

Neural Network Based Stereotyping for User Profiles

Q. Chen¹, A. F. Norcio² and J. Wang¹

¹Department of Information and Decision Sciences, Montclair State University, Upper Montclair, NJ; ²Department of Information Systems, University of Maryland, Baltimore County, MD, USA

This paper discusses an approach of establishing system models of users' task related characteristics, such as domain knowledge in human-computer interaction. Several neural networks are tested for the modelling process. These networks function as associative memories that capture the causal relationships among assumptions about the users' characteristics. The outputs from the networks are considered as stereotypes assigned to individual users. It is suggested that this approach can be expected to overcome some limitations of user modelling approaches in terms of pattern recognition and classification.

Keywords: Associative memory; Human-computer interaction; Stereo typing; User model

1. Introduction

Modern interface systems are event-driven. To adapt users' actions, it is very important for an interface system to be able to predict 'what this user knows' and 'what this user wants to do next'. This means that systems need to establish and maintain a set of assumptions (i.e. system beliefs) about the users' task-related characteristics. The construction of this set of assumptions is often referred to as *user modelling*. The assumptions may vary depending on the application domain. Usually, they are related to the users' plans, goals and domain knowledge, as well as their cognitive preferences [1]. These assumptions can be categorised in terms of several

dimensions. Rich [2] proposed a three-dimensional taxonomy of user models: the short-term vs. long-term dimension is concerned with the persistence of modelled user characteristics over time. The explicit vs. implicit dimension is concerned with whether the models are built by the users themselves, or by the system. The canonical vs. individual dimension looks at whether a model is specified for a single user or for a group of users. Generally, the dimensional categorisation of user models aims at classifying the assumptions about users in terms of the time period in which they are valid, the way in which they are elicited and represented, as well as the degree to which they are specified [3].

A hierarchical stereotype approach is the common user modelling technique used to initialise these assumptions. This approach predefines the assumptions into classes (i.e. stereotype classes), and organises them into a generalisation hierarchy [2,3], also called a stereotype hierarchy. Stereotyping concerns the classes of users, and makes rapid inferences about the users' characteristics. The upper-level stereotypes characterise generic information, and the lower-level stereotypes carry specific information and inherit the assumptions from their ancestors. An individual user profile is actually a substructure of the stereotype hierarchy. The function of the inference mechanism is to extract the substructures ascribed to users without causing any inconsistency.

2. Limitations in Conventional User Modelling

During human-computer interaction, user modelling proceeds with stereotype assignment through default reasoning, which allows a user model to retain the stereotypical knowledge about a user in the absence

Correspondence and offprint requests to: Dr Qiyang Chen, Department of Information and Decision Sciences, Montclair State University, Upper Montclair, NJ 07043, USA. Email: chenq@mail.montclair.edu

of evidence to the contrary. This approach provides a simple way in which to initiate a user model, and is successful in some applications. However, this approach has several limitations.

Since the reasoning process is based on default assumptions that may conflict with the new evidence obtained as the interaction progresses, the revision of stereotypical knowledge is necessary to handle the inconsistencies. A common suggestion is to use a dependency-directed backtracking process, referred to as *truth maintenance*, to eliminate the inconsistencies [4,5]. This process utilises the sequential logic to examine one piece of information at a time. It is often inefficient, and lacks the ability to detect noisy or inconsistent information that should be ignored [3]. Therefore, it is possible that the effort of maintaining consistency may bring in further conflicts in the subsequent interaction. Thus, model construction may fall into the dilemma where a non-monotonic process of reconciling conflicts is frequently involved, and eventually, no decision can be made after a period of interaction [6].

The pre-defined generalisation hierarchy limits the system's assumptions within each stereotype that can be only inherited by the descendant stereotypes. Therefore, it is difficult to update those assumptions that are no longer significant in the context of user modelling. Since a user may fail to fit any set of stereotypes, the modelling process may fail to associate any system decision to that user [3]. In such a situation, however, some assumptions distributed among the stereotypes might still be useful for characterising that user. In this sense, stereotyping approaches have only a limited ability of individualisation.

