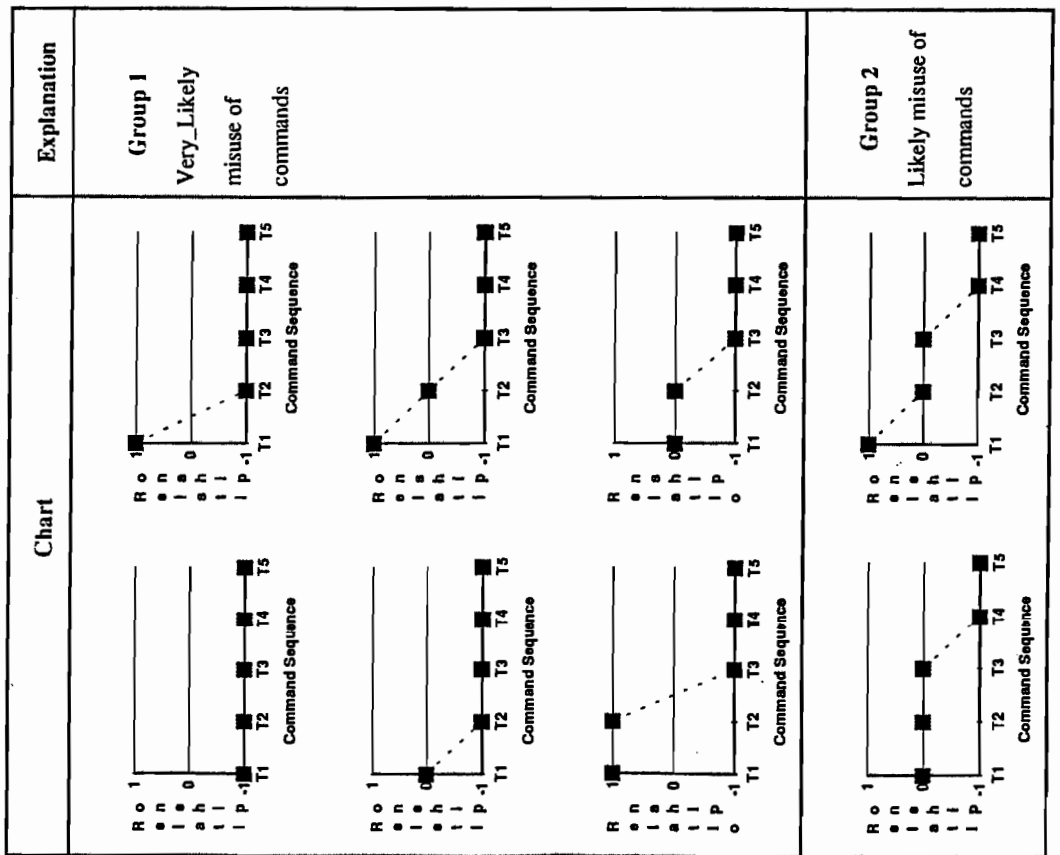


Fig. 2. Neural network input/output mapping.

