How Learner's Proficiency May Be Increased Using Knowledge about Users within an E-Learning Platform

Dumitru Dan Burdescu and Marian Cristian Mihăescu
University of Craiova, Craiova, Romania
Faculty of Automatics, Computers and Electronics
Software Engineering Department
E-mail: burdescu@software.ucv.ro, http://software.ucv.ro/~burdescu_dumitru/
E-mail: mihaescu@software.ucv.ro, http://software.ucv.ro/~mihaescu_cristian

Keywords: e-Learning, learning proficiency, knowledge acquisition

Received: July 24, 2006

Representing knowledge about the user of a web application by decision trees can offer remarkable information regarding a dataset. We have designed and developed an E-Learning platform that has built in the capability of monitoring and storing user traffic. A model of analysis of user traffic by building decision trees from gathered data is proposed. The analysis has two outcomes. Firstly, decision tree structure can give an objective measure of the interestingness and quality of the data. In this analysis we may see whether or not the data is representative or not and whether we may obtain sound knowledge. Secondly, the analysis may reveal the learner's behaviour in the continuous learning environment. The possible outcomes of this analysis are the learner's proficiency, accumulated knowledge, or learning curve.

Povzetek: Analiziran je vpliv uporabe spleta na hitrost učenja.

1 Introduction

During last decade it has been possible to observe the quick growth of interest in Web-based education. The purpose of the paper is to present a method of analysing data gathered from an e-Learning platform. The platform itself is a web application used by secretaries, professors, students and an administrator in a collaborative manner to accomplish a learning process. Each of these four roles has assigned a set of allowed actions. All users have to authenticate with a username and password and then the role and the set of allowed actions are determined.

The administrator, secretaries and professors have mainly management duties that set up the platform (i.e. sections, disciplines, course materials, tests, exams, etc.). The students, on the other hand, download course materials, communicate with secretaries and professors, and take tests and exams. All student activities are monitored and saved for off-line analysis. The goal is to employ different analysis methods on monitored data in order to accomplish user modelling and characterization. In this connection we present a method of building decision trees from gathered data.

The goal of the platform is to guide students in the educational process. The enforced educational technologies have two main outcomes. Firstly, the student benefits from the continuous learning environment in the way that his/her creativity and mental outlook skills are supported and even improved. On the other hand, this process is monitored with the purpose of obtaining performance assessments. The platform itself is a complex system with many capabilities that are at student's disposal. Monitoring user traffic and building decision trees represents a way of analysing data gathered from our e-learning platform with the aim of measuring how efficient was the learning process.

2 The E-Learning Platform

The main goal of the platform is to give students the possibility to download course materials, take tests or pass final examinations and communicate with all involved parties. To accomplish this, four different roles were defined for the platform: sysadmin, secretary, professor and student.

The main task of sysadmin users is to manage secretaries. A sysadmin user may add or delete secretaries, or change their password. He may also view the actions performed by all other users of the platform. All actions performed by users are logged. In this way the sysadmin may check the activity that takes place on the application. The logging facility has some benefits. An audit may be performed for the application with the logs as witness. Security breaches may also be discovered.

Secretary users manage sections, professors, disciplines and students. On any of these a secretary may perform actions like add, delete or update. These actions will finally set up the application such that professors and students may use it. As conclusion, the secretary manages a list of sections, a list of professors and a list of