In addition, conventional user modelling systems tend to be rule-based systems that often encounter problems of knowledge elicitation. It is often difficult to specify or update the relationships among the assumptions and stereotypes. Also, rule-based systems lack any learning ability. The dynamic maintenance of the rule base is often inefficient and error-prone [7].

In this paper, we present an alternative approach that utilises neural networks as the knowledge representation and inference mechanism to establish system beliefs about the users' task-related domain knowledge. Such beliefs are an important part of user profiles.

3. User Modelling and Pattern Recognition

It has been suggested that the system's beliefs about users should be based on the context of the user's

task performance. The fragmented pieces of observations may not result in any meaningful implications if they are not examined associatively [3,8]. In addition, the information observed about a user's characteristics may be mixed with noise or inconsistencies. Therefore, all aspects of the user's performance patterns must be examined before any system decision can be made. In other words, the information about a user should be processed by pattern recognition so that the system can establish complete and consistent user profiles.

Neural networks exhibit powerful features in pattern association and classification, fault tolerance, graceful degradation and signal enhancement. These features are also required in user modelling. In addition, neural networks can be trained to generalise inferences. In contrast, conventional stereotyping approaches that use rule-based inference to proceed through sequential logic may become seriously inadequate for processing pattern-formatted information, especially if there is incomplete, noisy or inconsistent information involved [9].

In neural network-based user modelling, network nodes represent system assumptions. All assumptions are considered to be relevant to each other in a spectrum that is valued from negative to positive. The modelling process extracts some assumptions to form a stereotype that fits a particular user. Unlike the hierarchical stereotyping approaches discussed in Section 1, which use a generalisation hierarchy to model users at the stereotype-level, the proposed approach proceeds at the assumption-level. It overcomes the limitation of the hierarchical stereotypes, which is unable to extract assumptions from different stereotypes to form a new stereotype.

Several network paradigms are used to test the proposed approach. The testbed application is to model the user's domain knowledge of Unix file commands. An interface system can tailor its response to users in file manipulation if the system has information about the users' domain knowledge. For example, formulating the help information on how to use *chmod* command may depend upon the user model, which shows whether or not the current user knows how to create a file or directory. This adaptation is particularly useful in applications such as help systems, information retrieval systems, tutoring systems and visual programming.

4. Using Pattern Association for Default Reasoning

Default reasoning is to infer *what the user knows* in a task domain [3]. The reasoning process creates

a large number of assumptions as the output, based on a small number of assumptions in the input [4]. Pattern association using neural networks can simulate default reasoning of rule-based systems where sequential logic is applied. Two networks, Bidirectional Associative Memory [10,11] and the Backpropagation model, are implemented and tested for pattern association.

4.1. Capture the Causal Relationships

A Bidirectional Associative Memory (BAM) is used to capture the causal relationships between assumptions. Each node represents an assumption. The relationships among the assumptions are weighted under certain conditions. Once a user's input from the dialogue channel is observed, it forms an input pattern to the BAM. The modelling process is initiated by activating a few nodes as input, and propagating the activation throughout the network. Once all nodes reach stable activation levels, the assumptions activated in the output pattern are considered to be the current system beliefs about the user's domain knowledge.

A weight matrix is used for representing the causal relationships. This matrix explores how the knowledge providers conceptualise the domain and the relevance among the assumptions in the domain. This study uses a group of Unix commands in file processing as the testbed (Table 1). A user profile of domain knowledge is a collection of assumptions on whether or not a user knows how to use these commands. There are 30 positive nodes and 30 negative nodes in the BAM. A positive node represents the assumption that a command is known to users; a negative node represents the assumption that a command is unknown to users.

A group of 34 programmers who have between two and ten years of Unix experience participated in the data collection. Each subject is asked to create a weight matrix expressing the causal relationships between assumptions. Assuming that a user under-

stands (or does not understand) a command, subjects are asked to choose other possible commands the user might also understand (or does not understand), and assign the belief values to the corresponding cells. For example, if it is believed that a user who knows command x may also know command y , then fill '1' into the corresponding cell in the matrix. Subjects may use any number between -1 and 1 to value such beliefs. For example, if the value of w_{ij} is 1 , it means that a user who understands command i must understand command j . The subjects are also instructed not to work on the commands that they are not familiar with. An average function is used to consolidate the matrices from the subjects.

