Building Intelligent Educational Networks

MARIAN CRISTIAN MIHAESCU
Software Engineering Department
University of Craiova
Bvd. Decebal, Nr. 107, 200440, Craiova, Dolj
ROMANIA
http://software.ucv.ro/~mihaescu_cristian/

Abstract: - An e-Learning platform together with its users may be seen as an Educational Network. Within such an Educational Network there are issues like: low reliability, unacceptable time response or bad resource management. The usual protocol between clients (users) and server (the e-Learning platform itself) is HTTP. This stateless protocol uses only request/response type interactions between clients and server. For improvement of afore mentioned issues there was built a module that gives the "intelligent" character of the Educational Network. This module records user performed actions, levels of data traffic and other performed activities in an attempt to solve or improve presented issues. The enforced mechanisms use state of the art machine learning algorithms, mathematical modeling and dynamic structures management, giving thus the intelligent character of the educational network.

Key-Words: - data traffic monitoring, machine learning, e-Learning, dynamic data structures

1 Introduction

The ever increasing demand for higher bandwidth, new applications and new protocols, as well as certain operations and manageability requirements of educational networks ask for transport technologies which ideally fulfill certain requirements. Scalability should range from Gbit/s to Tbit/s total capacity, per-channel bit rate should range from several Mbit/s to 40Gbit/s today with 100Gbit/s on the horizon and service flexibility should range from Fast Ethernet via Gigabit Ethernet (GbE) and 10GbE to OC-768 PoS and ultimately 100GbE in the future. Distances ranging from intra campus requirements to thousands of kilometers and management concepts (e.g. centralized management, distributed GMPLS control plane) are requirements that should be currently met in educational networks.

There was designed and developed an e-Learning platform called Tesys [1]. This platform has implemented facilities for following type of users: system administrators, secretaries, professors and students. Some activities implemented for students, like downloading course materials or taking tests or exams are sometimes very heavy regarding the computational load of the server and the data traffic transfer to and from the user.

This paper presents the structure and functionality of an Expertise Module (EM) that runs along the Tesys e-Learning platform. The main purpose the EM is to provide the intelligent character for the educational network implemented by Tesys. The functionality of the EM module is presented in Figure 1.

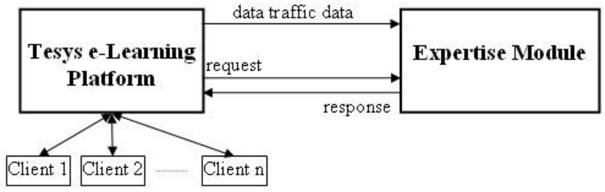

Figure 1. General structure of Educational Network

As presented in Figure 1 the input of EM is represented by data traffic data. The data are obtained by a custom implemented logging mechanism embedded within the platform's business logic. The platform is represented by the setup put in place in order to perform all necessary activities within the e-Learning process. The setup consists of course materials, test and exam quizzes that are set up by course managers and the overall setup performed by secretaries.

The data traffic is saved into structured format and after that it is fed to the Expertise Module. Once the EM is initialized, it can provide data back to Tesys e-Learning platform in the form of a response to specific requests. Within EM, there are two problems that are addressed.

One refers to employed business logic within the Expertise Module. This problem is considered from two points of view. One is from the point of view of the general architecture of EM and the other is from the point of view of analysis process itself.