students. Each discipline is assigned to a section and has as attributes a name, a short name, the year of study and semester when it is studied and the list of professors that teach the discipline which may be maximum three. A student may be enrolled to one or more sections.
The secretaries have also the task to set up the structure of academic years for all sections. The structure of an academic year is made of a list of periods. All periods that define the academic year are disjunctive in time and are characterized by a name, start date and end date. For each period there are also set up the exams that may be taken and the grants that are needed. For example, in winter examining session there may be taken only exams from the first semester and there is no need for grant from either professor or secretary. This way of defining what the student can do and when proved to be very flexible and easy to understand and use.
The main task of a professor is to manage the assigned disciplines while s discipline is made up of chapters. The professor sets up chapters by specifying the name and the course document. Only students enrolled in a section in which a discipline is studied may download the course document and take tests or examinations. Besides setting up the course document for each chapter, the professor manages test and exam questions. For each chapter the professor has to define two pools of questions, one used for testing and one used for exams. He specifies the number of questions that will be randomly extracted to create a test or an exam. Let us suppose that for a chapter the professor created 50 test questions and 60 exam questions and he has set to 5 the number of test questions and to 10 the number of exam questions that are to be randomly withdrawn. It means that when a student takes a test from this chapter 5 questions from the pool of test question are randomly withdrawn. When the student takes the final examination at the discipline from which the chapter is part, 15 questions are randomly withdrawn: 5 from the pool of test question and 10 from the pool of exam question. This manner of creating tests and exams is intended to be flexible enough for the professor.
All tests and exams are taken under time constraints. For each chapter the professor sets up a number of seconds necessary to answer questions that chapter. When a test or exam is taken all the seconds are summed thus obtaining a maximal interval of time in which the student has to finish the test. The elapsed and remaining time are managed on server side and presented to the student after each answered question.
Tesys application offers students the possibility to download course materials, take tests and exams and communicate with other involved parties like professors and secretaries.
Students may download only course materials for the disciplines that belong to sections where they are enrolled. They can take tests and exams with constraints that were set up by the secretary through the year structure facility.
Students have access to personal data and can modify it as needed. A feedback form is also available. It is composed of questions that check aspects regarding the usability, efficiency and productivity of the application with respect to the student's needs.

3 How and What Data are Monitored

The platform has two methods of monitoring user activity. First one is through a log file which records each executed action. Each action has a resulting row in the log file.
Since the business logic of the platform is Java based, log4j utility package was employed as a logging facility and is called whenever needed within the logic of the application. The utility is easy to use; logging process is managed by log4j.properties file. The next lines present how the utility was set up.

```
log4j.appender.R.File=D:/devel/Tomcat/idd.log
log4j.appender.R.MaxFileSize=1000KB
log4j.appender.R.MaxBackupIndex=5
```

These lines state that all the logging process will be done in idd.log file and will have a maximum file size of 100KB in maximum five files.
The main drawback of this technique is that the data from the file is in a semi structured form. This makes the information retrieval to be not so easy task to accomplish. On the advantages, logging activity may be very helpful in auditing the platform or even finding security breaches. This logging facility is also very helpful when debugging during development or when analysing peculiar behaviour during deployment.
To overcome the semi structured shape of logged activity a structured way of gathering activity information was enforced. The activity table was added in the database and all actions were recorded in the manner of one record per action. In the next table the structure of activity table is presented.

Field	Description
id	primary key
userid	identifies the user who performed the action
date	stores the date when the action was performed
action	stores a tag that identifies the action
details	stores details about performed action
level	specifies the importance of the action

Table 1: Structure of activity table

After five months of deployment, the activity table contains more than 50,000 records and we suppose that until the end of the learning cycle there will be close to 100,000 records. All this logged activity may also be very helpful in an audit process of the platform. The records from the activity table represent the raw data that will be further analysed.

The activity of a student may be seen as a sequence of sessions. A session starts when the student logs in and finishes when the student logs out. A session may be seen as a sequence of actions. The next figure presents

the activity diagram from platform point of view. Within the platform each student has an associated activity diagram.

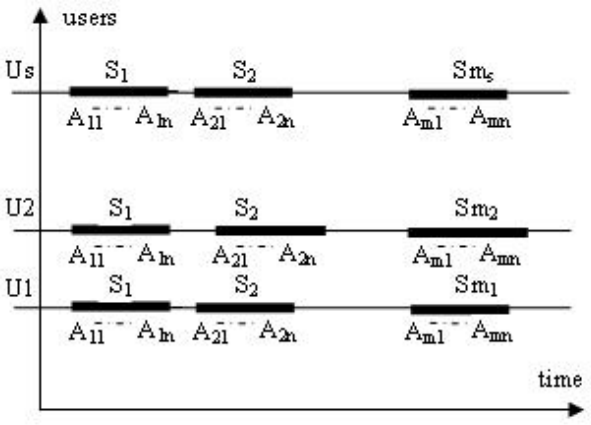

Figure 1: The activity diagram for platform users

In the diagram it may be seen the activity of all s users (U1,, U2, …, Us). The activity of each user is composed of a number of sessions. User Us in the diagram has m_s associated sessions. At finest level, a session is composed of a number of actions, session Sm_s has m_n associated actions. In a session, the first action is to login and the last one is logout. After one hour of inactivity the user is automatically logged out such that user sessions can be precisely determined.

