ML Tutor: An Application of Machine Learning Algorithms for an Adaptive Web-based Information System

A. Serengul Smith-Atakan, Ann Blandford

International Journal of Artificial Intelligence in Education (2002), to appear.

MLTutor: An Application of Machine Learning Algorithms for an Adaptive Web-based Information System

A. SERENGUL GUVEN SMITH[1] & ANN BLANDFORD[2]
[1] School of Computing Science, Middlesex University, Trent Park, Bramley Road, London, N14 4YZ, U.K.
E-mail: serengul1@mdx.ac.uk
URL: http://www.cs.mdx.ac.uk/staffpages/serengul/

[2] UCL Interaction Centre, University College London, 26 Bedford Way, London, WC1H 0AP, U.K.
E-mail: A.Blandford@ucl.ac.uk
URL: http://www.uclic.ucl.ac.uk/annb/

Abstract: One problem that commonly faces hypertext users, particularly in educational situations, is the difficulty of identifying pages of information most relevant to their current goals or interests. In this paper, we discuss the technical feasibility and the utility of applying machine learning algorithms to generate personalised adaptation on the basis of a user's browsing history in hypertext, without additional input from the user. In order to investigate the viability of this approach, we developed a Web-based information system called MLTutor. The design of MLTutor aims to remove the need for pre-defined user profiles and replace them with a dynamic user profile-building scheme in order to provide individual adaptation. In MLTutor, this adaptation is achieved by a combination of conceptual clustering and inductive machine learning algorithms. An evaluation technique that probes the detailed effectiveness of the adaptation is presented. The use of dynamic user profiles has been shown to be technically feasible; however, while a superficial evaluation indicates that it is educationally effective, the more thorough evaluation performed here shows that the positive results may be attributed to other causes. This demonstrates the need for thorough evaluation of adaptive hypertext systems.

INTRODUCTION

The idea of an intelligent personal assistant is not new. In the early 80's Michalski (1980) anticipated that individuals, in the expanding information society predicted by Bush (1945), would need intelligent personal assistants to cope with the overwhelming amounts of available information and the complexity of every day decision making. In addition, Michalski (1980) proposed that the knowledge and the function of such (computer based) assistants should be dynamic in order to adapt themselves to changing demands; in other words any such systems should be able to learn.

Traditionally, most research on computer based (machine) learning has dealt with the development of techniques for solving engineering problems and many of the systems developed have been tested on simplified artificial problems (Reich, 1994). Consequently, the machine learning research field has historically been very rich in terms of theoretical developments but lacks practical applications with direct links between theory and practice. This is particularly noticeable in the field of adaptive educational hypermedia, where many developments are restricted to small, hand-crafted systems and omit thorough evaluations of utility.

The research presented in this paper is an attempt to bridge the gap between theory and practice in the domain of WWW-based systems. The World Wide Web (WWW) is an excellent mechanism for the dissemination of information and a few WWW-based systems can provide support by adapting material to the needs of a user. A number of these systems employ machine learning techniques (e.g. NewsWeeder (Lang, 1995); Magi (Payne and Edwards, 1997); WebWatcher (Armstrong *et al*, 1995); LAW (Edwards *et al*, 1996); MANIC (Stern and Woolf, 2000)). However, the key challenge in such systems is to be able to capture an individual user's preferences and specific information needs and utilise this information to adapt the environment to the user. These systems do not meet the criterion for the ideal adaptive system as described by Brusilovsky (1996):

> " … while the user is simply working in an application system, the adaptation component watches what the user is doing, collects the data describing user's activity, processes these data to build the user model, then provides an adaptation. Unfortunately, such an ideal situation is very rarely met in adaptive hypermedia systems…".

With few exceptions, such as the referrer page based personalisation of PWW (Kushmerick *et al*, 2000), existing systems intrude by needing to interrogate users about their requirements or interests. This can be in the form of an initial registration process where user needs are assessed, or during interaction with the system in the form of questions or the provision of relevance feedback. The information gathered in this process is typically used to allocate a pre-defined stereotypical profile to a user, which is used to control adaptation (e.g. Pazzani *et al*, 1996). The development of such profiles is a very time consuming and laborious task and consequently most prototypes are restricted to one domain (Edwards *et al*, 1997). The knowledge bases of these systems are likely to be "hand-crafted" (Hohl *et al*, 1996). There is an overhead in generating such stereotypical profiles and a danger that an inappropriate one may be selected for a user.

The work reported here addresses the ideal as outlined by Brusilovsky: basing adaptation on browsing history. No additional feedback is required from a user and, by using machine learning techniques to dynamically analyse the browsing history, there are no requirements for pre-defined profiles to be constructed.

The prototype system implemented to test our approach is called MLTutor. Although it is not a tutoring system, it has been designed within an educational context to support task-oriented learning activity. MLTutor is a hypertext system that provides suggestions to the user on the basis of their recent browsing history, indicating pages that are relevant to the user's current area of interest. MLTutor is designed for use in an educational context on the World Wide Web (WWW). It aims to provide adaptation without the need for any prior knowledge regarding a user's background or interests.

Since work on MLTutor commenced, the idea of analysing information requests as a primary source of data has become more widespread. The use of clustering techniques to learn a user profile from a collection of documents has been investigated by Crabtree and Soltysiak (1998). In their system a user's word processing documents, emails and Web browsing activities are monitored to build an interest profile. Highest information bearing words are extracted from these documents to create vectors which are then clustered. These clusters are presented to users for feedback regarding relevance to their interests. The clustering technique used in this system is a top-down statistical approach. Crabtree and Soltysiak (1998) anticipate that the clusters produced by the system could be used for personalised information retrieval and filtering tasks. An overriding aim of this work was to reduce, as much as possible, the need for a user to provide input to the system. As with MLTutor the monitoring conducted by this system is unobtrusively performed, and keyword based page descriptions are used to cluster the accessed documents to identify user interests. The system aims to reduce the need for users to provide feedback but does not completely eliminate this.

The Syskill and Webert (Pazzani *et al*, 1996) system has similar objectives. The Syskill and Webert browsing assistant asks users to rate Web pages as interesting or not and learns a user profile which is used to suggest other pages of possible interest. One profile per topic is learned which is much more specific than a generic user profile could be. As with MLTutor this system

used machine learning techniques to build user profiles but relied on users rating documents, which are used as training data for the algorithms.

We present the system design, focusing on the design and test of the machine learning component, then describe the evaluation method employed and the results of that evaluation, and close with a discussion of the implications of this work and its relation to other studies.

SYSTEM DESIGN

MLTutor is a Web-based client server system which has been built with the intention of combining Internet technology with educational hypertext. The client component of the system incorporates the user interface and runs in a WWW browser. The client captures data which is transmitted to the server using Internet technology. The server component of the system is executed when requested by the client. The server contains a machine learning component (MLC) which analyses the received data and transmits results to the client. The server based MLC of MLTutor has been designed so that learning does not require any initial training. This is achieved by the novel integration of attribute-based clustering and inductive learning techniques. To enable this learning, an attribute database is embedded within the system; entries in the database describe pages in terms of presence or not of keywords within the hypertext content of the system. Within MLTutor the catalogue of keywords is based largely on hypertext anchors.

The prototype version of MLTutor contains four Web sites covering information on 'Environmental Science' issues. These four sites contain a total of 133 pages. There are no intrinsic links between the four sites. For this reason, bookmark links were provided to allow transition from one site to another. The Web technology employed in the design of the MLTutor prototype should ensure extendibility to the wider Internet environment, and to other topics.

The interface of MLTutor has been built in the Java language. The educational hypertext document is loaded into a Web browser frame following a logon procedure controlled by the MLTutor applet. Once the user has visited ten pages, the MLTutor applet passes the URL addresses of the last 10 hypertext pages visited by a user to the server. On the Web server, a CGI script starts running the Machine Learning Component (MLC) of the system. The output of this process produces a list of recommended hypertext pages and this information is sent to the MLTutor applet which consequently updates the information displayed in the suggestion frame as shown in figure 1.