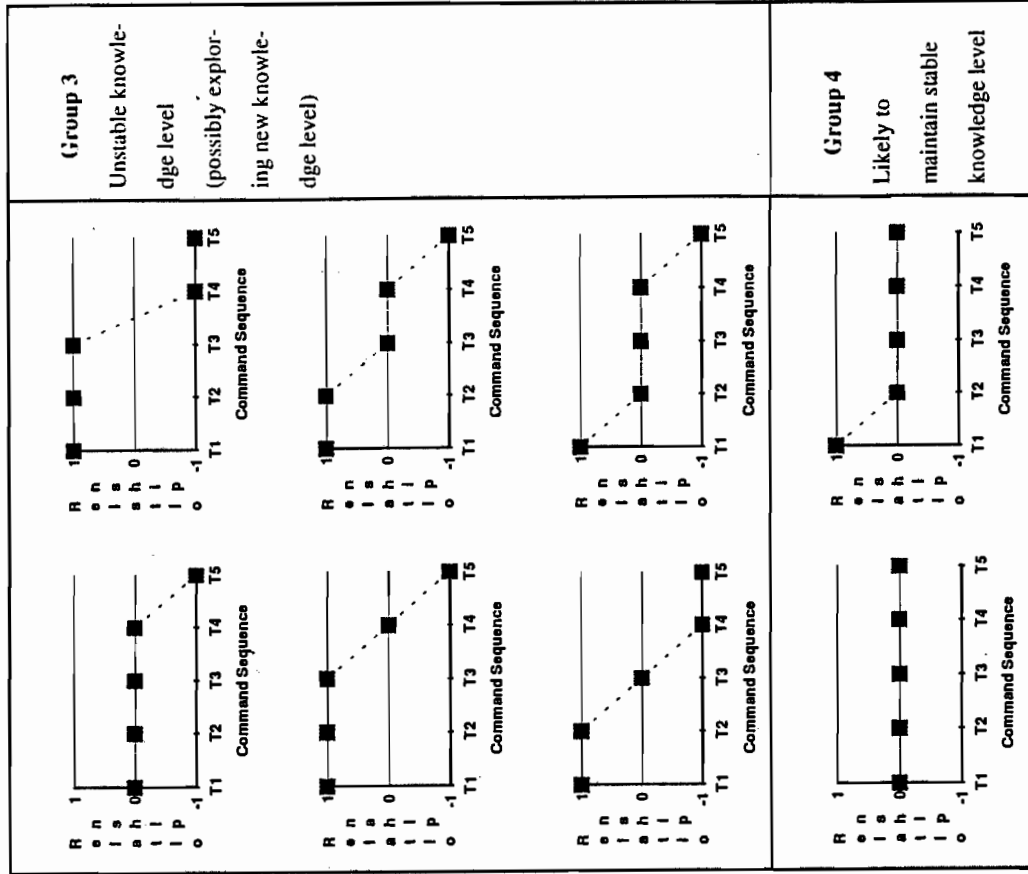


Fig. 2.—Continued

The purpose of this mapping is to determine the interdependence among the command sequences. This method can differentiate the stability of a user's performance pattern. Even though the neural network can identify the user's performance stability, it is not sufficient for inferring the user's actual knowledge level.

Therefore, additional analytic methods are required to identify the user's actual knowledge. Based on these additional methods, information

TABLE 2
Classification of a User's Performance Patterns

Classification	Explanation
Group 1 (VM)	Very—Likely misuse of commands
Group 2 (LM)	Likely misuse of commands
Group 3 (US)	Unstable knowledge level (possibly exploring new knowledge level)
Group 4 (LS)	Likely to maintain stable knowledge level
Group 5 (VS)	Very—Likely to maintain stable knowledge level

EXAMPLE : This section describes how an input vector EVV is trained to map into one of the five performance categories. As shown in Table 3, there are two EVVs, e.g., (1,0, -1, -1, -1) and (1,1,0,0, -1), that are preprocessed and mapped into Group 4 represented as (0,0,0,0,1) and Group 3 represented as (0,0,1,0,0), respectively. The data transformation and neural network mapping processes is described in Table 3. For instance, each EVV value (0, -1, or 1) is transformed into three-values entry, i.e., 0 is expressed by (0,1,0), 1 by (1,0,0), and -1 by (0,0,1). Thus, the neural network is trained to learn the whole combinations of five three-values entries for categorizing them into five categories.

3.5. NEURAL NETWORK MODEL AND RESULTS

Using Neuro Windows shells [19] running on MS Window, trials with different neural network configurations were conducted. The BP paradigm is used because it achieves effective classification results with reasonable training time required. This experiment consists of configurations of one hidden layer with various combinations of different numbers of process elements (PEs), learning rate, and momentum (five, six, and seven PEs in the hidden layer (0.40, 0.45, 0.50, 0.55, 0.60, and 0.65 for learning rate and 0.50, 0.55, 0.60, 0.65, and 0.70 for momentum are tried individually). Learning rate is a variable that can be specified to regulate how much the weights can be changed as the neural network is trained. Momentum is a