4 Building Decision Tree from Data

Choosing between two learning algorithms given a single dataset is not a trivial task [4]. From all these representations we think decision trees are a very good start in the process of data analysis. Decision trees, as structures, may give a very conclusive idea regarding the "goodness" of data we try to analyse. Starting an analysing process with shaping the data in the form of decision trees gives a very good idea whether or not the data that we have may lead to conclusive or important results. Still, the whole process is much more than choosing an algorithm. Many learning schemes have various parameters, and suitable values must be chosen for these. In most cases, results can be improved markedly by a suitable choice of parameter values, and the appropriate choice depends on the data at hand. For example, decision trees can be pruned or unpruned and, in the former case, a pruning parameter may have to be chosen. More generally, the learning scheme itself will have to be chosen from a range of available schemes. In all cases, the right choices depend on the data itself [1].

A decision tree is a flow-like-chart tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test and leaf nodes represent classes [1]. So, the first step is to define a list of attributes that may be representative for modelling and characterizing student's activity. Among the attributes there may be:

- the number of logins,
- the number of taken tests,
- the average grade for taken tests,
- the exam results
- the number of messages sent to professors.

The basic algorithm for decision tree induction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner. The basic strategy is as follows. The tree starts as a single node representing the training samples. If the samples are all of the same class, then the node becomes a leaf and is labeled with that class. Otherwise, an entropy-based measure known as information gain is used for selecting the attribute that will best separate the samples into individual classes. This attribute becomes the "test" or "decision" attribute at the node. A branch is created for each known value of the test attribute, and the samples are partitioned accordingly. The algorithm uses the same process recursively to form the decision tree. Once an attribute has occurred at a node, it need not be considered in any of the node's descendents. The recursive partitioning stops only when one of the following conditions is true. All samples for a given node belong to the same class. There are no remaining attributes on which the samples may be further partitioned. This involves converting the given node into a leaf and labeling it with the class in majority among samples [1].
Impurity measures are an important parameter regarding the quality of the decision tree. Many different measures of impurity have been studied. Some algorithms measure "impurity" instead of "goodness" the difference being that goodness should be maximized while impurity should be minimized [5, 6, and 7].
The first step is to create a set of instances that hold the attributes. In the database there are 20 tables that hold the necessary data. Each student will represent an instance and each instance will be defined by its own attributes.
The next step effectively builds the decision tree. The computational cost of building the tree is O(mn log n)[2]. It is assumed that for n instances the depth of the tree is in order of log n, which means the tree is not degenerated into few long branches.
The information gain measure is used to select the test attribute at each node in the tree. We refer to such a measure an attribute selection measure or a measure of goodness of split. The algorithm computes the information gain of each attribute. The attribute with the highest information gain is chosen as the test attribute for the given set [1].

5 Results

In the study the relations that contain needed data are:

activity – here there are stored all actions performed by users;

test_results – here there are stored the results of all tests passed by all students;

messages – here there are stored all the messages sent or received by all users of the platform;

The next important step is attribute definition. In this study each student represents an instance we have to set

up the attributes that define each instance. From our relations there may be defined a large number of attributes which may have more or less importance regarding overall predictive power. In this study each instance is defined by four attributes:

nLogings – number of loggings of the user. This number may be associated also with the number of sessions;

nTests – number of taken tests passed by the student;

avgTests – average of grades for passed tests;

nSentMessages – number of messages sent by the students.

We developed a dedicated application called DatabaseRetriever for querying the database and creating an arff file.