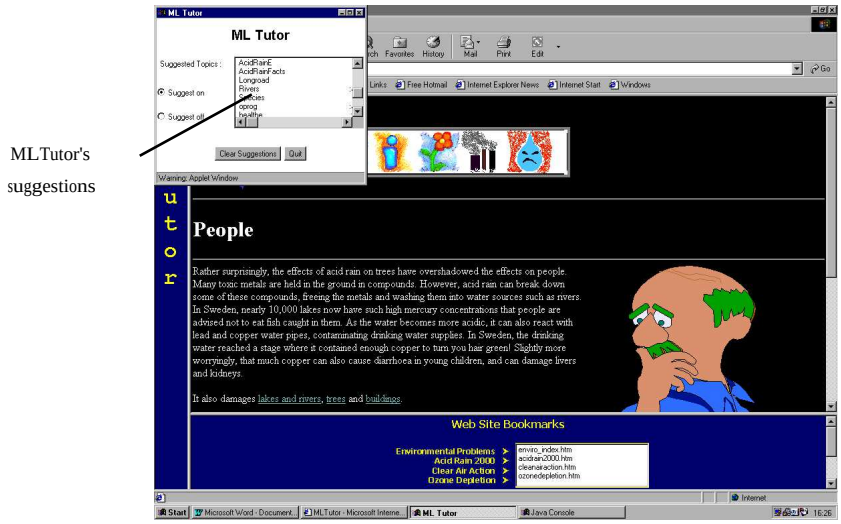

Figure 1: The MLTutor's suggestions, after 10 pages have been visited.

The user then has the choice of following links presented on the current page, selecting a different site (e.g. from the 'bookmark' list, or by typing a new URL) or following a link from the suggestion list.

From this point on, MLTutor presents updated suggestions on a regular basis. It processes the most recent ten pages at any time; the ten page limit was selected to minimise the browsing required before learning can take place while also ensuring that the learning algorithms are presented with a sufficient volume of data to allow effective learning. Alan Hutchinson, the developer of the conceptual clustering algorithm (Hutchinson, 1994) used in MLTutor, advised (personal communication) that ten pages is sufficient for the algorithm.

The phases of the learning process in the MLC are illustrated in figure 2 along with their relation to the attribute database.

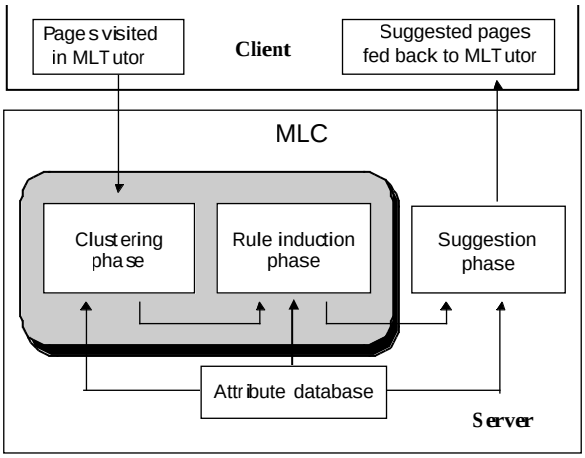

Figure 2: The machine learning component (MLC) design.

In the first phase of the learning process, clustering is utilised to find inherent patterns within the hypertext pages browsed by a user. To find these inherent classifications within the hypertext a simple conceptual clustering algorithm (Hutchinson, 1994) is used. Applying this method eliminates the need for initial "hand-crafted" stereotypical profiles or any additional input from a user.

In the second phase of learning, as shown in figure 2, a rule induction procedure is employed to generate rules. These rules describe the concept of cluster membership. This process works on the basis that the components of a cluster are in some way related to each other and as such are representative of a concept. Based on this, the content of a cluster is assumed to represent positive examples of a concept and anything beyond the boundary of the cluster is taken to be non-representative of that concept. The information concealed in clusters was initially interpreted by the ID3 algorithm (Quinlan, 1986) to create a set of rules which can be viewed as dynamically created user profiles. The rules generated by this phase are used to search for other hypertext pages within the original population of pages, classified as belonging to the concept.

The results of the learning process in the MLC are the pages classified as belonging to one of the learned concepts. These hypertext pages are related to those which have already been explored by a user and form the content of the *suggestion list* displayed within MLTutor as depicted in figure 3. The suggestion list has been developed as a *plug-in plug-out* component; with this feature disabled the MLTutor system is a simple hypertext browsing system.

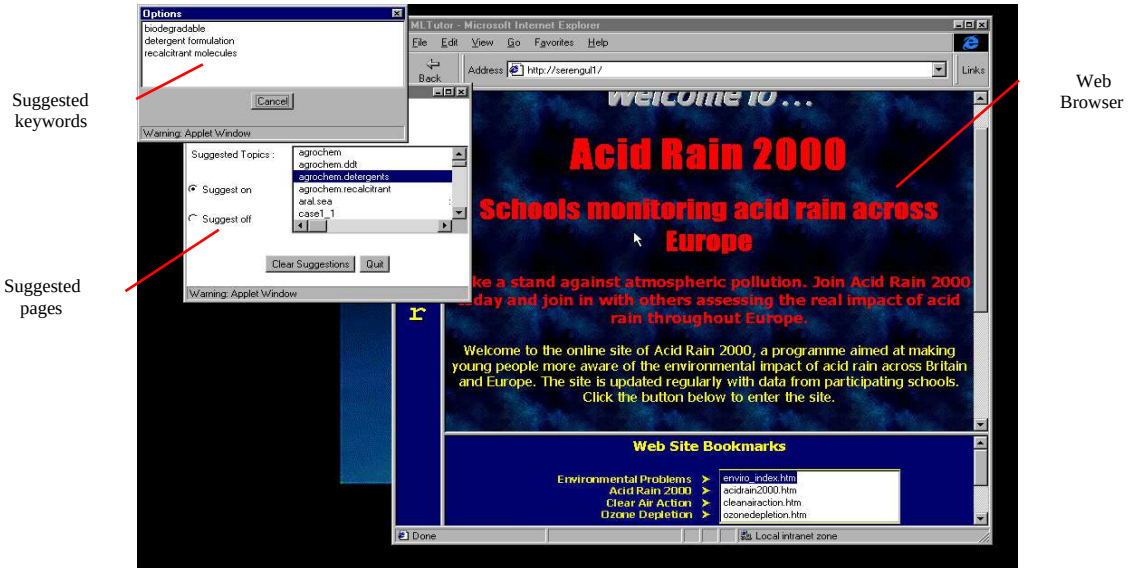

Figure 3: Selection of an entry from the suggestion list reveals the keyword pop-up window.

The clustering and rule induction processes within MLTutor make use of an *attribute database*. The first stage of attribute database construction is to assemble a list of keywords from the Web pages within the system. In the development of MLTutor a manual process was used to generate a catalogue of keywords which is largely based on embedded hypertext anchors. The long-term aim for the approach presented here is that it should support users accessing information from the Web, using meta-data about page content to index pages; various methods of automating the indexing have been considered, for example using the DC-Dot tool (UKOLN, 1998); although such an approach has not implemented, it is believed to be feasible as part of further work.

Having established a *keyword catalogue* from hyperlinks and other context related words and phrases from the pages contained within MLTutor, a binary vector was created for each page to show presence (explicit or implicit) or absence of each keyword on a page. In the binary representation the absence of a keyword on a page was coded with '0' and the presence with '1'. Each vector forms a single record of the *attribute database* as shown in figure 4.

page001 10000000100000010000000000000000000000000000000000000100000001000000000000000000000000.....acid.buildings
page002 10000000100000000100000001000000000000000000000000000000000000010010000000000000000.....acid.formation
page003 10000000100000000000000000000000000000000000000000000100000001000000000000000000000000.....acid.home
page004 10000000100000000000000000000000000000000000000010001000000000000000000000000000000000.....acid.how.big.problem
page005 00110000100010010000000000000000000000000000000000000100000001000000000000000000000000.....acid.lakes

Figure 4: The binary representation of five hypertext pages in the attribute database.

In the clustering process the attribute descriptions of hypertext pages visited by a user are obtained from the *attribute database* and, based on these attribute descriptions, the clustering process puts maximally similar pages into the same clusters.

The role of clustering in the MLC is illustrated in figure 5 below. Let the dark coloured circles in the first ellipse represent the hypertext pages visited by a user out of the available pages, then the second ellipse illustrates the results of clustering these hypertext pages. In this case the clustering has identified three distinct trends within the input and formed three clusters.