The propagation algorithm has many variances that can be either linear, non-linear or fuzzy logic [10,11]. This study uses a non-linear propagation activation algorithm to associate the input pattern with an output pattern. Two statuses can be reached using this algorithm: a deterministic output, or a mode in which outputs cycle among several patterns. For the second situation, a union operation is applied to the cyclic output patterns to generate the final output.

There are 160 different input patterns designed for training and testing the network, which sufficiently cover representative input patterns. The test results show that the output pattern has more activated assumptions than the corresponding input pattern. This implies that, given a small number of assumptions, a larger number of correlated assumptions is activated. This simulates the behaviour of default reasoning. For example, when the input pattern is $(0,0,0,0,0,1,0,0,\dots,0)$, which indicates that a user understands the command *chmod*, the associated output pattern is $(1,1,1,1,1,1,1,1,\dots,0)$, which indicates that the user also understands the commands of No. 1 to No. 10 in Table 1. Of the output patterns 95% satisfied the following conditions:

- (a) The advanced commands in the input yield less advanced commands. The advanced commands are those that are more difficult to understand than others. The criterion for determining the level of commands is based on general curriculum design for Unix file operation [12]. For example, the command *chmod* is more advanced than the command *mkdir*.
- (b) The inconsistent input does not yield inconsistencies in the output pattern. For example, if the input contains *chmod*, \neg *chmod* and *mkdir*, the output pattern contains the assumptions only implied by *mkdir*.

Table 1. The UNIX commands for file processing.

| | | | | | |
|----|-------|----|------|----|-------|
| 1 | cd | 11 | comm | 21 | awk |
| 2 | mkdir | 12 | cut | 22 | cat |
| 3 | rmdir | 13 | diff | 23 | join |
| 4 | ls | 14 | grep | 24 | paste |
| 5 | man | 15 | head | 25 | pr |
| 6 | chmod | 16 | tail | 26 | sort |
| 7 | cp | 17 | uniq | 27 | sed |
| 8 | mv | 18 | wc | 28 | tr |
| 9 | cat | 19 | less | 29 | fmt |
| 10 | rm | 20 | dd | 30 | touch |

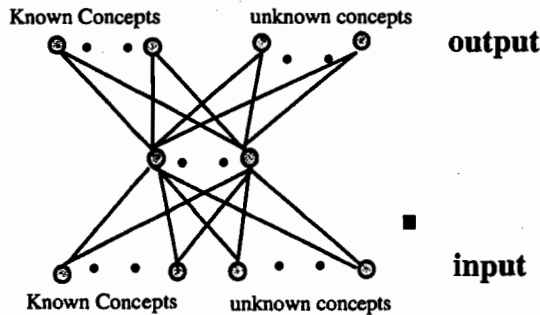


Fig. 1. A backpropagation network.

4.2. Pattern Association

A Backpropagation (BP) network is also tested for pattern association in user modelling. The BP network uses a nonlinear activation algorithm that enforces the ability of generalisation [9]. The pairs of input/output patterns from the BAM model are used to train the BP network to generalise the associative information stored in the BAM. A three-layer network is implemented, shown in Fig. 1. The nodes of *known concepts* represent the assumptions that the user knows how to use the commands. The nodes of *unknown concepts* represent the assumptions that the user does not know how to use the commands.

The training data allows the conflicting assumptions to offset each other's influence, so that inconsistencies do not appear in the output pattern. Table 2 shows the training and testing results. The recall accuracy is 99%. For the testing data (i.e. input patterns that are not in the training set), 94% of the testing results satisfy conditions (a) and (b) mentioned in Section 4.1.

4.3. Partial Training and Generalisation

We have also considered the situation in which a new assumption may be added in the modelling process, which implies structural change to the network. The network's ability to generalise inferences on associated assumptions must be tested. The structural change requires the network to be retrained. In order to retrain the network, and meanwhile to

preserve the effects of previous training, we designed and tested a partial training approach, described as follows:

- The training data for a new command is organised in the way that reflects the closeness between the new assumptions and the existing assumptions. In other words, the patterns of training data for the new commands can teach the network to trigger nodes that are triggered by functionally closed commands. For example, once a command *more* is added, it should yield similar assumptions as does the command *cat*.
- Fix all weights except for those that are on the newly added connections (i.e. the fixed connections do not participate in the training process).
- During the training period, present the new training data on both the input and output layers. The input vector contains the stimulus on the new command. The output vector contains all assumptions implied by the new command.