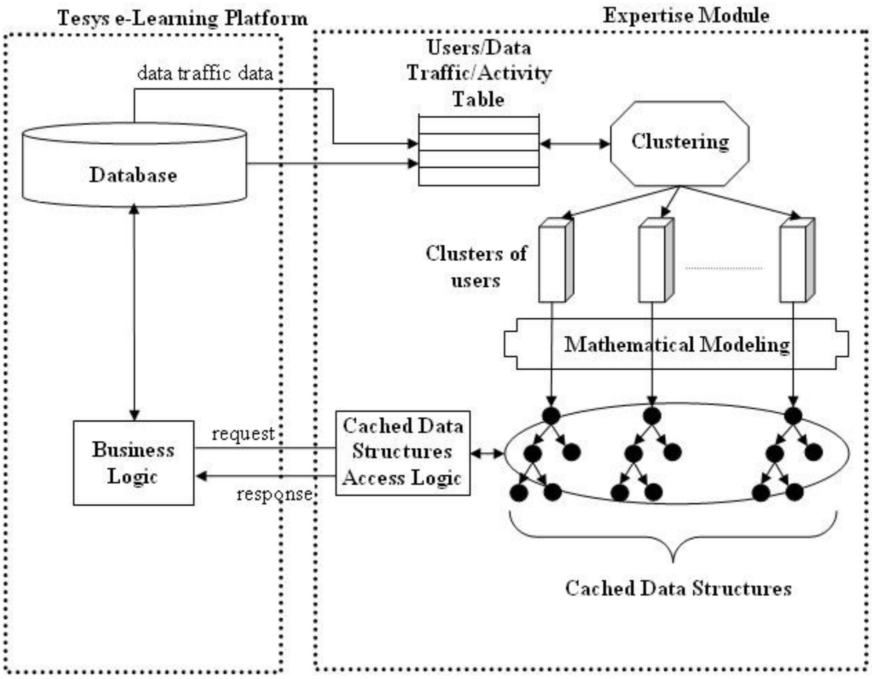

Figure 2. Detailed functionality of Expertise Module and integration with Tesys e-Learning platform

The second problem refers to the architecture of EM as a service. Once the EM has been instantiated it will work as a service for Tesys e-Learning platform. Under these circumstances the Intelligent Educational Network is represented by the combination of the Tesys platform and the Expertise module.

The activity of a student is seen as a sequence of sessions. A session starts when the student logs in and finishes when the student logs out. A session is represented by 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.

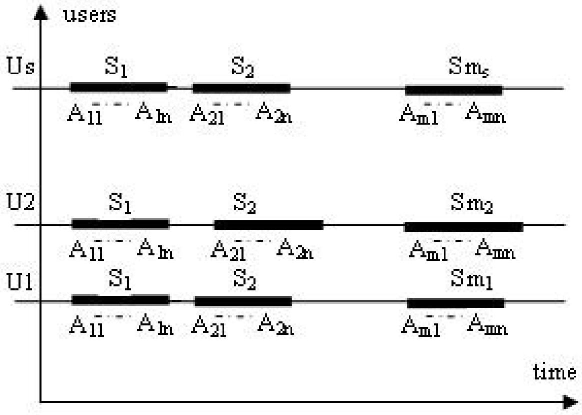

Figure 3. Activity diagram for students

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 ms associated sessions. At finest level, a session is composed of a number of actions, session Sms has mn 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. The notion of "user session" was defined as being a temporally compact sequences of Web accesses by a user. A new distance measure between two Web sessions that captures the organization of a Web site was also defined. The goal of Web mining is to characterize these sessions. In this light, Web mining can be viewed as a special case of the more general problem of knowledge discovery in databases.

Still, in performed experiments there were taken into consideration heavy traffic sessions. In this light, a session becomes more or less a time interval in which a user performs many requests as a part of a certain e-Learning activity. For example, taking a test is a heavy traffic session. Within a session there may be many heavy traffic sessions. For such sessions there was monitored the quantity of transferred data, the duration of the session and other information regarding that session like the user who performed the actions and the accessed resources. This data represent the raw data that is used as input in the analysis process.

Under these circumstances, a more detailed functionality diagram of Expertise Module is presented in Figure 2. The EM is basically an application that functions as a service for Tesys e-Learning platform. Its main goal is to decrease the response time of the platform by caching data in an intelligent way, increase reliability and provide efficient resource management. The intelligent feature of caching is given by the mechanisms implemented within the EM: clustering, mathematical modeling and data caching.

The Expertise Module has as input the data traffic data and the data from the database.