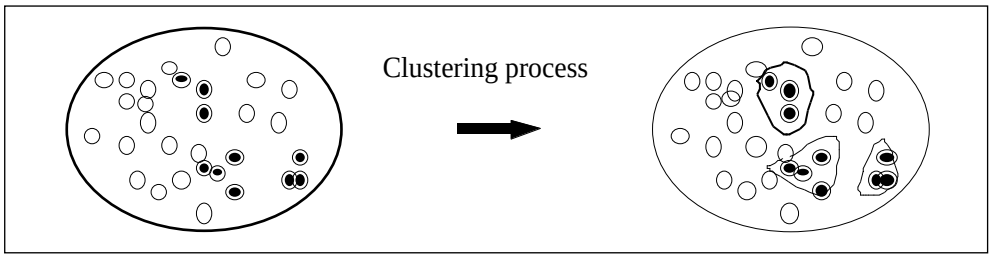

Figure 5 Input pages are partitioned into three clusters.

As a result of the clustering, the components of a cluster are in some way related to each other and as such are representative of a concept. Based on this the content of a cluster is assumed to represent positive examples of a concept and anything beyond the boundary of the cluster is taken to be non-representative of that concept.

The rule induction process generates attribute-based rules defining cluster membership. These rules are used to search the *attribute database* for other hypertext pages within the original population of pages classified by the rules as belonging to the concept. In figure 6 the grey zones extend the cluster boundaries and represent the hypertext pages suggested by the MLC as members of the same concept.

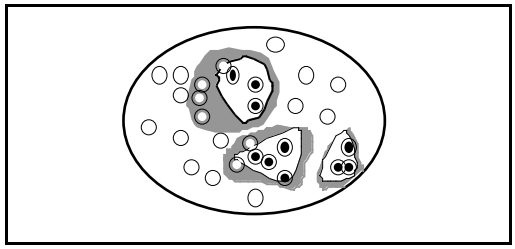

Figure 6: The rule induction process allows related pages to be suggested.

In MLTutor, the navigational steps taken by a user are the sole information source analysed to understand the user's motivation. This approach is the most unobtrusive way of gathering information about the user's interest during a task-oriented information search. Furthermore it eliminates the need for users to provide *relevance feedback* and eliminates the need for pre-defined stereotypical user profiles.

In summary, the machine learning approach proposed here facilitates a flexible, individualised approach to adaptation without the need for additional input from a user or pre-classification of users. This is achieved in the MLC by dynamically generating rules which hold generalised information about a user's current area of interest and are used to construct a *suggestion list.* This *suggestion list* is adaptive, reflecting the on-going browsing activity. While suggestions will only be related to previously browsed pages, as browsing activity moves into new areas the suggested items will begin to reflect this new focus. The suggested items will not be able to suggest topics that a user has given no indication that they are interested in.

The implementation of the *suggestion list* in MLTutor demonstrates the analysis performed by the MLC; however, the dynamically generated rules, based on the browsing activity, effectively form a profile indicating the user's current area of interest at a point in time. This profile is continually updated as further pages are accessed and new rules are created.

RULE INDUCTION STRATEGIES

Research in the field of symbolic machine learning (ML) has resulted in the development of a wide range of algorithms. Typically, learning in these algorithms is accomplished by searching

through a space of possible hypotheses to find an acceptable generalisation of a concept. However, ML algorithms vary in their goals, learning strategies, the knowledge representation languages they employ and the type of training data they use.

ML algorithms that do not require training are referred to as *unsupervised* algorithms e.g. *clustering* and *discovery* algorithms. Those that require training with a set of pre-classified examples are referred to as *supervised* learning algorithms e.g. *decision tree learning* such as *the Classification algorithm, ID3, C4.5* etc.

Decision tree algorithms accept a training set of attribute-based positive and negative instances of a concept which must all be presented before learning commences. Top down induction of decision trees is an approach to decision tree building in which classification starts from a root node and proceeds to generate sub trees until leaf nodes are created. It is possible to categorise conjunctive and disjunctive descriptions of concepts with decision trees, and *if-then* rules can easily be lifted from the trees.

Typically in decision tree learning, instances are represented by a fixed set of attributes; Mitchell (1997) states that decision tree learning is well suited when attributes take on a small number of disjoint values. This maps well to the attribute encoding scheme employed by MLTutor which is based on the presence or not of keywords in hypertext pages. This suggests that a decision tree building algorithm is a suitable candidate for the rule induction phase in the MLC. ID3 (Quinlan, 1986) was initially selected as being simple while providing the necessary functionality.

In order to test this a number of experiments were carried out. However, upon analysing the results of these experiments some concerns about the rule induction strategy used in the MLC were raised. Within the MLC of MLTutor the clustering of pages is used to detect inherent groupings within the hypertext pages browsed by a user, and the decision tree building ID3 algorithm is used to reveal information contained within the clusters.

For these experiments an *attribute database* with 123 keywords was used to test sets of browsed hypertext pages. Table 1 shows the two clusters created for these pages by the conceptual clustering algorithm.

Attributes that were common to all pages within the cluster are indicated in bold and referred to as *primary determinants* of cluster formation. Other attributes that are shared by only some of the pages are shown in italic and referred to as *secondary determinants* of cluster formation.

Attribute data base containing 123 keyword descriptions
Cluster 1 contains: page 5-7-11 ***common attributes*: 1 2 22**
Cluster 2 contains: page 2-3-4-9-10-2-32 ***common attributes*:** *31 84 99*

Table 1: Two clusters were created.

In this example the ID3 algorithm was applied to the data in the two clusters to induce decision rules for each cluster. Each cluster in turn was treated as positive training data for the rule induction process; the negative training data was the pages in the other cluster.

The ID3 algorithm uses an *information theoretic heuristic* to produce shallower trees by deciding the order in which to select attributes. The first stage in applying the information theoretic heuristic is to calculate the proportions of positive and negative training cases that are currently available at a node. In the case of the root node, this is all the cases in the training set. A value known as the *information needed for the node* is calculated using the following formula where p is the number of positive cases and n is the number of negative cases at the node.

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{n+p}\log_2\frac{n}{n+p}$$

All available attributes at the node are next considered. Available attributes are those that have not been used so far in the path from the root to the current node and so at the root all the attributes are available. For each of the available attributes in turn, for each value of the attribute, the number of the cases at the node which are positive and negative are counted. The formula above is used with these proportions to give a value called the *information needed for the attribute,* for example attribute A. A value is calculated for every value of attribute A.

This scaled sum is known as the E-score or *entropy* $E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i)$ and is subtracted from the information needed for the node to give an *expected information gain for attribute A.* The *information gained* by branching on A is therefore Gain (A) = I(p, n) – E(A).

The ID3 algorithm is summarised as follows:

Input: A training set
Output: decision trees

1. If all the instances are positive then terminate the process and return the decision tree.
2. If all the instances are negative then terminate the process and return the decision tree
3. Else

   Compute the *information gain* for all attributes, select an attribute with the *maximum information gain,* and create a root node for that attribute.

   Make a branch from the root for every value of the root attribute.

   Assign instances to branches.

   Recursively apply the procedure to each branch.

The concept descriptions produced by the ID3 algorithm are in the form of *if-then* rules which are used to select other pages to suggest to the user. The rule developed for cluster 1 of table 1 is depicted in figure 7.

IF				
	att-no	*22*	**is**	*1*
THEN	Yes			

Figure 7: The rule generated for cluster 1.

The pages within cluster 1 contain attributes 1, 2 and 22, and the ID3 algorithm needs only a single attribute, in this case 22, to create a rule describing the cluster. Consequently, only pages that contain the keyword represented by this attribute are eligible for suggestion using this rule. Within the context of the MLTutor this was not ideal as the suggestion rule failed to take into account all *primary determinants* of the cluster.

This is due to the fact that ID3 maintains a single current hypothesis as it searches through the space of the given concept. As a result, ID3 is incapable of representing alternative decision trees which are consistent with the available training data (Mitchell, 1997). In order to rectify this weakness the SG-1 algorithm was developed as an enhancement of ID3. Prior to using both the ID3 and the SG1 algorithms in the machine learning component of MLTutor, they were tested using sample training data, including data from Quinlan (Quinlan, 1983).

