The Analysis of User Behaviour of a Network Management Training Tool using a Neural Network

Helen DONELAN and Colin PATTINSON
Computer Communications Research Group
Leeds Metropolitan University,
Leeds, LS6 3QS, UK

and

Dominic PALMER-BROWN
School of Computing and Technology,
University of East London,
Longbridge Road, Essex, RM8 2AS

ABSTRACT

A novel method for the analysis and interpretation of data that describes the interaction between trainee network managers and a network management training tool is presented. A simulation based approach is currently being used to train network managers, through the use of a simulated network. The motivation is to provide a tool for exposing trainees to a life like situation without disrupting a live network. The data logged by this system describes the detailed interaction between trainee network manager and simulated network. The work presented here provides an analysis of this interaction data that enables an assessment of the capabilities of the trainee network manager as well as an understanding of how the network management tasks are being approached. A neural network architecture is implemented in order to perform an exploratory data analysis of the interaction data. The neural network employs a novel form of continuous self-organisation to discover key features in the data and thus provide new insights into the learning and teaching strategies employed.

Keywords: Network Management, Simulation based training tool, Neural Networks, Data Analysis.

1.INTRODUCTION

The simulation based approach to the task of training network managers was originally presented in [1] and aims to provide a tool for exposing trainees to a life like situation, where both normal network operation and 'fault' scenarios can be simulated without disrupting a live network. The approach makes use of a production-standard network management platform, interacting with processes (model agents) representing network entities. This paper presents the analysis and interpretation of the interaction between trainee network managers and the network management training tool, using neural networks. The aim is to provide a new insight into the interaction data in order to support and complement learning and teaching activities and outcomes.

The simulation tool has been successfully used in the training of network managers at undergraduate and postgraduate level. Tasks are set such as exploration exercises to identify active components of a network and the control of simulated 'fault' conditions. The student needs to quickly establish how to approach a given task, which devices need to be interrogated, parameters of which networking protocol offer the most relevant information on network traffic and what parameters need to be monitored in order to obtain information on the status of the network quickly and efficiently. The student selects commands that to the best of their knowledge represent the most appropriate course of action required to manage the network and data describing their actions is automatically computer logged to an output file. It is this data that is under scrutiny here in order to assess the capabilities and effectiveness of the trainee network manager from how the tasks are being approached. The data includes a description of the commands issued by the student that request current values of, or set up processes to monitor, various parameters of the network. The commands that may be used can be divided into groups defined by the layer (or networking protocol) that they apply to, such as IP, TCP, UDP, ICMP or SNMP. Other information includes the node/device within the network that the command is directed at, any associated variables and a date and time stamp for each command.

2.NEURAL NETWORK APPROACH

The conventional method for assessing the students' behaviour is through a detailed visual examination of the output file created during a network management session. Through prior knowledge of the task in hand, an inspection of the devices located within the network and commands used, a picture of the accuracy and completeness of the suggested solution and whether it was achieved within an acceptable time limit is given. Whilst it can be determined whether the overall desired outcome was achieved it can be difficult to follow the logical sequence of events, particularly in a lengthy task and if many devices are involved. It is therefore difficult to determine approaches and methods being adopted. Applying neural networks to the user behaviour data presents an opportunity to find hidden patterns and establish relationships between variables that would not have necessarily been identified otherwise. This may offer new information about the interaction occurring between student and simulation platform.

An unsupervised technique is adopted as unsupervised classification or clustering requires no a priori knowledge about the data. Classes or categories are formed by the neural network according to attributes of the data and it is then possible to make an intelligent interpretation of the results by determining which properties the neural network has used in its classification [2-4].

The objectives to be acheived through the establishment of relationships identified by the neural network include:

(i) A more detailed examination of the approaches being undertaken to complete specific tasks, i.e. are the preferred (taught) methods being implemented or alternative less (or more) efficient methods?

(ii) To observe whether basic commands are being over-used in place of more specialized, directed commands.

(iii) To investigate whether commands that are expected to follow a sequence of events, do so. i.e. is information about the network previously obtained, being used to its maximum potential?

(iv) Do some functionalities of the simulator impede rather than advance the students' progress?