All these data populates the users-data traffic-activity table. All data regarding users represent features (parameters) that define each instance (user). The instances enter a clustering process such that students with a high degree of similarity are grouped together. The data traffic transferred within each cluster is mathematically modeled and needed data is inserted into a cached data structure. The logic implemented in Cached Data Structures Access Logic is also responsible for interfacing with the business logic of the platform. The communication is accomplished in a client-server architecture. When the business logic of Tesys needs specific data it firstly sends a request to Expertise Module and more exactly to Cached Data Structures Access Logic. If requested data is not found than classical way of obtaining it is used.

There are many different ways for representing patterns that can be discovered by machine learning. From all of them we choose clustering, which is the process of grouping a set of physical or abstract objects into classes of similar objects [2]. Basically, for our plat-form we create clusters of users based on their activity and data traffic.

As a product of clustering process, associations between different actions on the platform can easily be inferred from the logged data. In general, the activities that are present in the same profile tend to be found together in the same session. The actions making up a profile tend to co-occur to form a large item set [3].

There are many clustering methods in the literature: partitioning methods such as [9], hierarchical methods, density-based methods such as [6], grid-based methods or model-based methods. Hierarchical clustering algorithms like the Single-Link method [4] or OPTICS [5] compute a representation of the possible hierarchical clustering structure of the database in the form of a dendrogram or a reachability plot from which clusters at various resolutions can be extracted, as has been shown in [7]. From all of these we chose to have a closer look on partitioning methods.

After creating clusters of users based on their activity and transferred data, the data traffic transferred within each cluster is taken into consideration by Mathematical Modeling Module which in fact estimates the value of H Parameter. As a matter of precaution the mathematical modeling starts only when clustering process has reached a certain level of accuracy. Mathematical Modeling module will estimate three plots: R/S plot, Variance-Time plot and the Periodogram plot. Once the value of H parameter is over a threshold (e.g. 0.8) data may be cached into the Cached Data Structures. This is the place where important data is stored and is ready to be accessed by Tesys e-Learning platform.

The Cached Data Structure implements a dynamical data structure used for managing in main memory the data that is available for deployment through the Tesys e-Learning Platform. For implementing this structure there are used AVL trees [8,9]. A AVL tree is a self-balancing binary search tree. In an AVL tree the heights of the two child subtrees of any node differ by at most one, therefore it is also called height-balanced. Lookup, insertion, and deletion all take O(log n) time in both the average and worst cases. Additions and deletions may require the tree to be rebalanced by one or more tree rotations.

2 Tesys e-Learning Platform

The main goal of the application is to give students the possibility to download course materials, take tests or sustain 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.

2.1 Main Functionalities and Architecture

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 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.

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.

The e-learning platform consists of a framework on which a web application may be developed. On server side we choose only open source software that may run on almost all platforms. To achieve this goal Java related technologies are employed. In figure 2 we present the most general view of the software architecture from MVC point of view.

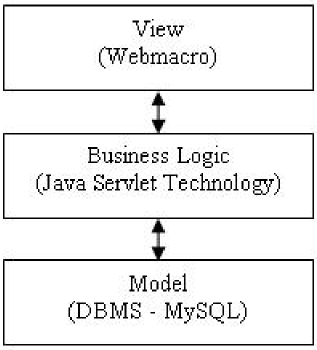

Figure 4. MVC architecture of the Tesys e-Learning platform

This architecture of the platform allows development of the e-learning application using MVC architecture. This three-tier model makes the software development process a little more complicated but the advantages of having a web application that produces web pages in a dynamic manner is a worthy accomplishment. The model is represented by DBMS (Data Base Management System) that in our case is represented by MySQL.