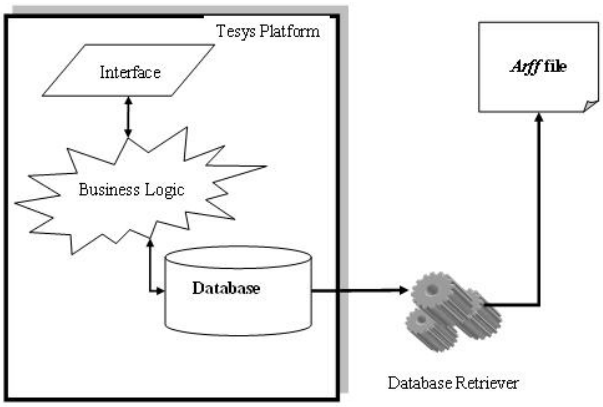

Figure 2: The functionality of DatabaseRetriver application

A system called Weka [3], which implements the decision tree building algorithm, uses arff format. In the next figure it is presented how the activity.arrff file is loaded into Weka workbench.

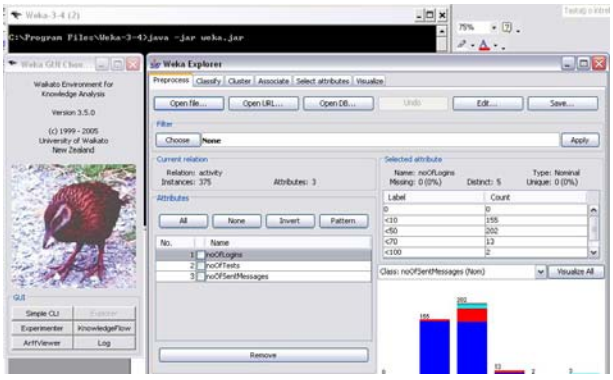

Figure 3: Loading the activity.arff file

In Figure 4 it is presented how J48 algorithm runs in Weka workbench after the *arff* file has been loaded.

The activity.arff file has a standard format which is composed of two sections. In the first one there is defined the name of the relation and the attributes. For each attribute there is defined the set of nominal values it may have. In the next lines it is presented the first section of the file.

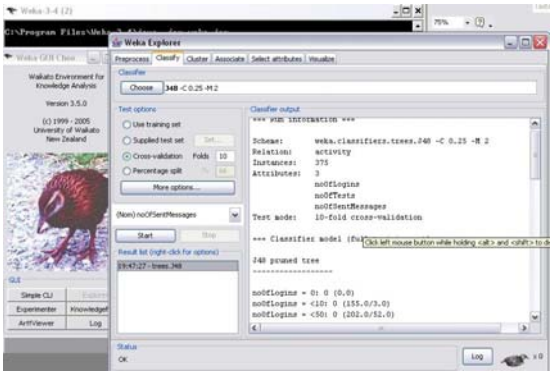

Figure 4: Running J48 algorithm

Here is a sample of the *arff* file.

@relation activity
@attribute nLogings {<10,<50,<70,<100,>100}
@attribute nTests {<10,<20,<30,<50,>50}
@attribute avgTests {<3,<6,<10}
@attribute nSentMessages {<10,<20,<30,<50,>50}

In this section of the file all attributes are defined. An important decision that is needed is to establish the granularity for each attribute which is represented by the number of nominal values it may take. As it can be seen from the above presented lines we consider five intervals for nLogings parameter: less than ten, less than fifty, less than seventy, less than one hundred and greater than one hundred. In the same way there is defined the set of possible values for each of the attributes.

The second section of the activity.arff file is represented by the data itself. Here are all the instances that will enter the classification process. In the next lines there are presented few instances that may be found in this section.

@data
<50,<20,<3,<10,
<50,>50,<6,<20,
<10,<20,<3,<10,
<50,<10,<3,<10,
<100,<50,<10,<50,

Each row represents an instance. For example, the first row represents an instance (a student) which entered the platform less than fifty times, took less than twenty tests, obtained an average of grades for taken tests less than three and sent less than ten messages to professors. In the same way there can be interpreted all other instances.