THE SG-1 RULE INDUCTION ALGORITHM

The ID3 algorithm was used as the basis for the rule induction phase in the initial prototype of MLTutor. The algorithm is used to generate classification rules, which are then used to provide adaptive navigational support. However, as described above, the ID3 algorithm was found not to be ideally suited to this application.

The idea of replacing ID3 with C4.5 in MLTutor was considered, however, the additional features introduced by C4.5 were not seen to offer significant benefit due to the nature of the data in MLTutor. Consequently, the SG-1 algorithm, an enhancement of ID3, was developed.

SG-1 is a decision tree building algorithm based on ID3. Within ID3, if there are several ways of building the decision tree at a given point, the algorithm takes into account only one of these. In contrast, SG-1 has the ability to produce multiple decision trees in this scenario and treats each possible alternative equally, building a subtree for each of them. Conceptually, the SG-1 algorithm can be visualised as building three-dimensional trees.

The SG-1 algorithm is as follows.

Input: A training set
Output: Multiple decision trees

1. If all the instances are positive or negative then terminate the process and return the decision tree.

2. Else
   Compute *information gain* for all attributes. For each attribute with *the maximum information gain* create a root node for that attribute.

   Make a branch from each root for every value of the root attribute.

   Assign instances to branches.

   Recursively apply the procedure to each branch.

The consequence of the ID3 enhancement is described in the following example. Suppose the information theoretic heuristic adds attribute A to the decision tree as the root as shown in figure 8. If there are two values for attribute A, two branches are added to the tree and the next attribute is selected.

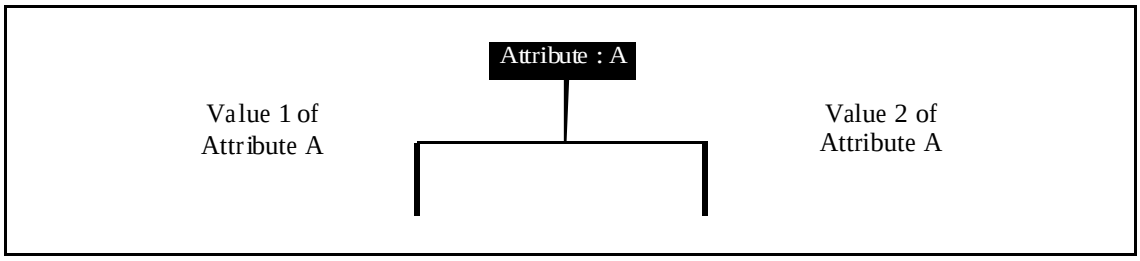

Figure 8: Two values for attribute A.

The information theoretic heuristic is again used to select the next attribute. If two attributes have the same maximum information gain, say B and C, the ID3 algorithm does not indicate which of these eligible attributes to select. The SG-1 algorithm builds a subtree for each of these attributes. The first subtree as seen in figure 9 is built when considering attribute B.

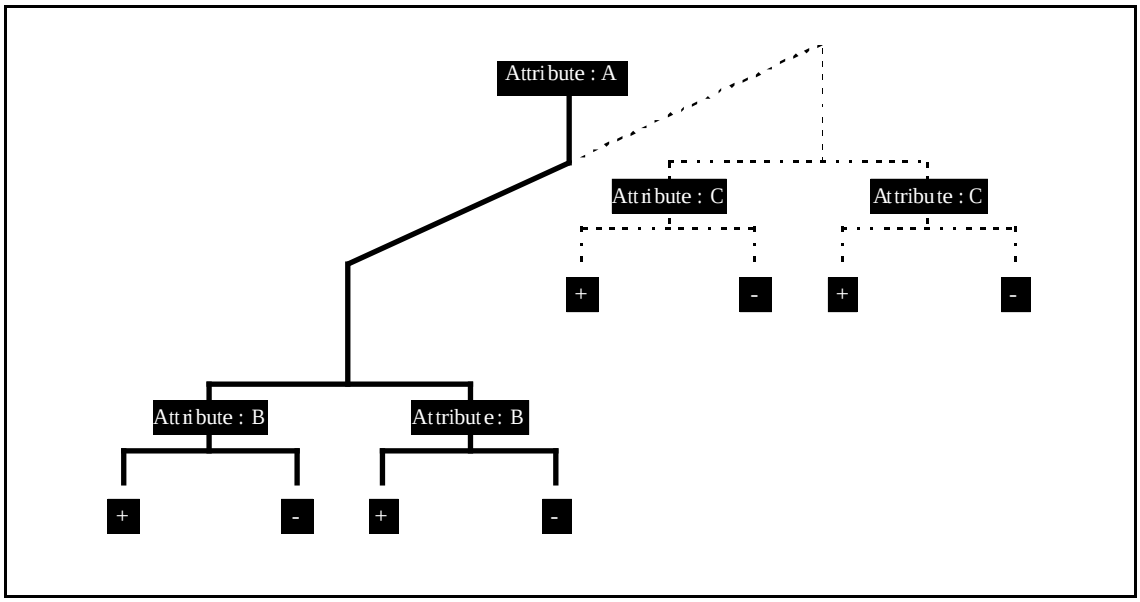

Figure 9: Multiple attribute eligibility, in this case attribute B is added to the tree.

Similarly, when the SG-1 algorithm considers attribute C, the second eligible attribute, a second subtree is constructed.

Decision rules from the SG-1 tree are created as for ID3. However, the additional dimension introduced by SG-1 potentially results in more decision rules being created as SG-1 maintains multiple concept descriptions.

In the example above, four rules for positive classifications are generated and similarly four rules for negative classifications. In this case the standard ID3 algorithm would have produced two negative and two positive classification rules only. The additional rules produced by SG-1 take into account alternative *primary determinants* from the cluster formation process. By using a disjunction of these rules to search for suggestions, the list of suggestions generated is more representative of the concept the cluster covers.

The SG1 algorithm was implemented and tested with the conceptual clustering algorithm, and found to perform satisfactorily, overcoming the shortcomings of ID3 identified in this context. This was therefore implemented with MLTutor, which was then evaluated using a battery of techniques to assess different aspects of effectiveness.

EVALUATION

In order to evaluate the effectiveness of the MLC of MLTutor, an empirical study was conducted. The evaluation aimed to assess the feasibility and also the utility of using machine learning techniques for the analysis of an individual user's navigational pattern.

Evaluating a system is an intricate task and, as noted by Hook (1997), it becomes more difficult if the system is adaptive. In recent years, although adaptive hypertext research has produced promising results (in terms of assisting personalised information gathering in an educational context) as reported by Brusilovsky (1996), a weak point of this research field is the lack of comprehensive empirical studies to measure the usefulness of adaptation within such systems and between such systems. One reason for this is that there is no standard or agreed evaluation framework for measuring the value and the effectiveness of adaptation yielded by adaptive systems.

One typical approach to determining the effectiveness of adaptation has been to compare the performance of an adaptive system against a version of the system with adaptation disabled. Hook (1997) states that adaptivity should preferably be an inherent part of a system, and so if it is removed from the system, the system may not be fully functional.

In MLTutor, the suggestion list is the adaptive component of the system. When the suggestion list feature is disabled, the MLTutor system becomes a simple hypertext browsing system. Consequently, it is feasible to create a non-adaptive version of MLTutor for comparison purposes. However, we considered additional questions about the details of the adaptation:

- If the previous ten pages form (typically) three or four clusters, which is the one the user is most likely to be interested in? The most likely candidates are the cluster containing the most recently visited page and the one containing most of the recently visited pages. To address this, two variants of the MLC were implemented, as described below.
- For a static web structure, it is possible to pre-classify the pages into clusters; while this would not be possible for the total web, we investigated the relative merits of pre-classifying against dynamic clustering based on the user's browsing patterns.

These questions were investigated within one study by comparing the performance of MLTutor versions containing four variants of the MLC. The four variants were compared with each other and against a non-adaptive control version.

Cluster selection strategies in MLC

The four versions of MLTutor were constructed with alternative cluster selection strategies as outlined below. The versions that employ pre-clustering do not use the conceptual clustering algorithm dynamically or the SG-1 algorithm to generate suggestions.