After partial training, the weight pattern on the newly added connections is established. This weight pattern can provide a similar activation in the output for the similar assumptions in the input. Four new commands (*pwd*, *more*, *cmp*, *cpio*) are added to the network. The training results are shown in Table 3. The recall accuracy is 100%. The network converges much faster. For the testing data (i.e. input patterns that are not in the training set), 92% of the testing results satisfy conditions (a) and (b) mentioned in Section 4.1.

The test result also shows that functionally correlated commands yield the same assumptions in output. This result implies that the network can generalise its reasoning ability to adapt new assumptions without being totally retrained. This feature is particularly important for a dynamic modelling process that updates the structure of system belief space frequently.

Unlike the hierarchical stereotyping approach, which uses inheritance as the basic form of reasoning and generalisation, our approach implements the generalisation by deriving the similarity of pattern representation. This approach has the ability of default reasoning through pattern association, and of uncertainty management due to the nature of neural

Table 2. Training and testing results.

| | |
|---------------------------|--------|
| Learning rate | 0.50 |
| Number of training cycles | 34,000 |
| Number of training data | 160 |
| Testing data | 90 |
| Recall accuracy | 99% |

Table 3. Training and testing results for partial training.

| | |
|---------------------------|-------|
| Number of training cycles | 4,400 |
| Number of training data | 40 |
| Testing data | 25 |
| Recall accuracy | 100% |

networks. In addition, the partial training approach preserves the previous training effects to allow fast adjustment of the system beliefs, which facilitates the dynamic adaptation in the modelling process.

5. Analysis on Completeness

As suggested in this study, because the assumptions about users are often correlated, a user model should be created by pattern recognition and classification in order to generate relatively complete and consistent system beliefs about users. We should consider three basic aspects that affect the performance of pattern recognition in the context of user modelling:

- *Consistency*: this means that inconsistent assumptions in input do not yield the inconsistent assumptions in output.
- *Ability of generalisation*: given a new input pattern that is not included in the training set, or changing the network structure to adapt to a new assumption, the network can still generate the correct output without being totally retrained.
- *Completeness*: given an input pattern, the associated output pattern reveals correlated information. This means that, given a few assumptions as input, a system can produce adequate assumptions as output.

We have discussed the first two aspects in the previous sections. For testing the completeness of pattern association, we conducted a comparison study similar to that presented in Chen and Noreio [13]. A hierarchical cluster analysis is used to test whether the pattern association is relatively complete. A comparison is made between the output patterns generated by the networks, and the clusters derived from a distance matrix in cluster analysis.

The same set of commands (Table 2) is used to create the distance matrix. We group the commands according to their functional relationship. Commands can appear in more than one group. Commands are sorted in each group by the level of difficulty (i.e. the first command in a group is the easiest to understand, and the last one in a group is the most difficult to understand). Each cell of the matrix is assigned the value of frequency for a pair of commands that is grouped. Note that this distance matrix is different from the BAM matrix in semantics. The BAM matrix only captures the causal relationships from among the assumptions, whereas the distance matrix reflects the functional relationships among the assumptions. For example, a user who knows the command *fmt* must know the command *cd*, which is implied by a positive link between these

two commands in the BAM matrix. However, these two commands are not functionally related, which is implied by a smaller cell value in the distance matrix. Generally, the commands listed in the second column of Table 1 (Nos 11 to 20) are likely to be grouped together, because they relate to extracting information from files. The commands in the third column (Nos 21 to 30) are likely to be grouped together because they relate to file manipulation and transformation.

Clusters are created using the distance matrix and compared to the output patterns from the networks. The comparison shows that the output from the networks exhibit both functional relationships and causal relationships among the commands. In other words, the network output implies that if a user knows a command in one cluster, he/she may also know some commands in the same cluster. Furthermore, an advanced command not only triggers the commands/assumptions within the cluster, but also triggers some less advanced commands/assumptions in other clusters. It is also noticed that functionally-related commands in the same clusters do not have the same power to trigger each other. This implies that less advanced commands cannot trigger advanced assumptions, even though they are functionally related. This means that the network output reveals more information than cluster analysis, because it reveals not only categorical relationships, but also the causal relationships among assumptions.