(v) Does the logical sequence of events demonstrate a basic understanding of the network structure?

In general, as well as being able to assess the students capabilities and effectiveness as network managers and highlight areas where knowledge is lacking, the results from this research also provide an opportunity to assess teaching methods and emphasize areas that may need further explanation.

There have been many projects involving NNs for user data analysis and pattern discovery. Zhang et al [5] used NNs for learning relations between textual data to aid the construction of hypertext computer assisted learning material.

3. THE NEURAL NETWORK

The P-ART has been selected to perform the classification of the user data. This is because it makes use of a novel learning algorithm that helps to classify all the data according to different attributes and yet allows a level of control over the properties of the categoreis being formed. These algorithms are summarised below and described in detail in [2-4].

Performance-guided ART (P-ART) Architecture
The P-ART architecture is illustrated in Fig. 1.

The P-ART network is a modular, multi-layered architecture and was originally presented in [2]. It comprises two P-ART modules and an external input that influences the learning technique and therefore the properties of the category groupings that are formed. The learning combines Learning Vector Quantisation with Adapaptive resonace Theory (ART). ART [6] is an unsupervised adaptive technique that maps *n*-dimensional input vectors into a number of output categories or classes based upon the input pattern's features. The network performs real-time learning and can form new output categories when a novel input pattern is presented.

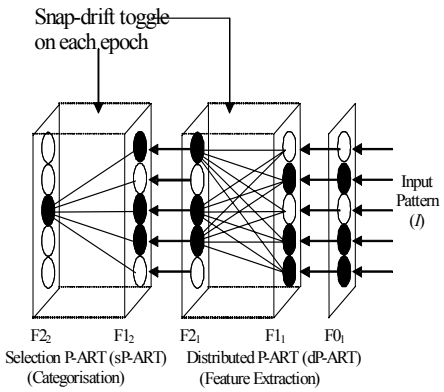

Figure 1: P-ART Network

The P-ART network is composed of 2 modules, a Distributed P-ART (dP-ART) network, and a Selection P-ART (sP-ART). The $F1_1 \leftrightarrow F2_1$ connections of the dP-ART network and $F1_2 \leftrightarrow F2_2$ of the sP-ART are interconnected through weighted bottom-up and top-down connections that can be modified during the learning stage. For clarity, only the connections from the F1 layer to the active (winning) F2 node in each P-ART module are shown. On presentation of an input pattern at the input layer $F0_1$, the dP-ART will learn to group the input patterns according to their general features using the novel learning principles of the snap-drift algorithm recently developed [2-4]. The new version used here is fully self-organising, and toggles between snap and drift learning modes on successive epochs (complete passes of the data).

The 'Snap-Drift' Algorithm
In an environment where new patterns are introduced over time, the learning utilises a novel snap-drift algorithm based on fast, convergent, minimalist learning (snap) and cautious learning (drift). Snap is based on a modified form of ART; and drift is based on Learning Vector Quantization (LVQ) [7]. In general terms, the snap-drift algorithm can be stated as:

$$w = \alpha(Fast\ \_\ Learning\ \_\ ART) + \sigma(LVQ) \tag{1}$$

In this paper, α and σ are toggled between (0,1) and (1,0) at the end of each epoch. The point of this is to perform two complementary forms of feature discovery within one system. The ART style learning acquires features characterized by the intersection of patterns, whereas LVQ performs clustering, discovering features that are averaged across patterns.

The Distributed P-ART (dP-ART) Learning
On presentation of an input pattern, the bottom-up activation is calculated. Then the D $F2_1$ nodes with the highest bottom-up activation are selected:

$$T_J = \max\{T_J \mid J = 1,2,....., M\} \tag{2}$$

D is set to 3 in this application. The three $F2_1$ nodes learn according to:

$$w_{Ji}^{(new)} = \alpha(I \cap w_{Ji}^{(old)}) + \sigma(w_{Ji}^{(old)} + \beta(I - w_{Ji}^{(old)})) \quad (3)$$

where w_{ji}= top-down weights vectors; I = binary input vectors, and β = the drift speed constant = 0.5. When α =1, w updates simply to:

$$w_{Ji}^{(new)} = (I \cap w_{Ji}^{(old)}) \quad (4)$$

This invokes fast minimalist learning, causing the top-down weights to reach their new asymptote on each input presentation:

$$w_J \rightarrow I \cap w_J^{(old)} \quad (5)$$

In contrast, when σ = 1:

$$w_{Ji}^{(new)} = (w_{Ji}^{(old)} + \beta(I - w_{Ji}^{(old)})) \quad (6)$$

This causes a simple form of clustering or LVQ at a speed determined by β. Overall the learning is a combination of the two forms of adaptation as the mode is toggled between snap and drift. The novel bottom-up learning of the P-ART is a normalised version of the top-down learning:

$$w_{Ji}^{(new)} = \frac{w_{iJ}^{(new)}}{|w_{iJ}^{(new)}|} \quad (7)$$

where $w_{iJ}^{(new)}$ = top-down weights of the network after learning.

4.METHODOLOGY

A methodology is developed and presented here that enables the application of a neural network to any 'interaction' or 'user behaviour' data so it may be implemented in other on-line teaching environments. The aim is to minimise the pre-and post-processing tasks that are required through the definition of a structured approach. In order to obtain an analysis of this kind of data through the use of a neural network (NN), there are several considerations that need to be given to the method and the stages of transformation the data must undergo before it is suitable as an input to a neural network. A methodology is developed and realised in the form of a set of procedures that embrace the following processes: (i) Pre-processing: conversion of the 'raw user data' into a form suitable for input to the neural network as the computer logged data contains both qualitative and quantitative data and therefore requires careful processing and encoding (ii) Selection of optimal neural network parameters, (iii) Post-processing: manipulation of the results in order to provide a novel and intelligent analysis. These stages, including considerations on the contextual aspect of the data and descriptions of how the data is encoded, are discussed in this paper and described in terms of the network management user behaviour being analysed. The methodology is summarised by way of a flow chart in Figure 2.

There are some fundamental issues regarding *collection* and *initial assessment* of the data that need to be addressed initially such as: How much data is available? Is there a constant output of data? Is there sufficient data to adequately train and test the network?

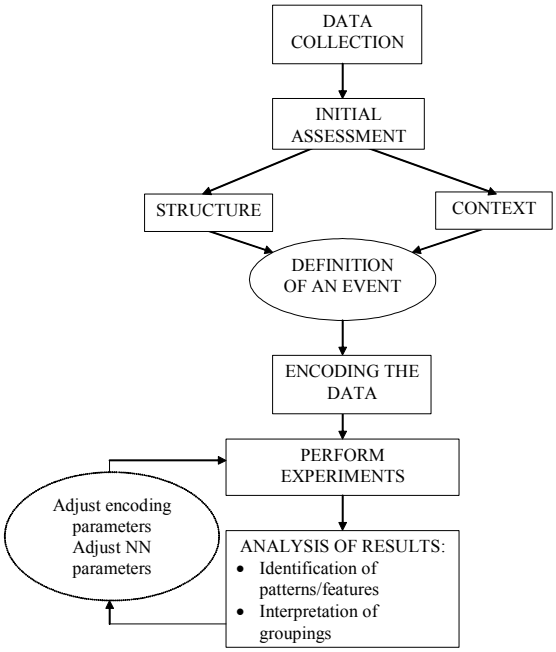

Figure 2: General Methodology

The performance of a NN is dependent on the training data. The training data must be representative of the task being learnt and tends to be chosen through trial and error before finding an acceptable training data set [8]. The production of the network management data is on-going but for the purposes of this paper a sample has been used to provide an initial evaluation of the data. Fifty-five datasets are used, where a single dataset represents a trainee undertaking a single network management task.

The next consideration in pre-processing the data is how to *structure* the encoded data in order to ensure all appropriate variables contribute towards the NN's decision making and is in a form suitable for input to a NN. The data is initially primary encoded into a list of consecutive events, where an *event* comprises all the information required to describe what is taking place at a given instant in time. The aim is to ensure that each event comprises the same amount of information in order to introduce some structure to the data.