MLTutor Version 1

Rule induction based on the cluster containing the latest page visited: Within this version of MLTutor the data passed to the MLC of the system is clustered dynamically based on the most recent ten pages visited. The pages within the cluster that contains the latest page visited by the user are treated as positive training data for the SG-1 rule induction process. The negative training data for the rule induction process are the pages passed to the MLC which are not within the cluster containing the latest page visited.

MLTutor Version 2

Rule induction based on the most heavily weighted cluster: In this version of MLTutor the data passed to the MLC of the system is assigned a weight based on how recently the page was visited by the user. The most recently visited page is given the highest weight and the least recently visited page is given the lowest weight. A total weight for each cluster generated by the clustering process is calculated by summing the individual weights of the pages that formed the cluster. This strategy aims to reduce the impact of the most recent page being a temporary diversion. The pages within the heaviest cluster are treated as positive training data for the SG-1 rule induction process. The negative training data for the rule induction process are the other pages passed to the MLC which are not within the heaviest cluster.

MLTutor Version 3

Pre-clustering using the latest page: Within this version of MLTutor the complete set of pages available within the system were clustered beforehand and these clusters stored within the system. The stored cluster which contains the last page visited by the user is selected and the other pages within this cluster, excluding those in the input data passed to the MLC, are selected for suggestion. This cluster selection strategy would not be suitable for a general domain as the space of possible pages is not predefined.

MLTutor Version 4

Pre-clustering using weights: Within this version of MLTutor the complete set of pages available within the system were clustered beforehand and these clusters stored within the system. The data passed to the MLC of the system is assigned a weight based on how recently the page was visited by the user, as per version 2.

Experimental set-up

The evaluation study aimed to compare the adaptive MLTutor against a non-adaptive hypertext (the same web pages without the suggestion list), and also to compare the alternative adaptation strategies outlined above. We were interested in users' perceptions of the different versions, in whether the adaptive component had a significant effect on their task performance (in terms of their ability to perform well on a teacher-defined information task) and in whether the adaptation component had a great influence on user behaviour with the system.

Thirty people were recruited to participate in the empirical study. As many as possible (about 50%) were environmental science students; the remainder were computer scientists. Subjects were balanced across versions of MLTutor (so that, as far as possible, equal numbers of environmental and computer scientists used each version).

An instruction sheet illustrating the MLTutor logon procedure and use of the navigational tools available within the system was prepared. A version of the instructions, with descriptions of the adaptive features removed, was also prepared for users of the control version.

An expert on environmental science set 13 questions based on the information covered in the MLTutor system, and also provided model answers indicating the hypertext pages which contained the answers. Answers to some of these questions specifically required several pages to be visited. A marking scheme for answers was established.

The participants were asked to complete the tasks set by the expert using the allocated variant of MLTutor. Following this, they were asked to complete a short questionnaire on their views of the system they had used.

Raw data collection

Data for the empirical analysis was captured in several forms:

- Answer Sheets: In the experiment, participants were asked to answer 13 questions while browsing the Web documents within MLTutor and provide their answers in written form. The written answers of the participants were assessed against the model answers.
- User feedback questionnaire: In order to gather feedback on various aspects of the system a user feedback form was prepared consisting of two sections. In the first section, six questions asked for feedback on a scale; in the second section, the participants were given the opportunity to comment on various features of the system and provide any other feedback.
- Log files: During the experiment each participant's interaction with MLTutor was recorded in a log file. The MLTutor log files were used to identify the paths taken through the hypertext to reach pages containing the answers to the set tasks. Log files are the most effective method of unobtrusively determining how often adaptive features are being used and to trace interaction with the system.

Measures

The following data was extracted or calculated for the empirical analysis:

- Participant task results: These were used to see whether better performance was achieved using an adaptive MLTutor version, and if so which one, compared to the non-adaptive version. Each answer was marked using the marking scheme devised by the domain expert.

- Time spent to complete tasks: Calculated from log files, this data was used to see whether total task completion time was reduced by any MLTutor version.
- Total link usage: Extracted from log files, this data was used to see whether total link usage was reduced for any MLTutor version, suggesting more purposeful browsing.
- Total suggestion list link usage: Extracted from log files, this data was used to see whether suggestion list link usage was higher for any MLTutor version, suggesting more focused suggestions.
- Total site transition link usage: Extracted from log files, this data was used to see whether site transition via bookmark link usage was reduced for adaptive MLTutor versions compared to non-adaptive.
- Navigation paths: Data within the log files was analysed to determine how participants reached pages containing answers to the tasks – directly or via the suggestion list – and whether tasks were answered correctly from these pages.

RESULTS

Analysis of the participants' answer sheets

In order to proceed with the answer sheet analysis the participants' written answers were assessed against the expert recommended model answers. The participants' scores were analysed based on usage of adaptive and non-adaptive versions of MLTutor. Means and standard deviations based on this evaluation criterion are summarised below (table 2).

	Non-Adaptive MLTutor	**Adaptive MLTutor**
Mean	**17.9**	**19.6**
Stdev	4.2	4.2

Table 2: Summary of means and standard deviations of scores.

These results show that scores using an adaptive version are slightly higher than for the non-adaptive version of MLTutor. Further tests for statistical significance could not be conducted due to inadequate sample sizes.

In the next phase of evaluation the participants' scores were analysed in terms of the MLTutor version used. Means and standard deviations based on this evaluation criterion are summarised below (table 3).

	Version 1	**Version 2**	**Version 3**	**Version 4**
Mean	**21.4**	**20.7**	**20.2**	**16**
Stdev	3.3	2.2	4.9	4.7

Table 3: Summary of means and standard deviations of scores for adaptive versions.

The results indicate that the highest mean score was attained by users of MLTutor version 1, which was an adaptive version, and the lowest mean score was attained by users of MLTutor version 4 –again, an adaptive version. The second lowest mean score was attained by users of the non-adaptive MLTutor version.

Standard deviations for these results are quite high because each version was tested by participants with differing backgrounds (environmental and computer scientists), which resulted in large variations in test scores for each version.

In summary, these results only partly support the hypothesis that an adaptive version of MLTutor results in higher scores compared to the non-adaptive version. Although the results are

suggestive, the number of participants who tested the variants of MLTutor is too small to apply any further parametric statistical analysis.

Analysis of feedback questionnaires

The participants in the evaluation were asked to complete a feedback questionnaire after using MLTutor to answer the set tasks. Here, we focus on the qualitative (freeform) responses.

For each free form question on the feedback questionnaire, the comments of participants were examined with a view to identifying common themes and significant comments. The common themes identified from comments made by users of the non-adaptive MLTutor were as follows:

- Unstructured information
- Not a tutoring system
- There is no inter-link between sites

The common themes identified from comments made by users of the adaptive variants of MLTutor were as follows:

- Easy to use
- Site transition links are useful
- Some suggestions are good, but there should be more useful suggestions
- Suggests already visited pages
- Time delay before useful suggestions appear
- The entries in suggestion list were very difficult to understand
- A layman may need further background knowledge
- Provides a short-cut to relevant pages
- Made aware of links which were not apparent
- Offers different forms of navigational aid

The comments made regarding the non-adaptive version of MLTutor, a plain HTML browsing system, tend to support the decision to develop the adaptive MLTutor. It was suggested by a non-computer literate user that a layman may need further background information than that provided by this implementation of MLTutor.

In many cases, information on the WWW is unstructured and the adaptive features of MLTutor aim to overcome this by providing links between sites which are otherwise unconnected. Users liked the seamless integration of web-sites with a common interface and, as noted by participants, the adaptation provided by MLTutor gave them additional choice while searching for information.

In addition, participants found the suggestion list entries a convenient shortcut to relevant pages that were otherwise not apparent or readily accessible. Although it was not heavily used (see below), the alternative form of navigation introduced by the suggestion list seemed to be appreciated.

Analysis of log files

The participants' browsing activity were recorded in a log file in the background during the experiment. The usage of three types of links and the task completion time were assessed from these log files. The types of links examined are built-in links, suggestion list links and site transition links via bookmarks.