The activity.arff has 375 instances, each one corresponding to a student.

After running the algorithm the obtained tree had 17 leaves (which represent in fact classes) and 25 nodes. The time to build the model was 0.13 seconds.

The *decisionTreeModel.txt* file contents the obtained decision tree. Figure 4 presents the obtained model.

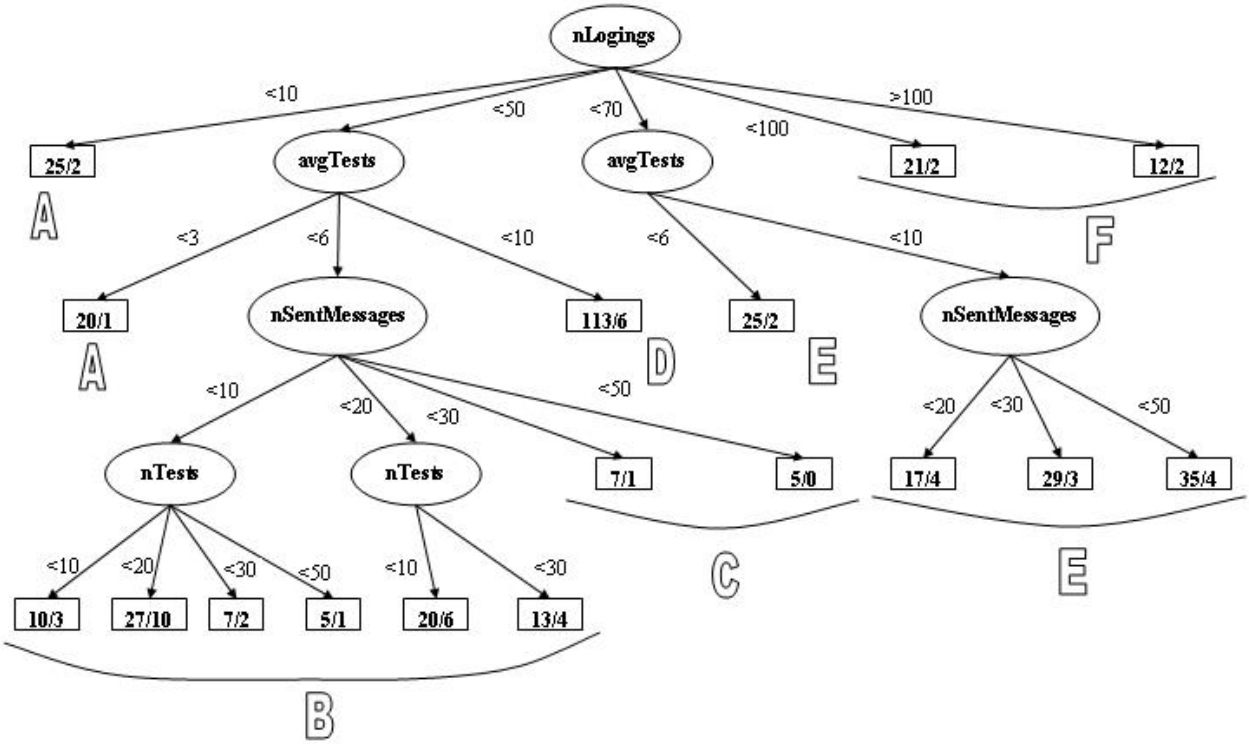

Figure 4: The decision tree in graphical form

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     activity
Instances:    375
Attributes:   4
              nLogings
              nTests
              avgTests
              nSentMessages
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

nLogings = <10: (25/2)
nLogings = <50
|  avgTests = <3 (20/1)
|  avgTests = <6
|  |  nSentMessages= <10
|  |  |     nTests = <10 (10/3)
|  |  |     nTests = <20 (27/10)
|  |  |     nTests = <30 (7/2)
|  |  |     nTests = <50 (5/1)
|  |  nSentMessages= <20
|  |  |  nTests = <10 (20/6)
|  |  |  nTests = <30 (13/4)
|  |  nSentMessages= <30 (7/1)
|  |  nSentMessages= <50 (5/0)
|  avgTests = <10 (113/6)
nLogings = <70
|  avgTests = <6 (11/3)
|  avgTests = <10
|  |  nSentMessages= < 20 (17/4)
|  |  nSentMessages= < 30 (29/3)
|  |  nSentMessages= < 50 (35/4)
nLogings = <100 (21/2)
nLogings = >100 (12/2)