TABLE 3
Examples of Data Transformation for Neural Network Training Input

Transforming	Mapping
(1,0, -1, -1, -1) → (1,0,0), (0,1,0), (0,0,1), (0,0,1), (0,0,1) → (0,0,0,0,1)	(1,1,0,0, -1) → (1,0,0), (1,0,0), (0,1,0), (0,1,0), (0,0,1) → (0,0,1,0,0)

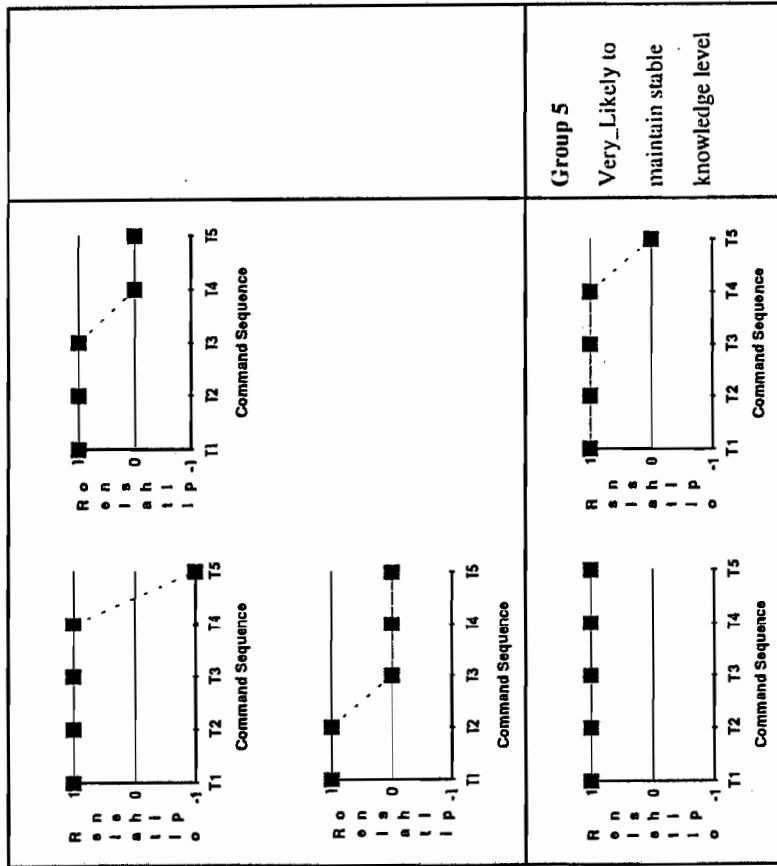


Fig. 2.—Continued

about the knowledge level of each command adopted as well as the average knowledge level based on all the commands used can be determined. Consequently, by using the fuzzy production rules, a user's actual knowledge level can be determined.

There is a more detailed description regarding how the compositional rule of inference can further analyze this information in Chiu [4]. This paper mainly focuses on using the FCM and the neural network to identify the stability of a user's command performance.

3.4. NEURAL NETWORK OUTPUT ANALYSIS

There are five possible classification outcomes when the neural network is given the input data. The outcomes of the neural network are summarized in Table 2.

TABLE 4
Neural Network Architecture

Layer	Number of Layers	Number of PEs
Input layer	1	15
Hidden layer	1	6
Output layer	1	5

variable that determines the proportion of the last weight change that is added to the new weight change. Output threshold is the value that is used to postprocess the original neural network output value. Epoch is defined as a full pass through the neural network of all the data in the training set. This operation classifies all the values greater than the threshold value (i.e., 0.9) into 1 and the rest into 0. The optimum BP neural network architecture, parameter, and training/test results are summarized in Tables 4-6. A screen shot from the final training and testing session is shown in Figure 3.

4. CONCLUSION

This paper discusses an innovative approach that integrates the FCM, a neural network, and fuzzy compositional rules of inference to deduce a user's knowledge level of the UNIX commands. Experimental results support the possibility of using neural networks for identifying the user's knowledge level in an adaptive human-computer interface in a real-time mode.