A quantitative data analysis was applied to this data. The primary objective of this particular analysis is to investigate relative performance of using adaptive MLTutor compared

with non-adaptive MLTutor. Means and standard deviations based on this evaluation criteria are summarised below (table 4).

Version	Mean	Time spent	Links followed	Sites transitions	Suggestions visited
0	**Mean**	**73.2**	**111.2**	**9.3**	**Facility not available**
	Stdev	24.6	49.2	8.1	
1	**Mean**	**62.2**	**85.7**	**7.7**	**4.3**
	Stdev	17.8	28.6	4.7	4.6
2	**Mean**	**53.7**	**82.8**	**8.2**	**4**
	Stdev	18.4	55.1	5.4	4.7
3	**Mean**	**53.2**	**85**	**5.2**	**2.8**
	Stdev	23.1	34.1	1.0	2.6
4	**Mean**	**56.3**	**71.3**	**7.3**	**6.5**
	Stdev	14.6	16.8	7.6	5.4

Table 4: Mean and standard deviation analysis of evaluation criteria.

The results indicate that the mean total task completion time was reduced for users of the adaptive versions, as was the total number of links followed. Although the suggestion list was used relatively infrequently by most users, it raised awareness of alternatives and helped make interactions more efficient. In this way, it appears to have had an indirect beneficial effect.

Cross analysis of log file and participant scores

In addition to the analysis described above, further in-depth analysis was conducted in order to investigate the use of built-in and suggestion-list links. The objective of this analysis was to determine how frequently built-in links and MLTutor system suggestions were followed by the participants and utilised to complete the given tasks.

For this analysis, four categories of link usage were established as follows:

X1: If a user accessed the expert recommended page by a built-in link and answered the question correctly when no suggestion link was available.

X2: If a user accessed the expert recommended page either by a built-in link or by a suggestion list link and answered the question correctly when suggestion link was available.

O1: If a user accessed the expert recommended page by a built-in link and failed to answer the question correctly.

O2: If a user accessed the expert recommended page either by a built-in link or by a suggestion list link and failed to answer the question correctly.

Ideally, six categories would have been preferred allowing built-in and suggestion list links to be analysed individually, but the data available from log files was not sufficient to allow this and so usage of these link types were merged to form categories X2 and O2.

In order to assist with this analysis, a *link usage analysis form* was created (see table 5). Part 1 of the form contains details of the participant to whom the form refers. Part 2 of the form gives a count of the number of times a suggestion list link was used to access the pages listed in part 4. Part 3 of the form contains counts of the number of times a built-in link was used to access the page numbers in part 4 of the form. Part 4 of the form lists the pages which contain the expert recommended model answers. In part 5 of the form there is a row for each of the set tasks. Within each of these rows, the pages containing the model answers to the question the row corresponds to, are highlighted in dark grey. For example pages 63 and 127 are the recommended pages to answer question 4.

This form is populated with data from various sources. The log file of each participant was analysed in conjunction with the answer sheet completed by the participant. Each answer was rated in terms of the four categories defined above and transferred to part 5 of the form. Data to complete part 2 and part 3 of the form was also extracted from log files.

1	**Participant** Group: Version:																														
2	Suggestion-list Links																														
3	Built-in Links																														
4	Expert rec. Links	**2**	**5**	**12**	**61**	**62**	**63**	**74**	**77**	**87**	**88**	**94**	**95**	**101**	**104**	**109**	**110**	**112**	**114**	**115**	**116**	**119**	**121**	**122**	**123**	**126**	**127**	**128**	**129**	**130**	**132**	**133**
5	**Question 1**		■					■	■	■		■	■		■	■	■															
	Question 2	■		■							■	■		■																	
	Question 3					■	■											■								■						
	Question 4						■																				■					
	Question 5																									■	■	■				
	Question 6																									■	■					■
	Question 7																						■			■			■	■	■	
	Question 8				■													■			■											
	Question 9																		■													
	Question 10				■														■													
	Question 11																		■	■	■											
	Question 12																					■										
	Question 13					■																		■	■							

Table 5: A link usage analysis form.

From an examination of the data gathered during the experiment, it became apparent that the pages containing answers to the questions had been accessed via various types of links i.e. suggestion list and built-in. Furthermore, the pages containing the answers had been repeatedly accessed, which had not been anticipated prior to the experiment. It was thus not clear which type of link had been used to answer the set tasks. This highlights a flaw in the design of the experiment, as the time the tasks were answered was not time stamped or audited. This would not be feasible with the paper-based format of the task completion which was used in this experiment. An on-line exercise, as suggested by one of the participants on the feedback questionnaire would be a solution to this problem.

The total occurrences for the categories X1 and X2 were summed from the *link usage analysis forms* and transferred to table 6.

	1st Participant	**2nd Participant**	**3rd Participant**	**4th Participant**	**5th Participant**	**6th Participant**
Version 1	X1= 13 X2= 0	X1= 7 X2= 3	X1= 6 X2= 3	X1= 10 X2= 1	X1= 15 X2= 0	X1= 18 X2= 0
Version 2	X1= 12 X2= 3	X1= 6 X2= 0	X1= 7 X2= 2	X1= 9 X2= 0	X1= 7 X2= 0	X1= 10 X2= 0
Version 3	X1= 8 X2= 0	X1= 10 X2= 3	X1= 12 X2= 1	X1= 11 X2= 3	X1= 10 X2= 0	X1= 11 X2= 0
Version 4	X1= 4 X2= 0	X1= 7 X2= 3	X1= 2 X2= 7	X1= 15 X2= 0	X1= 3 X2= 8	X1= 6 X2= 2

Table 6: Summary of the X1 and X2 categories in the *link usage analysis form.*

Within the population of participants, across the adaptive variants of MLTutor, the total occurrences of categories X1 and X2 are equal to 219 and 39 respectively.

The percentage of correct answers where the answer page has been visited by built-in links only is $\frac{X1}{X1+X2} = \frac{219}{258} = 85$ and the percentage of correct answers where the answer page has been visited using only suggestion list links is less than $\frac{X2}{X1+X2} = \frac{39}{258} = 15.$ These results indicate that the preferred navigational method for completing tasks was to follow available built-in links. Although the results of the statistical analysis are indicative of the benefits of adaptivity, the log file and participant score cross analysis contradicts this evidence. The simple quantitative evaluation criteria used in the analysis, widely used to measure the effectiveness and efficiency of adaptation in adaptive hypermedia research, on their own, fail to reflect the actual usefulness of the adaptation. The results indicate that the use of an adaptive feature does not prove anything unless related to improved performance in some way, and that superficial measures of performance may be misleading. The enhanced results alone may simply be the consequence of an alternative interface and not directly a consequence of any adaptation. This issue needs further investigation.

Analysis of the MLC

Much of the adverse user feedback on the suggestion list focused on the existence of irrelevant suggestions. Since the clustering phase of the MLC is largely responsible for the quality of the suggestions made, a further technical analysis of MLTutor was conducted, focusing on the performance of clustering within MLTutor. Two aspects of the clustering algorithm used were investigated: sort step sensitivity and the 'bin' cluster effect.

Sort step sensitivity

The conceptual clustering algorithm (Hutchinson, 1994) employed by MLTutor contains a sort step which orders pairs of pages based on the distance between them. If two pairs of pages are equally far apart they can legitimately appear in any order following the sort. The order of the pairs of pages has a bearing on the clusters produced by the algorithm. The decision to use integer values to encode the page attribute descriptions and the metric used to measure the distance between pages leads to the likelihood of there being a high incidence of equally distant pages. The impact of this in terms of the resultant clustering in MLTutor is hard to determine from the available information but could have had an impact on the clustering and ultimately the suggestions made by MLTutor.

The 'bin' cluster

For the implementation of pre-clustering in versions 3 and 4 of MLTutor, all 133 pages available within the system were clustered applying the same algorithm used in the dynamic variants of MLTutor.

Running the clustering algorithm on 133 pages initially generated 10 clusters; however the 10^{th} cluster contained a large number of pages. The clustering algorithm was re-applied to this data. This second phase of clustering produced three more clusters and yet again another quite large (final) cluster. This re-clustering process was repeated until no further subdivisions could be achieved. Having repeated the clustering process six times, 16 clusters were created. The 16^{th} cluster was still very large.