2.2 Data Traffic Logging Mechanisms

In this part we shall focus on describing the monitoring capabilities of the platform when running. The platform implements two ways of monitoring activity. Since business logic is implemented in Java, the log4j utility package was chosen to be used in order to log specific events. The next lines present how the utility was set up.

```
log4j.appender.R.File=D:/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 1000KB in maximum five files.

This utility package is also used in debugging process and the logs may be very useful in finding security breaches like unsuccessful attempts to log in or run actions that are not allowed.

The main disadvantage of this technique is the semi structured way in which information is stored. This makes the information retrieval and analysis to be not so easy.

The second method of monitoring user activity within the platform is through a database table called activity. In this table a record is added each time a user performs an action. In the next table it is presented the structure of activity table.

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

The details field stores specific information regarding the action that was executed. For example, if a secretary modifies the profile of a student in the details field there will be stored information about what fields were updated.

The level field specifies the importance of the executed action. There are defined three level of importance: 0,1 and 2 where level 0 specifies the critical actions.

So far, in activity table there are close to 40,000 recorded actions in almost four month of running the platform. At the end of the cycle there are expected almost 100,000 recorded actions.

3 The Expertise Module

The Expertise Module implements the classical steps of classical clustering presented in figure 5 [10]. Clustering produces initial categories in which values of a data set are classified during the classification process.

From all clustering algorithms categories we chose to have a closer look on those that use partitioning methods. Firstly, k-Means algorithm is taken into consideration since is simple and straight forward. That is why fuzzy C-means algorithm was employed. The procedure follows the standard knowledge discovery [10] but is accustomed for our specific situation.

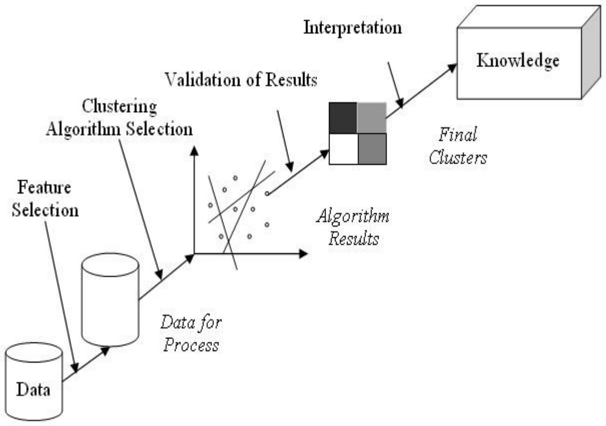

Figure 5. Steps for clustering process

k-Means algorithm works for a database of n objects and k, the number of clusters to form, a partitioning algorithm that organizes the objects into k partitions (k<n), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, often called similarity function, such as distance, so that objects within a cluster are "similar", whereas the objects of different clusters are "dissimilar" in terms of database attributes. So, the first step is to define a list of attributes that may be representative for modeling and characterizing student's activity.

The classic k-means algorithm is a very simple method of creating clusters. Firstly, it is specified how many clusters are being thought: this is the parameter k. Then k points are chosen at random as cluster centers. Instances are assigned to their closest cluster centre according to he ordinary Euclidean function. Next the centroid, or the mean, of all instances in each cluster is calculated – this is the "means" part. These centroids are taken to be the new centre values for their respective clusters. Finally, the whole process is repeated with the new cluster centers. Iteration continues until the same points are assigned to each cluster in consecutive rounds, at each point the cluster centers have stabilized and will remain the same thereafter [3]. From a different perspective for a cluster there may be computed the following parameters:

$$\mu = \frac{x_1 + x_2 + ... + x_n}{n}$$, the means

$$\sigma = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + ... + (x_n - \mu)^2}{n-1},$$ the standard deviation

p, the probability

The sum of all probabilities for all clusters is 1. If we know which of the distributions each instance came from, finding the parameters is easy. On the other hand, if the parameters are known finding the probabilities that a given instance comes from each distribution is easy. Given an instance x , the probability that it belongs to clus-ter A is:

$$\Pr[A \mid x] = \frac{\Pr[x \mid A] - \Pr[A]}{\Pr[x]} = \frac{f(x;\mu_A,\sigma_A)p_A}{\Pr[x]}$$