The benefits of applying neural networks can be achieved with increased system performance. A more powerful neural network combined with FCM and compositional rules of inference is to be explored next. Furthermore, how to apply this approach into real world applications such as a computer-aided instructional system, an on-line advice system, or a

TABLE 5

Training Parameters	
Learning rate	0.45
Momentum	0.65
Output threshold	0.9
Upper bound of average error	0.008

TABLE 6
Training and Testing Results

Number of training set	243
Number of testing set	50
Average error of training	4.609178E-03
Number of epoch	45
Testing accuracy rate	100%

tutorial system, all of which require identifying the user's domain knowledge level, can be further explored.

APPENDIX

Subsets of UNIX File Commands and Description

No.	Command	Description
1	cat	concatenate and print files
2	cd	change working directory
3	chgrp	change group of a file or directory
4	chmod	change mode of a file or directory
5	chown	change owner of a file or director
6	cp	copy files
7	crypt	encrypt/decrypt files
8	cut	cut out selected columns from each line of a file
9	echo	print argument
10	ed	text editor
11	fsck	check the consistency of file systems and directories
12	grep	search a file for a pattern
13	id	print user and group id's and names
14	ls	list contents of directories
15	makekey	generate encryption key
16	mkdir	make directory
17	mv	move files
18	pack	compress files
19	pwd	working directory name
20	rm	removes files
21	sort	sort and/or merge files
22	su	become super-user or another user
23	unpack	uncompressed files
24	vi	screen editor
25	TypeError	spelling error of UNIX-based commands
26	UnRecognized	unrecognized non UNIX-based commands

REFERENCES

1. B. P. Bergeron, A. N. Morse, and R. A. Greenes, A generic neural network-based tutorial supervisor for computer aided instruction, *Proc. Fourteenth ISSS Annual Symposium of Medical Applications in Medical Care*, 1990, pp. 435-439.
2. G. A. Carpenter, Neural network models for pattern recognition, *Neural Networks* 2(4):243-257 (1989).
3. M. Caudill, Using neural nets: Fuzzy cognitive map, *AI Expert*, June: 49-53 (1990).
4. C. Chiu, Reasoning about domain knowledge level for dynamic user modeling in adaptive human-computer interfaces: A fuzzy logic/neural network approach, Ph.D. Dissertation, University of Maryland Graduate School, Baltimore, Maryland, 1993.
5. C. Chiu, A. F. Norcio, and K. E. Petrucci, Using neural networks and expert systems to model users in an object-oriented environment, *Proc. 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 1991, pp. 1943-1948.
6. M. C. Desmarais, and M. Pavel, User knowledge evaluation: An experiment with UNIX, *Human-Computer Interaction-INTERACT'87*, (H. J. Bullinger and B. Shackel, eds.), Elsevier Science Publishing, Amsterdam, 1987, pp. 151-156.
7. R. Eberts, Knowledge acquisition using neural networks for intelligent interface design, *Proc. 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 1991, pp. 1331-1335.
8. E. Fix, Modeling human performance with neural networks, *Proc. Third International Joint Conference Neural Networks*, San Diego, CA, 1990, 1-247-252.
9. K. L. Fox, R. R. Henning, J. H. Reed, and R. P. Simonian, A neural network approach towards intrusion detection, *Proc. 13th National Computer Security Conference*, 1990, pp. 125-134.
10. J. Gaschnig, P. E. Klar, E. Shortliffe, and A. Terry, Evaluation of expert systems: Issue and case studies, in *Building Expert Systems*, (F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, eds.), Addison-Wesley Publishing, Reading, MA, 1983.
11. S. G. Kochan, and P. H. Wood, *Exploring the UNIX System*, 2nd ed., Hayden Book, New York, 1989.
12. B. Kosko, Fuzzy cognitive maps, *Int. J. Man-Machine Studies*, January: 65-75, (1986).
13. B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamic System Approach to Machine Intelligence*, Prentice-Hall Publishing, Englewood Cliffs, NJ, 1992.
14. L. Kraak, Multi-level user interfaces: Software tools and application, *Human-Computer Interaction-INTERACT'87*, (H. J. Bullinger and B. Shackel, eds.), Elsevier Science Publishing, Amsterdam, 1987, pp. 703-708.
15. A. J. Maren, Neural networks for man/machine systems, in *Handbook of Neural Computing Applications*, (A. J. Maren, C. T. Harston, and R. M. Pap, eds.), Academic Press, Inc., New York, 1991, pp. 419-425.
16. G. A. Miller, The magical number seven plus or minus two: Some limits on our capacity for processing information, *Psych. Rev.*, 63(2):81-96 (1956).
17. A. Newell, Reasoning problem solving and decision processes: The problem space as a fundamental category, in *Attention and Performance*, vol. III, (R. Nickerson, ed.), Erlbaum Publishing, Hillsdale, NJ, 1980, pp. 693-718.
18. R. Taber, Knowledge processing with fuzzy cognitive map, *Expert Systems With Applications* 2:83-87 (1991).