The network management data is already in a format that lists events, it simply needs condensing into the relevant information. For the user behaviour data under scrutiny here, this takes the form: {*Action* ; *Node Description* ; *Duration*}. *Action* describes the command issued by the student, *Node Description* refers to which device the command relates to and whether it is active or inactive and *Duration* is the time taken to execute the current command taken from time of issue of the previous command. Determining the structure of an event is the first level of encoding and the resulting event is called the primary encoded data. This process reduces the feature space and simplifies the data therefore care must be taken to ensure relevant information is not excluded.

Another consideration is how to interpret the *context* of the data. Although the data is in a serial format, relationships between consecutive and subsequent events may be an important factor that needs to be considered. When the NN receives an input vector, it compares it to previously stored input patterns and then either puts it into the class that most closely matches it, or if no such class exists, creates a new one. Any patterns that are being identified within the data are so across input vectors. Therefore the length and quantity of data within each input vector is extremely important. The investigation requires several stages where each stage can be described in terms of the length of the input vector, which is a multiple of events (1xEvent, 2xEvent etc.). The first stage is simply concerned with the occurrence of individual events, i.e. an input vector is equal to 1 event. Whilst this provides information on the significance of each of the different events within a network management session, no information is afforded on the context of events. The second stage of investigations tackles the contextual aspect of the data. An input vector presented to the NN comprises 2 or more consecutive events. Consecutive input vectors comprise overlapping events as illustrated in the following equation. The final, binary encoded m-th input vector that is presented to the initial layer of the neural network, for the N-th stage of investigations, where N is the number of events per input vector (NxEvent) is represented by equation (8), where E_{m+n} is the $(m+n)^{th}$ event and $\bigcup$ represents the concatenation function.

$$I_m = \bigcup_{n=0}^{N-1} E_{m+n} \tag{8}$$

Encoding the Network Management Data

Once the structure of an event has been established, each component (*Action*, *Node Description* and *Duration*) that makes up the event can be individually coded and the coded components concatenated to form the overall input vector. It is necessary to know how many different values each of these components may take to facilitate the implementation of an appropriate coding scheme. Examples of the coding scheme for the *Action* component are illustrated in Table 1 in terms of a sample of the possible actions, their primary encoded form, and the final coded format ('O' represents binary '1' and 'X' binary '0'). The first segment of the codeword distinguishes between categories or layers the action belongs to, such as IP or TCP. The second segment distinguishes between the different actions within a category.

Table 1: Encoded *Action* Component of an Event

	Name (Primary Encoding)	**Secondary Encoding**	
1	System Info(SI)	OXXXXXXX	XXXXXXX
2	IF Status(IF St)	XOXXXXXX	OXXXXXX
3	IF Parameter(IF P)	XOXXXXXX	XOXXXXX
4	IF Usage(IF Usg)	XOXXXXXX	XXOXXXX
5	IF Error(IF Err)	XOXXXXXX	XXXOXXX
6	IF Quality(IF Q)	XOXXXXXX	XXXXOXX
7	IP Stats(IP Stat)	XXOXXXXX	XXXXXOX
….	……	…..	…..
22	Start Monitor(Smon)	XXXXXXXO	XXXXXOX
23	Help (Help)	XXXXXXXO	XXXXXXO

The second component of an event is the *Node Description*. Again, the codeword is segmented. The first bit of the codeword indicates whether the node is active (O) or inactive (X) and the remaining 14 bits are used to distinguish between the active nodes. Finally, the third component is the *Duration*. This is useful as it gives an indication of the time taken to execute a command. A coarse coding scheme [9] is used, where neighbouring sub-divisions are allocated codewords that differ from each other by 1 bit position. An example event in its original and encoded formats are illustrated below.

Original User Data: `System Information{NODE node53 node53 192.3.47.4 0 {}}Fri Apr 4 15:47:32 2003`
Primary Encoded Data: `{ SI ; 1/4 ;` t_0 `}`
Secondary Encoded Data:

{OXXXXXXXXXXXXXXOXXXOXXXXXXXXXXXXXX}
Action Node Description Duration

Neural Network Parameters

There are several parameters concerning the dimensions of the NN and the level of categorisation. The NN architecture has been presented in detail elsewhere [2-4]. In this paper, the vigilance parameter is set to 0.3, which in effect means that the criterion for allowing a node to respond and learn the input pattern is a 30% match. The number of input neurons to the dP-ART is determined by the length of the input vector. The number of output neurons for the dP-ART and therefore the input neurons the sP-ART, has been set at 500 which has proved large enough to avoid saturation of all the output neurons of the dP-ART. The number of output neurons of the sP-ART has been set at 150 in order to limit the number of output classes that can be formed to a manageable number and yet ensure that the majority (approximately 95%) of inputs are classified.