Where $f(x;\mu_A,\sigma_A)$ is the normal distribution function for cluster A, that is:

$$f(x;\mu_A,\sigma_A) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The EM algorithm takes into consideration that we know neither of these things: not the distribution that each training instance came from, nor the parameters μ, σ or the probability. So, we adopt the procedure used for the k-means clustering algorithm and iterate. Start with initial guess for the five parameters, use them to calculate the cluster probabilities for each instance, use these probabilities to estimate the parameters, and repeat. This is called the EM algorithm for "expectation-maximization". The first step, the calculation of cluster probabilities (which are the "expected" class values) is "expectation"; the second, calculation of the distribution parameters is "maximization" of the likelihood of the distributions given the data [3].

Although the EM algorithm is guaranteed to converge to a maximum, this is a local maximum and may not necessarily be the same as the global maximum. For a better chance of obtaining the global maximum, the whole procedure should be repeated several times, with different initial guess for the parameter values. The overall log-likelihood figure can be used to compare the different final configuration obtained: just choose the largest of the local maxima [3].

Within the Expertise Module the mathematical modeling is accomplished by estimating the self-similarity and long-range dependence character of data traffic [12, 13 and 14]. A process is considered to be self-similar if Hurst parameter satisfies the condition:

$$Y(t) = a^{-H} Y(at) \quad t > 0, a > 0, 0 < H < 1$$

where the equality is in the sense of finite-dimensional distributions. A second definition of self-similarity that is more appropriate in the context of standard time series, involves a stationary sequence $X = \{X(i), i > 1\}$. Let

$$X^{(m)}(k) = (1/m) \sum_{i=(k-1)m+1}^{km} X(i), \quad k = 1,2,...$$

It is not possible to use the definition to check whether a finite traffic trace is self-similar or not. Instead different features of self-similarity such as slowly decaying variances are investigated in order to estimate the Hurst parameter H.

Parameter H can take any value between 1/2 and 1 and the higher the value the higher the degree of self-similarity. For smooth Poisson traffic the value is H=0.5. There are four methods are used to test for self-similarity. These four methods are all heuristic graphical methods, they provide no confidence intervals and they may be biased for some values of H. The rescaled adjusted range plot (R/S plot), the Variance-Time plot and the Periodogram plot, and also the theory behind these methods, are described in detail by Beran [15] and Taqqu et al. [14]. Molnar et al. [16] describes the index of dispersion for counts method and also discuss how the estimation of the Hurst parameter can depend on estimation technique, sample size, time scale and other factors.

The Expertise Module has also implemented the logic responsible for managing the data that is to be retrieved to Tesys platform at request. In this prototype implementation there were implemented the basic operations on the AVL tree structure: insertion, look-up and deletion.

The reason for choosing the AVL tree data structure is because it has the advantage of being simpler to implement than other self-balancing binary search trees, such as red-black trees or multiway trees like B-trees or T-Trees while their average-case performance is just as efficient.

Since the implemented operations are the classic ones the node structure has an very important role. Table 2 presents the structure of a node from the AVL tree structure.

Field	Description
activityId	**key-identifies the performed activity**
metaInfo	**meta information regarding the performed activity**
userId	**identifies the user who performed the actions within the session**
date	**stores the date when the actions were performed**
data	**stores the data**

Table 2. Structure of node from the AVL tree

The *activityId* field is the key of the structure. Whenever a user starts performing a specific activity than this activity is looked-up in the Cached Data Structure. The *metaInfo* field holds information specific to performed actions. For example, in the case of a DOWNLOAD action in the *metaInfo* field there will be stored information regarding the time duration of the session, the average data traffic speed in bytes/second and data about the assets involved (file names, sizes, locations, etc.) The *userId* field is a foreign key that identifies the user who performed the actions within the heavy traffic session. The date field records when the actions were performed and the data field holds the important data that is to be retrieved to Tesys e-Learning platform at request.