During the pre-clustering process it was observed that each re-clustering attempt produced one heterogeneous cluster – which in each case appeared to be the largest cluster – that contained a combination of items which seemed to be semantically unrelated. This paper refers to this last and largest cluster as the 'bin' cluster. The 'bin' cluster has implications for both dynamic clustering and pre-clustered versions of MLTutor and the effect was observed in the users' log files.

In the case of dynamic clustering, if the cluster selected for inducing rules is the 'bin' cluster then it is very likely that any suggestions produced for this cluster would be unfocused.

This problem was not appreciated when the MLTutor system was initially tested, and now needs to be addressed as a matter of priority.

SUMMARY OF ANALYSIS

The aim of the comparative study conducted on MLTutor was to determine whether users of the adaptive versions of the system were able to perform tasks more successfully than users of the non-adaptive version and, if so, whether any adaptive version was more successful than others. The principal measure used in the evaluation was the answers to questions completed by participants while using a version of MLTutor. Additionally, link usage and time taken to complete the exercises were used.

In terms of test scores achieved by the participants, the users of the adaptive versions scored on average higher than the users of the non-adaptive version. Within the versions themselves there was no clear favourite: users of three of the adaptive versions scored on average better than users of the non-adaptive version, while users of one of the adaptive versions scored particularly poorly. While it might be tempting to re-run the empirical study with larger user groups, this has not been done because qualitative and analytical results have highlighted areas where further development should be conducted before conducting further empirical evaluation.

Findings of the cross analysis indicate that there are a number of weaknesses related to the evaluation methods used in the field of adaptive hypermedia. These are that:

- The use of quantitative measures, on their own, may fail to reflect the actual benefits of adaptation.
- The use of an adaptive feature may not prove anything unless related to improved performance in some way.

In terms of MLTutor, although the results of the statistical analysis are indicative of the adaptivity having an effect on users, the *log file and participant score cross analysis* indicates that the adaptivity available within the system has hardly been used. By simply measuring differences in performance between adaptive and non-adaptive versions of a system, a fundamental assumption that the difference is due to the adaptivity is made. However, the cross analysis conducted as part of this research suggests that this assumption may not be valid.

Beyond the quantitative data gathered during the experiment, qualitative data was also collected in the form of feedback comments from the participants. Not all feedback from the participants who took part in the evaluation was positive. A number of enhancements to MLTutor were suggested by participants as follows:

- Already visited pages can be re-suggested without indicating that they have already been visited. There is no history-based annotation within the entries of the suggestion list and adding this feature was suggested as an enhancement to the system. An option to hide already visited pages from the suggestion list might also be considered.
- For the research prototype implementation, the file names, as opposed to the page titles, were used in the suggestion list. File names are not always as meaningful as the page titles, which should be used for any further development of the system.
- The suggestion list should be keyword based allowing links to pages containing the keywords as opposed to the current implementation which lists pages and allows the display of keywords covered on that page.
- The suggestion list should be permanently visible as opposed to being on a floating popup that can be obscured by the main browser window. Participants also commented that the content of the suggestion list should be more structured and an alternative representation, for example a graphical representation, would be useful.
- Users also commented on the time delay in the suggestion process which is due to the need to build up a list of pages to generate suggestions from and the periodic nature of

> the suggestion list refresh. Within MLTutor 10 pages have to be visited before suggestions can be made. This is due to the need to collect sufficient data for the machine learning algorithm to process. Once 10 pages have been visited, the suggestion list is refreshed periodically. The time between refreshes is set within the application and can easily be adjusted.

These comments from users of the system suggest areas where effort should be directed for future development work with the system.

In addition a number of technical issues with the conceptual clustering algorithm used in the MLC were identified. These technical issues with the clustering algorithm will need to be addressed before further development of MLTutor takes place.

DISCUSSION

With this deeper understanding of the strengths and limitations of the implementation of MLTutor, we can relate this development to work in adaptive hypertext.

Browsing path analysis

Balabanovic (1997) points out that a machine learning approach to support browsing without any particular goal in mind is useless. MLTutor has been designed for use in an educational context and specifically supports task-oriented browsing. If the browsing activity of a user is aimless, it is difficult to determine any regularity or any meaningful pattern in that behaviour. However, if the intention of a user's navigation is to complete a number of specific tasks, or to answer questions during browsing, then monitoring the user interaction in order to provide guidance becomes more feasible (Taylor and Self, 1990).

MLTutor uses machine learning techniques to search for patterns within the content of material accessed during a user's information seeking activity. Beaumont (1994) argues that the bandwidth of the information contained in a user's browsing pattern might be too narrow to elicit information about the user's interest; however, as noted by other researchers (Hirashima *et al*, 1998; McEneaney, 1999), browsing patterns are a fundamental source of information representing the user's interaction with the system. While such patterns have been investigated by several researchers (Sun *et al*, 1995; Lieberman, 1995; McEneaney, 1999) who have applied various AI techniques, such as heuristic search, dynamic programming and neural networks, little work has been done on the application of machine learning techniques to dynamically build user profiles in the field of adaptive hypermedia.

Within MLTutor, any patterns identified in the browsing history by the MLC of the system are used to dynamically generate suggestion rules which are used to recommend pages related to the browsing. By this mechanism the MLTutor system aims to help users to locate information relevant to their current interest.

The suggestion rules within MLTutor are generated by use of a novel combination of clustering and inductive machine learning algorithms. The dynamically generated rules form a profile that holds a generalisation of the user's current area of interest. This profile is updated as further pages are visited and new rules created.

In future development, augmenting the recommendations given by the MLC of MLTutor with *social navigation* (Dieberger *et al*, 2000; Riedl, 2001) techniques should also be considered. Social navigation is currently viewed as a new means of aiding users to find their way through an information space by making information trails left by previous users available to new users.

Attribute based systems

Information retrieval systems rely on document categorisation strategies and these strategies are typically based on keyword descriptions of information. These keyword descriptions are used by information retrieval systems to relate associated documents together. Similarly, attribute

based machine learning algorithms process attribute descriptions of objects. In both cases the selection of features to describe objects is fundamental to performance.

The MLTutor has based attribute encoding on a classification of hyperlink anchors within the hypertext. Hypertext links facilitate navigation and, as such, typically relate pages at a conceptual level. A distinct advantage of using hypertext anchors to describe documents is the ease with which they are identified within the hypertext. Although this strategy relies on hypertext documents being constructed in a sensible manner the strategy has been shown as a viable option. However, the MLTutor design is capable of incorporating any attribute based scheme for encoding documents due to the nature of the machine learning algorithms employed.

Adaptive navigational support

As a result of the machine learning process within MLTutor, the suggestion list is adaptive to the ongoing and changing requirements of the user. This is achieved by use of a sliding window technique that takes into account recent browsing. The use of browsing paths alleviates the need for any additional feedback from a system user at all.

Furthermore, the adaptivity provided by MLTutor can be disregarded without penalty or overhead to the user. By ignoring system suggestions, or turning the facility off, the MLTutor becomes a plain hypertext browsing system familiar to all users of the WWW. This solution to providing adaptivity is intended to prevent alienation of potential users; some users may not need to see suggestions and others may not be comfortable with unfamiliar adaptive features within the interface.

WWW based systems

While the WWW offers huge potential for distance learning, the mechanism of the Web can be employed to deliver information within an individual organisation or classroom. The WWW can be considered a vast network of interlinked documents, effectively a huge hypertext system. Within this network specific sites cater for specific needs, often with links to other related sites. However, the Web of documents is growing in an unregulated and unstructured manner with important connections between highly related documents missing. Consequently, finding specific or relevant information on the Web can be difficult.

The MLTutor presents a solution to this issue by introducing an adaptive facility, which aims to assist in the search for information. The MLTutor dynamically generates an individual profile in the form of suggestion rules based on a user's history of browsing activity. These rules are used to suggest additional pages the user may be interested in. A significant benefit of the MLTutor is that suggestions may relate to documents the user has not yet seen, or may not be aware of, as they are not directly connected to the document the user has accessed. The suggestion list mechanisms in MLTutor allows direct access to these additional pages even though no link physically exists between them.