5.RESULTS

The P-ART is implemented in C++ and a total of 2700 input vectors are input to the NN at each stage of the investigation. The inputs are binary vectors, constructed using the encoding scheme detailed previously, and the output is a corresponding column of numbers indicating the output class selected for that input. An analysis is performed on the output of the neural network in order to see how the data has been classified. The results identify both commonly occurring combinations of events and other interesting, less common, sequences of events that provide insights into the students' behaviour. This is used to compare instances of good and bad practice and reveal patterns embedded within the data that may not have been recognized through other methods. Actions are referred to by number or primary encoded format. These and their basic functionalities within the network management simulator are summarised in Table 2 below.

Firstly, it is necessary to assess the number and size of the output classes that have been formed, i.e. how many output nodes of the NN are used and how many input vectors are assigned to each of these. The results are easily rearranged, grouped and manipulated in order to make different comparisons.

Table 2: Summary of Actions

	Primary Encoding	Description of functionality
1	SI	Display information of the system group
2	IF St	Display information of the interfaces
3	IF P	Display interface parameter like speed
4	IF Usg	Display interface statistics
5	IF Err	Display interface statistics
6	IF Q	Show error and discard rate for each interface
7	IP Stat	Display statistics and parameters of IP layer
8	IP A	Show IP addresses used by this device
9	IP R	Display routing table
10	IP ARP	Display other devices its been in contact with
11	TCP Stat	Display statistics and parameters of TCP layer
12	TCP C	Display status of existing TCP connections
13	UDP Stat	Display statistics and parameters of UDP layer
14	UDP L	Display status of existing USP listener
15	ICMP Stat	Display stats and parameters of ICMP layer
16	SNMP Stat	Display stats and parameters of SNMP layer
17	W	Walk through MIB tree and print object values
18	SetP	Set SNMP parameters
19	monV	Monitor an SNMP variable in a stripchart
20	Del	Delete the monitoring proceess
21	Lmon	Set up monitoring process
22	Smon	Start monitoring process
23	Help	List choice of actions

Results are reorganized and displayed by output node in order to visualise the input vectors clustered within each output class. The second stage is to establish which of the classes formed are major, significant and minority classes. Major classes are the most commonly used classes, significant classes are smaller but still populated sufficiently have an impact on conclusions, whereas minority classes are those that are only used once or twice and have little impact so are disregarded. Once the results are grouped by output class the key or dominant features of each group are established. A dominant feature is defined as a feature common to over 90% of the input vectors within that class. It may take a single value or a group of values, e.g. action 1 (SI) or actions 2-6 (all IF type actions). A class may be defined by a single dominant feature, such as an action, or by several, such as action and node. Where the dominant features are discussed the terminology {*action* ; *node* ; *duration*} is adopted. For example { 12 ; 1/- ; - } represents an output class where a majority of the input vectors within this class are action 12 (TCP C) implemented on an unspecific active node.

1xEvent

These results provide information on the significance of events. Table 3 summarises some of the classes that have an action or group of actions as a dominant feature along with their size (combined size where more than one class exists with the same feature).

Action 1 (SI) is by far the most frequently used action. This is expected as this is the conventional method by which to obtain standard information regarding the network devices. Action 21 (Lmon) is also a common action which is also expected as this enables the monitoring of interface (network card) loads and is a common networking requirement.

In respect to types of actions, it is the IP actions that are the most commonly applied, which implies a good use of commands. IP layer actions and action 12 (TCP C) reflect requirements to determine network topology through address structure and are encouraged methods of exploration.