The managing is performed by Cached Data Structure Access Logic. This business logic interfaces with Data Model Logic for insertion and with the logic of Tesys for look-up and deletion. Whenever a call for look-up data within the Cached Data Structure this is accomplished firstly according to *activityId* field. Once the correct activity has been identified than the *metaInfo* field is inspected. When the needed data matches regarding the looked asset the data is then retrieved to Tesys e-Learning platform for deployment.

4 Experimental Results

The study started by setting up the Tesys e-Learning platform. This means that all the learners, course managers and secretary accounts have been created and the platform was populated with data: course materials, test and exam questions.

This platform is currently in use and has three sections and at each section, four disciplines. Twelve professors are defined and more than 650 learners. At all disciplines, there are edited almost 2500 questions. In the first month of usage, almost 500 tests were taken. In the near future, the expected number of learners may be close to 1000.

Recording learner's activity under these circumstances provides great information regarding user traffic. After six month of usage, there are more than 40,000-recorded actions.

Once the platform was up and running the Expertise Module started receiving data regarding users, user's traffic and user's performed actions. As presented, each user represents an instance for the clustering process and is represented by a set of parameters.

After the parameters (features) have been set they are computed. In Figure 5 this step is named Feature Selection and produces the Data for process. Data is represented by the whole history of all users which may be found in relations of the database (e.g. activity, exam results, test results, messages, etc.) and in semi structured log files. The Feature Selection will produce the set of instances (sometimes called points) that will represent the input for Clustering Algorithm Selection. Depending on algorithm a number of clusters is obtained each instance being assigned to one or more clusters.

Validation of results produces the final clusters that implement the model. The validation procedure has two main outcomes: firstly it proves the correctness of results for current dataset and gives an idea of how the model will perform on new data.

In the next paragraphs there will be described in detail the whole process of knowledge discovery. Everything starts with the data from the database of the e-Learning platform.

The database of the platform contains 21 relations. Among the most important ones are: *user*, *role*, *userrole*, *usersections*, *sections*, *questions*, *testquestions*, *exam-questions*, *test-results*, *exam-results*, *messages* and *activity*.

The preparation gets data from the database and puts it into a form ready for processing of the model. Since the processing is done using machine-learning algo-rithms implemented in Weka workbench [11] and custom implementation, the output of preparation step is in the form of an *arff* file. Under these circumstances, we have developed an offline Java application that queries the platform's database and crates the input data file called *activity.arff*. This process is automated and is driven by a property file in which there is specified what data/attributes will lay in *activity.arff* file.

The most important step in this procedure is the attribute selection and the granularity of their nominal values. The number of attributes and their meaning has a cru-cial importance for the whole process since irrelevant attributes may degrade classification performance in sense of relevance. On the other hand, the more attributes we have the more time the algorithm will take to produce a result. Domain knowledge and of course common sense are crucial assets for obtaining relevant results.

For a student in our platform we may have a very large number of attributes. Still, in our procedure we use only three: the number of loggings, the number of taken tests and the number of sent messages. Here is how the arff file looks like:

```
@relation activity
@attribute nLogings {0,<10,<50,<70,<100,>100}
@attribute nTests{0,<10,<20,<30,<50,>50}
@attribute noOfSentMessages
{0,<10,<20,<30,<50,>50}
@data
<50,<10,<10,
<50,<20,0,
<10,<10,0,
```

As it can be seen from the definition of the attributes each of them has a set of five nominal values from which only one may be assigned. The values of the attributes are computed for each of the 650 students and are set in the *@data* section of the file. For example, the first line says that the student logged in less than fifty times, took less than ten tests and sent less than ten messages to professors.

In order to obtain relevant results we pruned noisy data. We considered that students for which the number of loggings, the number of taken tests or the number of sent messages is zero are not interesting for our study and degrade performance and that is why all such records were deleted. After this step there remained only 268 in-stances.

Running the EM algorithm from Weka package created three clusters. The procedure clustered 91 instances (34%) in cluster A, 42 instances (16%) in cluster B and 135 instances (50%) in cluster C. The following table shows in which cluster the instances belong after running the EM algorithm.

Instance	Cluster A	Cluster B	Cluster C
1	1	0	0
2	1	0	0
3	0	1	0
…….	…	….	…
268	0	0	1

Table 2. Distribution of instances after EM algorithm

For estimation of Hurst parameter there was chosen a 3 hours interval, between 18:00 and 21:00 which is considered to be a heavy traffic period.

The interval from 18:00 to 21:00 was chosen for close analysis. The R/S plot estimated H parameter to a value of 0.89. The time-variance plot showed a slope of -0.320 which means a value of H of 1+slope/2=0.84. The IDC (Index of Dispersion for Counts) shows an H parameter of 0.88. In Periodogram plot there may be observed a value of H = 0.85. These methods do not obtain exactly the same values but values are over 0.5 which is a good indication of traffic's self-similarity. Having in mind that non-stationary traffic may be easily taken as self-similar stationary traffic there were also examined smaller intervals of time bins. H parameter was estimated for each of the 6 intervals of 30 minutes between 18:00 and 21:00. In this way, there was estimated H parameter for three hours from a complete interval of 24 hours.

The fact that traffic is found to be self-similar does not change its behavior but it changes the knowledge about real traffic and also the way in which traffic is modeled. It has lead many [17] to abandon the Poisson-based modeling of network traffic for all but user session arrivals. Real traffic, well described as self-similar, has a "burst within burst" structure that cannot be described with the traditional Poisson-based traffic modeling.

After another three month of running the Cached Data Structure reached a number of almost 800 nodes and a size of almost 250 MB. From this point we started to study how response time is influenced by the employed architecture. There was build a mechanism that performed two requests virtually at the same time: one for normal retrieval of data from the hard drive and one that looked up for the data within the cached data structure. The results are good, especial at subsequent similar requests, due to time locality of data. Table 3 presents the obtained results for three situations.

Activity	meta info	Normal access time retrieval	Cached access time retrieval	Difference
DOWNLOAD	**materieId = 7** **courseId=5** **chapterId=3** **size=5MB**	**0.9 s**	**0.78 s**	**-13.3%**
DOWNLOAD	**materieId = 10** **courseId=1** **chapterId=2** **size=3MB**	**0.8 s**	**0.67 s**	**-16,25%**
DOWNLOAD	**materieId = 6** **courseId=3** **chapterId=5** **size=0.9MB**	**0.6 s**	**0.65 s**	**+8.3%**

Table 3. Experimental measurements – comparison between normal access and cached access

The experimental results showed in general a decrease in access time. Still, there are some situations then the response time is greater when the response is delivered through Expertise Module. This is the situation when the overhead introduced by auxiliary logic of Data Retrieval Module is greater that the normal retrieval. The worst-case time situation showed an overhead of 35% over the normal access while best-case time situation showed an improvement of 30%. The average-case performance shoed a general decrease in access time retrieval of 13.5 percent.

4 Conclusions

This paper presents an Expertise Module that runs along an e-Learning platform and whose goal is to decrease the access time of users to assets, increase reliability and enhance resource access and management. This module was divided into three levels of analysis: clustering, mathematical modeling and data caching. The first level creates clusters of students based on their performed activities and transferred data traffic. At this level there is used EM clustering algorithm implemented by Weka system. The user clustering process produced three clusters of users.