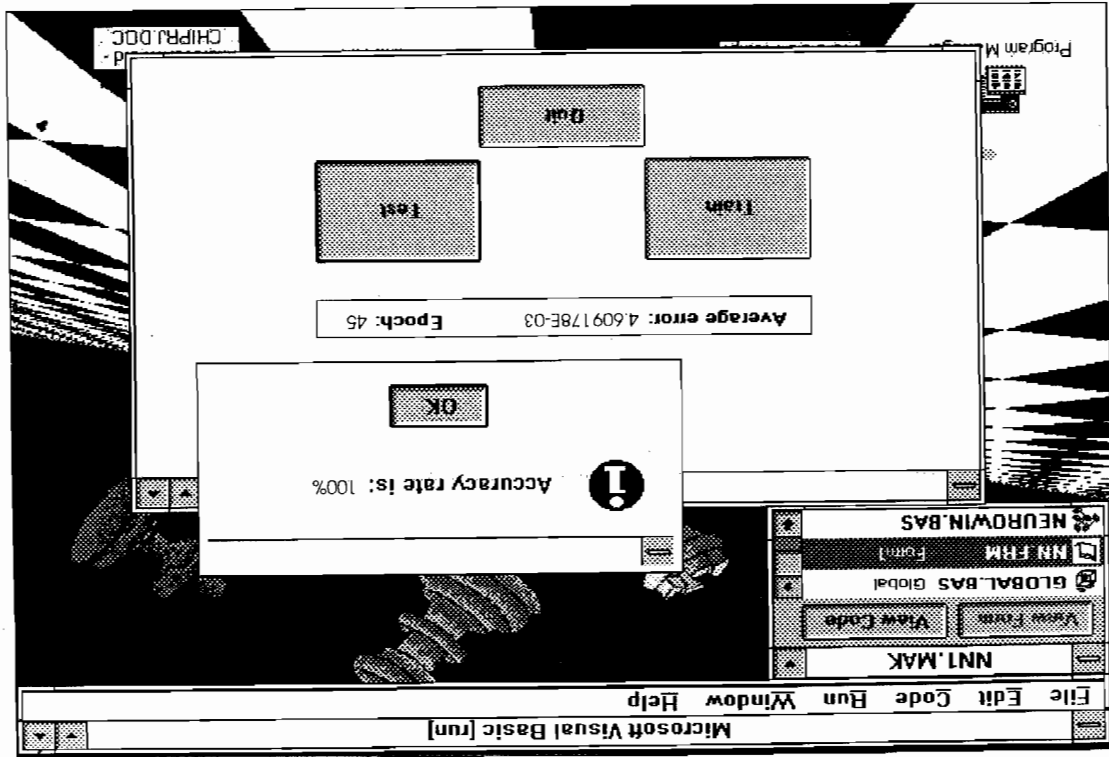


Fig. 3. Screen shot of the training and testing results.

19. Ward Systems Group, NeuroWindows, 1991.
20. N. Ye, and G. Salvendy, An adaptive interface design using neural networks, in *Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals*, (H. J. Bullinger, ed.), Elsevier Science Publishing, Amsterdam, 1991, pp. 435-439.
21. L. A. Zadeh, Outline of a new approach to the analysis of a complex systems and decision process, *IEEE Trans. Systems, Man, and Cybernetics* 3(1):28-44 (1973).

Received 1 April 1992; revised 2 August 1993