Number of Leaves  :         17

Size of the tree :   25

Time taken to build model: 0.13 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         321               85.6    %
Incorrectly Classified Instances        54               14.4    %
Kappa statistic                          0.7085
Mean absolute error                      0.0664
Root mean squared error                  0.1994
Relative absolute error                 40.9    %
Root relative squared error             70.4213 %
Total Number of Instances              375

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
 0         0          0         0         0        
 0.844     0.09       0.938     0.844     0.888    <10
 0.9       0.166      0.764     0.9       0.826    <20
 0         0.005      0         0         0        <30
 0         0          0         0         0        <50
 0         0          0         0         0        >50
```

=== Confusion Matrix ===

```
  a b c 1 d e f     <-- classified as
45 1 0 2 0 0       |  a = 0
  1 80 10 7 4 1    |  b = <10
  0 0 12 0 0 1     |  c = <20
  0 2 2 13 0 2     |  d = <30
  7 7 1 0 92 0     |  e = <50
  2 0 1 0 1 93     |  f = >50
```

The most important part is the data analysis, which ensures that the model is valid and provides solid knowledge. The stratified cross-validation evaluation technique revealed that 321 (85.6 %) instances were correctly classified and 54 (14.4%) were incorrectly classified. The confusion matrix showed exactly the distribution of incorrectly classified instances among classes.

6 Conclusion

An e-Learning platform is currently deployed and used by almost 400 students and 15 professors. The platform embeds mechanisms for monitoring and storing user's activity. The platform's architecture is based on MVC (Model-View-Controller) paradigm ensuring application's scalability in development process. There are two implemented ways of monitoring activity: through log files and into relations that represent the model of the platform.

This platform has implemented capabilities of monitoring and saving user activities. An off-line application creates a data file in arff format that is used as input data for classification algorithms implemented in Weka system.

The results of running classification algorithms on recorded data showed that student's activity may be successfully classified as a function of specific activities. This may be the first step in modelling user activity and characterizing his/her learning proficiency based on past activity.

We may say that we have implemented an e-Learning platform that implements specific functionalities but which also benefits from the knowledge obtained in presented analysis process. The final outcome of the analysis module is that it may be regarded as a decision support system that feedbacks knowledge into the original e-Learning system in order to achieve certain goals. This approach may be a great benefit for students of the platform since their activity may be guided and coordinated in order to achieve pedagogical or psychological goals.

The next step may involve performance evaluation of the algorithm but with another set of attributes or even running other algorithms on data obtained from the current e-Learning platform. The final goal is to obtain a robust, scalable and accurate activity characterization model from which student's behavioural patterns may be extracted.

References

[1] Jiawei Han, Micheline Kamber "Data Mining – Concepts and Techniques" Morgan Kaufmann Publishers, 2001.

[2] Ian H. Witten, Eibe Frank "Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations" Morgan Kaufmann Publishers, 2000.

[3] www.cs.waikato.ac.nz/ml/weka

[4] Salzberg, S. "On Comparing Classifiers: Pitfalls to Avoid and a Recommended Approach" Data Mining and Knowledge Discovery 1:3 (1997), 317-327.

[5] Fayyad, U.M. & Irani, K.B. "The attribute specification problem in decision tree generation". In Proceedings of the Tenth National Conference on Artificial Inteligence, pp. 104-110, San Jose, CA, AAAI Press, 1992.

[6] Quinlan, J.R. "Induction of decision trees" Machine Learning, 1, 1986, 81-106.

[7] Buntine, W., & Nibblet T. "A further comparison of splitting rules for decision-tree induction" Machine Learning, 8, 1992, 75-85.