The mathematical traffic modeling was performed on data obtained for users that belong to a certain cluster. Once the clusters are created the data traffic transferred by students that belong to a cluster of students is analyzed for self-similarity. Mathematical modeling estimates the self-similarity of data traffic. This is accomplished by heuristic graphical methods: R/S plot, variance-time plot, IDC plot, periodogram plot. The analysis is performed rigorously for a three hours interval, from 18:00 to 21:00 but also for the whole day.

All the analysis follows a proposed analysis process that has as input data regarding executed actions and transferred bytes within the platform and has as output estimates of the Hurst parameter.

Values found for Hurst parameter are very promising. All calculations showed values above 0.7 and many times above 0.8 which indicate a good level of self-similarity.

The differences regarding Hurst parameter are due to estimation method, bin size and point of time.

When this characteristic is met than the data is inserted into the cached data structure represented by a dynamic structure, more precisely an AVL tree. The Expertise Module provides data for Tesys e-Learning platform from the AVL tree as requested.

A AVL tree structure is used as caching data structure. This is mainly performed to obtain data in timely manner.

The Expertise Module has been tested on data obtained from the e-Learning platform on which 650 learners were enrolled and had activity for six month. The results are satisfactory and prove that the Expertise Module can be successfully used in an e-Learning process for decreasing the time response.

The Expertise Module is planned for running on the same e-Learning platform (same disciplines and same test and exam questions) but on different set of learners. This may lead to further and continuous improvement of the system.

The Expertise Module may also run near other evaluation environments in order to decrease the time response. Within each module there may be used different machine learning techniques or data structures such that different setups may be tested.

References:

[1] Burdescu, D.D., Mihăescu, M.C., 2006. *Tesys: e-Learning Application Built on a Web Platform.* Proceedings of International Joint Conference on e-Business and Telecommunications. Setubal, Portugal, pp. 315-318.

[2] Jiawei Han, Micheline Kamber "Data Mining – Concepts and Tech-niques" Morgan Kaufmann Publishers, 2001.

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

[4] Ester M., Kriegel H.-P., Sander J., Xu X.: "A Density-Based Algo-rithm for Discovering Clusters in Large Spatial Databases with Noise", Proc. KDD'96, Portland, OR, pp.226-231,1996.

[5] Sander, J., Qin, X., Lu, Z., Niu, N, Kovarsky, A. Automated Extrac-tion of Clusters from Hierarchical Clustering Representations. PAKDD'03.

[6] Nasraoui O., Joshi A., and Krishnapuram R., "Relational Clustering Based on a New Robust Estimator with Application to Web Mining," Proc. Intl. Conf. North American Fuzzy Info. Proc. Society (NAFIPS 99), New York, June 1999.

[7] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," Proc. of the 20th VLDB Conference, pp. 487-499, Santiago, Chile, 1994.

[8] G. Adelson-Velskii, E.M. Landis, 1962. An algorithm for the organization of information. Doklady Akademii Nauk SSSR, 146:263–266, (Russian). English translation by Myron J. Ricci in Soviet Math. Doklady, 3:1259–1263, 1962.

[9] Donald Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Pages 458–475 of section 6.2.3: Balanced Trees. Note that Knuth calls AVL trees simply "balanced trees".

[10] Fayyad, M.U., Piatesky-Shapiro, G., Smuth P., Uthurusamy, R., Advances in Knowledge Discovery and Data Mining. AAAI Press., 1996.

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

[12] W. Willinger, M. S. Taqqu, R. Sherman, and D. Wilson, "Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level," in Proceedings of SIGCOMM '95, pp. 100-113.

[13] W. Willinger, V. Paxson and M.S. Taqqu, "Self-similarity and heavy tails: structural modeling of network traffic," in A practical guide to heavy tails: statistical techniques and applications, R. Adler, R. Feldman and M.S. Taqqu, Eds. Birkhauser, Boston, 1998.

[14] M. S. Taqqu and V. Teverovsky, "On estimating the intensity of long-range dependence in finite and infinite variance time series," in A practical guide to heavy tails: statistical techniques and applications, R. Adler, R. Feldman and M.S. Taqqu, Eds. Birkhauser, Boston, 1998.

[15] J. Beran. Statistics for Long-Memory Processes. Chapman & Hall, New York, 1994.

[16] S. Molnar, A. Vidacs and A. Nilsson, "Bottlenecks on the Way Towards Fractal Characterization of Network Traffic: Estimation and Interpretation of the Hurst Parameter". International Conference on the Performance and Management of Communication Network, Tsukuba, Japan, 17-21 November 1997.

[17] V. Paxson and S.Floyd, "Wide-area traffic: The failure of poisson modeling," in IEEE/ACM Transactions on Networking, vol. 3, no. 3, pp. 226-244, 1995.