Table 3: Dominant Features for 1xEvent Results

Action	Number of classes	Class size
1 (SI)	4	813
2 (IF St)	2	49
3 (IF P)	1	47
6 (IF Q)	1	10
8 (IP A)	2	153
9 (IP R)	1	169
10 (IP ARP)	3	181
12 (TCP C)	2	148
21 (Lmon)	1	164
17-23 (monitor actions)	5	146
2-6 (IF actions)	2	200
11-16 (TCP, UDP, ICMP and SNMP)	1	13

Due to the way the NN has grouped certain inputs it is possible to compare classes that feature actions directed at an inactive node with those directed at active nodes. For example, the four output classes associated with action 1 (SI) are summarised in terms of their dominant features and size in Table 4.

Table 4: Major Output Classes Featuring 'SI'

Output Class	Dominant Features	Class Size
5	{ 1 ; 1/9 ; 1-5 }	64
14	{ 1 ; 0/0 ; 1 }	215
16	{ 1 ; 1/- ; 1-7 }	455
28	{ 1 ; 0/0 ; - }	79

Classes 5 and 16 illustrate action 1 applied to an active node within the network and have a combined size of 519. Classes 14 and 28 illustrate the same action but applied to an inactive node, and have a combined size of 294. The latter two classes imply inefficient practice or limited knowledge of the network structure and make up 36% of the total number of occurrences of this command.

Similarly, the classes featuring action 8 (IP A), as an individual dominant feature are illustrated in Table 5. Those featuring this action applied to an inactive node (output class 47) make up 27% of the total.

Table 5: Output Classes Featuring (IP A)

Output Class	Dominant Features	Class Size
30	{ 8 ; 1/- ; - }	112
47	{ 8 ; 0/0 ; 1-4 }	41

However, for action 10 (`IP ARP`), as illustrated in Table 6, the percentage of cases this action is applied to an inactive node, is only 9%, illustrating a more efficient application of this particular command.

Table 6: 1xEvent Output Classes Featuring (`IP ARP`)

Output Class	Dominant Features	Class Size
2	{ 10 ; 1/1 ; 1-2 }	36
34	{ 10 ; 1/- ; 1-5 }	128
40	{ 10 ; 0/0 ; 1-4 }	17

2xEvent and 3xEvent

Whereas the 1xEvent results are useful to determine the frequency of specific events, the 2xEvent and 3xEvent results can be used to identify relationships between consecutive events.

A strong relationship exists between action 1 (SI), and the monitoring actions. Several output classes (combined size of 241) have been created that illustrate this action both preceding and following a monitoring action. Another interesting observation is the formation of several output classes that feature inactive nodes as dominant features in consecutive events. For the 2xEvent results, four output classes are formed that feature an inactive node in both events. One of these is a major class (size 105) and contains instances when the SI command is repeatedly implemented on an inactive node. This behaviour implies an ineffective use of the SI command, both due to its repetition and it being directed at an inactive node. Major classes where inactive nodes appear as dominant features in both events of an episode make up 7% of the overall major classes for the 2xEvent results. Extending this investigation to the 3xEvent results to determine how common it is that three consecutive events feature an inactive node, it is seen that four classes are formed. One is of a significant size (33) and again shows repeated use of the SI command on an inactive node.

As expected from the 1xEvent results, the SI command features most prominently. For the 2xEvent results it is a dominant feature of four out of the fourteen major classes (45%) and is a dominant feature of both events in three of these. For the 3xEvent results, many of the classes formed are not defined by multiple specific dominant features. This is because although the input vector length has increased the vigilance parameter has remained the same in order to encourage a more generalised clustering of the input patterns.

One interesting result is the output classes that highlight events that follow action 1 (SI) – i.e. what the trainee manager does once basic system information has been obtained. The most popular course of action following SI is a repetition of the same action (combined size 165). The network management simulator has a functionality that allows the selection of multiple nodes and the application of a single action to each of the nodes selected. Whilst this feature exists, it is not an efficient method for collating information on the network as redundant information is gathered and therefore has implications on the bandwidth required due to unnecessary network traffic being generated. The output classes discussed here, with the repetition of the command over 3 consecutive events, implies the use of this simulator feature, which in turn implies lack of consideration to the way in which the exploration of the network is conducted. In comparison, the preferred course of action to follow the SI action is the use of IP layer commands to provide a more thorough and yet directed interrogation of network devices. This does appear as a dominant feature, but less frequently than the repeated use of SI (combined size 49).