6. User Classification using ART

In our approach, the output from a network is considered as a user's current knowledge pattern. This pattern can be further classified into a certain category, based on which an interface system can generate its response. An Adaptive Resonance Theory (ART) module is tested for classifying the users' knowledge patterns. It receives other network outputs and classifies them based on the pattern similarity. Conceptually, the comparison layer of ART stores the patterns learned from the unsupervised training process. The closeness of the input patterns is controlled by a single scalar called *vigilance*. The comparison layer recalls a pattern that is closest to the input. The categorical nodes in the recognition layer responds to the recalled pattern in a competitive style [15]. Eventually, only one categorical node is activated for the given input. Since the ART module does not need supervised training, it is particularly useful when there is a lack of categorical information about users.

The outputs of the BP network are used as the

input to the ART module. Five output nodes in the recognition layer are used to indicate user categories. They represent the categories of expert, expert-intermediate, intermediate, intermediate-novice, and novice. The data used for training and testing the BP module is used for ART module training and testing. The results show that the network can match the stored-patterns to input patterns, and activates the corresponding categorical node in output layer. The vigilance is set to 0.8, with a learning rate of 0.9. The test results also show that, even though the test patterns are different from the training patterns, the comparison layer can still invoke a pattern that matches the input with a slight variance.

7. A Framework of Module Integration

The network modules used in this study can be integrated into a framework in which each model functions as a knowledge source that can work either independently or cooperatively. This framework has been simulated by presenting the output of a network to the input layer of another network. Figure 2 shows the simulation process. The construction of the BAM matrix is the process of knowledge elicitation. The BAM's input and output are used as input (or training data) to the BP model; the output from the BP or BAM are presented to the ART module for pattern classification. The output from any network is viewed as part of the current user profile.

In this framework, each module can be used separately for multiple purposes. The training pro-

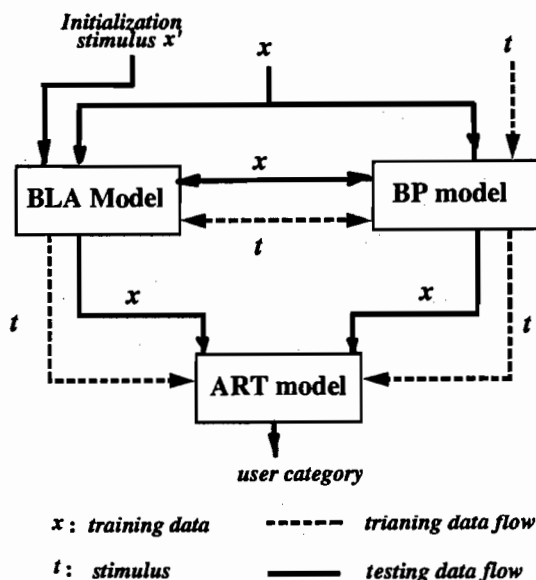


Fig. 2. Simulation of an integrated network environment.

cess can be conducted for either a single module or the whole system, which makes system adaptation simple and flexible. In addition, it has been suggested that if a neural network system is composed of several modules, each module can provide an insight about current system beliefs, which improves the system's ability to explain intermediate assertions in the reasoning process [3].

We used NeuralWorks Professional II/Plus [16] as the tool for creating networks and integrating them into this framework. This tool allows us to exchange input/output files between different network modules easily.