The machine learning approach employed to build the list of suggestions ensures that the suggestions made are relevant to the user's current area of interest. Effectively, the MLTutor provides a mechanism for re-structuring a hypertext document to cater for individual preferences without restricting access in any way.

This approach has synergies with the approach suggested by Stotts and Furuta (1991) who proposed a flexible structure, to overlay a fixed structure, as a solution to personalising a hypertext system.

As an alternative to analysing a Web page, a more fine-grained approach, splitting a HTML page content into smaller chunks (Maglio and Farrell, 2000) may be applied. This approach may be possible with the growth of XML and the facility to define tags and the structural relationships between them.

CONCLUSION

This paper has presented the work undertaken on the MLTutor development to date. The objective of this research was to design, implement, test and evaluate a prototype system capable of testing the technical feasibility of using machine learning techniques to analyse browsing patterns within hypertext, and to use this analysis to provide adaptive navigational support without the need for pre-defined stereotypical profiles or user relevance feedback.

The MLTutor implementation serves as a proof of concept, demonstrating the feasibility of the approach. As a practical application of established machine learning algorithms, this work has identified limitations of algorithms that appeared to be well suited to this purpose; this has resulted in the implementation of SG1 (as an adaptation of ID3), and future work will be addressed at overcoming the limitations of the conceptual clustering algorithm discussed above.

In order to evaluate the system a comparative empirical study was conducted. The evaluation of adaptive systems is particularly complex as the results of the adaptation are personal to a specific user's set of circumstances and, as such, an empirical study is the most appropriate strategy for evaluation. The quantitative results from the evaluation study are indicative of improved performance for the adaptive versions but, due to sample size, not conclusively so. However, the qualitative elements of the study identified a number of issues that can very usefully be addressed before running any further empirical trials. In particular, it became apparent that the conceptual clustering algorithm has limitations (concerning sort step sensitivity and the generation of a 'bin' cluster) that need to be addressed; users also made various useful suggestions about the presentation of the suggestion list, as noted above.

Although faults in the experimental design limit the conclusions that can be drawn about MLTutor, the results of the evaluation do show that MLTutor is a robust and functional system and suggest the potential benefits of using a machine learning approach to provide adaptivity based on an analysis of browsing patterns. The empirical methodology also highlighted issues about the design that would have been missed by a simple quantitative analysis, and serves as a model for future studies of adaptive hypertext.

ACKNOWLEDGEMENT

This work was supported by a studentship from the School of Computing Science, Middlesex University. We are grateful to David Benyon, Paul Cairns, Alan Hutchinson, Ian Williams, Chris Huyck, Chris Kindberg, Steve Torrance, the 30 subjects took part in the experiment and fellow research students for useful discussions and feedback on this work as it was progressing.

REFERENCES

Armstrong R., Freitag D., Joachims T. and Mitchell T. (1995). WebWatcher: A Learning Apprentice for the World Wide Web. *AAAI Spring Symposium on Information Gathering from Distributed, Heterogeneous Environments*, Stanford, CA.

Balabanovic M. (1997). Exploring versus Exploiting when Learning User Models for Text Recommendation. *User Modeling and User-Adapted Interaction*, Vol. 8, pp 71-101, Kluwer Academic Publishers.

Beaumont I. (1994). User Modelling and Hypertext Adaptation in the Tutoring System ANATOM-Tutor. *UM'94 Fourth International Conference on User Modelling*, 4(1), pp 21-45.

Brusilovsky P. (1996). Methods and techniques of adaptive hypermedia. *User modelling and User-Adapted Interaction.* Special issue on: Adaptive Hypertext and Hypermedia, Vol. 6, No. 2-3, July 1996.

Bush V. (1945) As We May Think. *The Atlantic Monthly*, July 1945. Vol. 176, No. 1, pp 101-108.

Crabtree B. I. and Soltysiak S. J. (1998). Identifying and tracking changing interests. *International Journal on Digital Libraries* 1998. Vol. 2, pp 38-53.

Edwards P., Bayer D., Green C.L. and Payne T. R. (1996). Experience with learning agents which Manage Internet-Based Information. *In AAAI Spring Symposium on Machine Learning in Information Access*, pp 31-40, Menlo Park, CA-AAAAI.

Edwards P., Green C. L., Lockier P. C. and Lukins T.C. (1997). Exploiting Learning Technologies for World Wide Web Agents. *Intelligent World Wide Web Agents*. Colloquium organised by Professional Group C4 (Artificial Intelligence). Monday, 17 March 1997, Savoy Place, London.

Dieberger A., Dourish P., Hook K., Resnick P. and Wexelblat A. (2000). Social navigation: Techniques fo Building More Usable Systems. *Interactions*, ACM Press New York, NY, USA, Vol. 7, Issue 6 Nov./Dec. 2000, pp 36-45.

Hirashima T., Matsuda N., Nomoto T. and Toyoda J. (1998). Context-Sensitive Filtering for Browsing in Hypertext. *Proceedings of IUI'98*, San Francisco CA, USA, pp 119-126.

Hohl H., Bocker H. D. and Gunzenhauser R. (1996). Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming. *User Modelling and User-Adapted Interaction*, Vol. 6, No. 23, pp 131-155.

Hook K. (1997). Evaluating the Utility and Usability of an Adaptive Hypermedia System. *Proceedings IUI'97*, pp 179-186, Orlando, Florida USA.

Hutchinson A. (1994). *Algorithmic Learning*. Oxford University Press Inc, New York.

Kushmerick N., McKee J. and Toolan F. (2000). Towards Zero-Input Personalization: Referrer-Based Page Prediction. *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based System, AH'2000*, Brusilovsky P. Stock O. and Strapparava C. (Eds), LNCS 1892, pp 133-143.

Lang K. (1995). NewsWeeder:Learning to filter Netnews. *Proc. 12th International Machine Learning Conference* (ML95), Morgan Kaufmann, San Francisco, pp 331-339. .

Lieberman H. (1995). Letizia: An Agent That Assists Web Browsing. *Proceedings of 14th International Joint Conference on Artificial Intelligence, Montreal, Canada*, pp 924-929.

Maglio P. P. and Farrell S. (2000). LiveInfo: Adapting Web Experience by Customization and Annotation. *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based System, AH'2000*, Brusilovsky P. Stock O. and Strapparava C. (Eds), LNCS 1892, pp 144-154.

McEneaney J. E. (1999). Visualizing and Assessing Navigation in Hypertext. *Proceedings of Hypertext '99*, Darmstadth, Germany, pp 77-84.

Michalski R. S. (1980). Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2*, pp 349-361.

Michell T. M. (1997). Does Machine Learning Really Work? *AI magazine*, Vol.18, No.3, pp 11-20.

Pazzani M., Muramatsu J. and Billsus D. (1996). Syskill & Webert: Identifying interesting web sites. *Proc. Amer. Conf. on Artificial Intelligence (AAAI `96)*, pp. 54-61.

Payne T. R. and Edwards P. (1997). Interface agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. *Applied Artificial Intelligence*, 11(1), pp 1-32.

Reich Y. (1994). Towards Practical Machine Learning Techniques. In *Proceedings of the First Congress on Computing in Civil Engineering (Washington, DC)*, pp. 885-892, ASCE, New York.

Riedl M.O. (2001). A computational model and classification framework for social navigation. *Proceedings of the International Conference on Intelligent User Interfaces 2001*, ACM Press New York, NY, USA, pp 137 – 144.

Quinlan J. R. (1986). Induction of Decision Trees. *Machine Learning*, Vol.1, pp 81-106.

Stotts D. P. and Furuta R. (1991). Dynamic Adaptation of Hypertext Structure. *Hypertext'91 Proceedings*, pp 219-231.

Stern M. K.and Woolf B. P. (2000). Adaptive Content in an Online Lecture System. *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'2000*, Brusilovsky P. Stock O. and Strapparava C. (Eds), LNCS 1892, pp 227-238.

Sun C. T., Ching Y. T. and Lin F. X. (1995). Modelling Hypermedia Navigation: An AI Approach. *Proceedings of AI-ED'5*, Greer J. (Eds),Washington, DC, AACE, pp 365-372.

Taylor C. and Self A. J. (1990). Monitoring hypertext users. *Interacting with Computers* Vol. 2, No. 3, pp 297-312.