6.CONCLUSIONS AND FURTHER WORK

A novel method for the analysis of interactions between a trainee network manager and a network management training platform has been presented. The method has been used to uncover hidden patterns in user behaviour and highlight instances of good and bad practice, which has consequently lead to novel insights into the learning experiences of the trainee.

The overall project can be broken down into three stages, as illustrated in Figure 3.

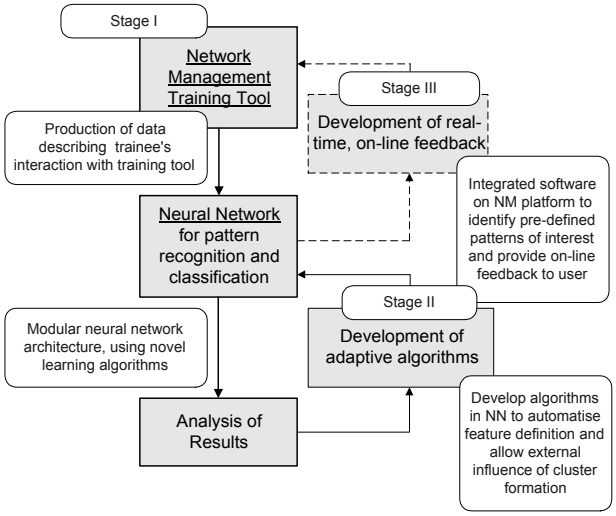

Figure 3: Future Stages of the Project

Stage I has been described here and planned future stages include development of integrated post-processing operations to enable automated definition of the dominant features of the clusters formed and the development of algorithms that allow analyst influence on the type of clusters being formed (Stage II). The eventual aim is to develop and integrate a real-time feedback system to the user of the Network Management Simulator (trainee), to provide on-line tutoring to support and enhance learning and teaching strategies offered by the training tool (Stage III). The feedback system would respond to real-time operation of the simulated network with advice in response to the approaches chosen by the trainee network manager.

7.REFERENCES

[1] C. Pattinson, "A Simulated Network Management Information Base", **Journal of Network and Computer Applications**, 23, 2000, pp.93-701.

[2] Lee, S.W.; D. Palmer-Brown; J.Tepper; C.Roadknight. 2002. "Performance-guided Neural Network for Rapidly Self-Organising Active Network Management." In **Soft Computing**

Systems: Design Management and Applications, A. Abraham et al (Eds.), IOS Press, 22-31.
[3] S.W**.** Lee, D. Palmer-Brown, J.A. Tepper, C.M. Roadknight, "Snap-Drift: Real-time, Performance-guided Learning", **Proceedings of the International Joint Conference on** Neural Networks (IJCNN'2003), Portland, Oregon, 20-24 July 2003.
[4] Lee, S. W., D. Palmer-Brown, et al. (2004). "Performance-Guided Neural Network for Rapidly Self-Organising Active Network Management**." Accepted for Neurocomputing**.
[5] Zhang, S.; H. Powell and D. Palmer-Brown 2001. "Methods for Concept Extraction using ANNs and Stemming Analysis and Their Portability Across Domains." **In Proceeding of The 2nd Workshop on Natural Language Processing and Neural Network** (Tokyo, Japan), 62 - 79.
[6] Grossberg, S. 1976. "Adaptive Pattern Classification and Universal Recoding. I. Parallel Development and Coding of Neural Feature Detectors," **Biol. Cybern**., Vol. 23, 121 - 134.
[7] Kohonen, T. 1990. "Improved versions of learning vector quantization', In **Proceedings of Int. Joint Conf. Neural Networks,** Vol. 1. 545-550.
[8] Callan, R. 1999. **The Essence of Neural Networks**. Prentice Hall, Europe.
[9] Eurich, C.W.; H. Schwegler, and R. Woesler, "Coarse Coding: Applications to the Visual System of Salamanders." **Biol. Cybern**., Vol. 77, 1997, pp. 41-47.