8. Discussion

The proposed approach is inspired by the neural networks' ability of pattern recognition and association. This ability is particularly useful for stereotyping users' domain knowledge in the following way:

- *Default reasoning and generalisation*: given an assumption about a user's domain knowledge which can be incomplete, the system is able to associate more assumptions to that user. This network ability simulates default reasoning in a very simple way. As discussed in Section 4.3, the ability to respond to novel inputs that bear some similarity to previously learned examples can be used for generalisation. Also, the neural network models can handle partial or erroneous cues without any ill-effect [15]. As discussed in Section 4.1, the conflict assumptions in input do not yield conflicting output, which is crucial for default reasoning [6].
- *Individualisation*: as discussed in Section 2, the modelling process is not confined by predefined stereotypes (i.e. sets of assumptions). The system models a user's domain knowledge at the assumption level rather than at the stereotype level, which means that the networks can specialise the user models better, because more combinations of assumptions can be created to stereotype individual users.
- *Knowledge elicitation*: conventional rule-based systems require extensive work on knowledge elicitation to set up rule bases, which tends to be a time-consuming and error-prone process [8,9]. The proposed approach only focuses on the extraction of causal relationships between each pair of assumptions, which is an easier task than establishing an entire rule base.
- *Reducing system overhead*: it is widely recognised that the implementation and maintenance of neural

networks is simpler than that of rule-based systems [8,9]. Furthermore, rule-based systems often involve either complicated conflict resolutions in default reasoning, or the belief value revision in evidential reasoning [6]. In contrast, the consistency can be easily maintained in neural network systems due to their ability to handle inconsistent or incomplete information. Modelling is more efficient because there is no need for a Truth Maintenance Sub-system (TMS) [4], which is required in a rule-based system.

9. Conclusion

This paper presents a study that utilises a set of neural networks as the knowledge base and a reasoning mechanism for modelling user profiles of domain knowledge. It organises all assumptions into associative memories in which the relationships among the assumptions are weighted under certain conditions. This approach views user modelling in terms of pattern recognition and pattern classification. Compared to rule-based systems or decision tree-based systems, it has shown several advantages, such as rapid default reasoning and generalisation, insensitivity to inconsistent input, pattern classification, and learning ability. Also, this approach is easier to implement and maintain. The knowledge elicitation process is simpler than the rule-based construction, because only the causal relationships are considered to initiate the modelling process. Further research is aimed at incorporating the task information into user models to provide a more comprehensive basis for system adaptation.

References

1. Norcio AF, Stanley J. Adaptive human-computer interfaces: A literature survey and perspective. *IEEE Trans System, Man and Cybernetics* 1989; 19(2): 399-408
2. Rich E. User modeling via stereotypes. *Cognitive Sci* 1979; 3: 329-354
3. Chen Q. Building intelligent agents in interface of design systems. *Proc 3rd Int Conf on Engineering Design and Automation*, Vancouver, Canada, August 1999
4. de Kleer J. An assumption-based TMS. *Artificial Intelligence* 1986; 28(2): 127-162
5. Ballim A, Wilks Y. Beliefs, stereotypes and dynamic agent modeling. *User Modeling and User-Adapted Interaction* 1991; 1(1): 33-66
6. Huang X, Mccalla GI, Greer JE, Neufeld E. Revising deductive knowledge and stereotype knowledge in a student model. *User Modeling and User-Adapted Interaction* 1991; 1(1): 87-117
7. Hayes-Roth F. Making intelligent systems adaptive. In: Vanlehn K (ed), *Architecture for Intelligence*, Lawrence Erlbaum, Hillsdale, NJ, 1991
8. Fu L. A neural network model for learning domain rules based on its activation function characteristics. *IEEE Trans on Neural Networks* 1998; 9(5): 787-795
9. Carpenter GA. Neural network models for pattern recognition and associative memory. *Neural Networks* 1989; 2: 243-257
10. Kosko B. Adaptive bidirectional associative memories. *Applied Optics* 1987; 26: 4947-4960.
11. Simpson PK. *Artificial Neural Systems: Foundation, Paradigms, Applications and Implementations*. Pergamon Press, 1990
12. Dent J, Gaddis T. *Guide to Unix Using Linux*. Course Technology, 2000
13. Chen Q, Norcio AF. Modeling a user's domain knowledge with neural networks. *Int J Human-Computer Interaction* 1997; 9(1): 25-40
14. Wilson M. Task models for knowledge elicitation. In: Diaper D (ed), *Knowledge Elicitation: Principles, Techniques and Applications*, Ellis Horwood, 1989
15. McClelland JL, Rumelhart DE. *Parallel Distributed Processing, Vol 2*. MIT Press, 1986
16. *Software Reference for Professional II/Plus and NeuralWorks Explorer*, NeuralWare, Inc. 1998