Improving the Quality of Knowledge Discovery Process by Information Gain Computing

Dumitru Dan Burdescu, Marian Cristian Mihăescu
Software Engineering Department
University of Craiova
Craiova, Romania
{mihaescu, burdescu}@software.ucv.ro

Bogdan Logofatu, Costel Marian Ionaşcu
CREDIS Department /
Analysis and Statistics Department
University of Bucharest/Craiova, Romania
logofatu@credis.ro, icostelm@yahoo.com

***Abstract*— Knowledge discovery process is one of the key activities in improving the quality of a system. This paper presents a custom approach for improving the quality of a knowledge discovery process based on information gain computing. The baseline knowledge discovery process is based of M Trees and is used to cluster learners from an e-Learning environment based on parameters representing performed activities. The baseline process is improved by assigning weights to each parameters according with the information gain computed for each parameter.**

Keywords- knowledge discovery; M Tree; information gain; e-Learning.

I. INTRODUCTION

This paper addresses the problem of building a higher quality knowledge discovery process for an e-Learning platform. The knowledge discovery process is based on data that represent activities performed by learners in an e-Learning environment. The learners are distributed into clusters using an M Tree structure. The items that are involved in the process are represented by learners and each learner is described by a set of parameters. The creation of the hierarchical structure (in this case the M Tree) is based on the notion of distance between items. A trivial perspective is to normalize the parameters and to assign equal importance (weight) for each parameter. From the data analyst point of view this approach simplifies the analysis procedure but the obtained knowledge may not reflect the data from qualitative point of view. This paper introduces the concept of weight for all the features that describe the instances. The paper introduces the concept of weight as an automatically objective computed value. Computing a weight for each parameter may represent an overhead for the analysis process but the obtained knowledge may have more contextual value. Under these circumstances, the main issue becomes the procedure of assigning weights to parameters. Computing the weight is based on entropy and information gain computing for each defined feature. After the information gain is computed for each feature, a proportional weight is assigned for each feature. Therefore, the classical Euclidian distance formula between items becomes a weighted one.

The obtained clusters represent in a more realistic manner the input data since each parameter is weighted according with the amount of knowledge it brings into the data.

The second section presents the related work in the field. The third section presents the employed infrastructure and methods. Section four presents the analysis process and section five presents a sample experiment. The final section presents the conclusions and future works.

II. RELATED WORK

One domains that is discussed in this paper is clustering as state of the art machine learning methodology. Here, the clustering quality computation as a main tool in assessing the quality of the knowledge discovery process is discussed. The second domain regards information theory and is concerned with entropy and information gain as a mechanism for determining the weight of each parameter that describes data items. These two domains are put together in a framework that aims at improving the quality of knowledge discovery process for an e-Learning application [1].

Clustering is a machine learning technique used to group a set of items into subsets. This technique may be used in educational domain to enhance our understanding of learning process to focus on identifying, extracting and evaluating variables related to the learning process of students [2]. K-means clustering is a widely used method that is easy and quite simple to understand [3]. Cluster analysis describes the similarity between different cases by calculating the distance. These cases are divided into different clusters due to their similarity. There are studies [4] that use students data to analyze their learning behavior to predict the results and to warn students at risk before their final exams. The theoretical background of k-means is presented in [5]. This well-known clustering algorithm tends to uncover relations among variables already presented in dataset and is implemented by tools and libraries [6, 7].

M Trees [8] are spatial data structures that may be used for clustering data that is described by a set of parameters. The main drawback of this approach is that even though it is able to employ more features during the search, these features are compared using a single distance function. Another drawback regards the fact that the distance function does not make any difference between parameters. This drawback is discussed in this paper by computing the information gain for each parameter. An extension of the M tree [9], which goes further, is able to compare different features with arbitrary distance functions. This approach is used in general in situations when a custom query mechanism based on multiple features are required. In general, the output of the procedure is represented by computed centroids for each obtained cluster. An item belongs to one cluster or another according with the minimum distance to a centroid. This approach is used when the goal of the procedure is just to cluster a new test instance. Still, more complex queries (range or k-nearest neighbor) may also be set up. In this scenario, we may be interested in finding all the items that reside in a certain range or the closest k items.

When a clustering algorithm is used, it is necessary to test its performance, and compare it with that of other methods. Such comparisons are difficult to be performed but the effort is necessary because the quality of the clustering process must be assessed. The commonly used approach uses a benchmark that implements a set of quality assessment metrics.

Finally, the information gain is computed for each attribute that describes the items from the input dataset. The computing is based on entropy computing as an average measure of information [10, 11].

III. EMPLOYED INFRASTRUCTURE AND METHODS

A. The e-Learning Environment

E-Learning systems are mainly concerned with delivery and management of content (e.g., courses, quizzes, exams, etc.). Since we are speaking about a web platform the client is represented by the browser, more exactly by the learner that performs the actions.

Defining the e-Learning infrastructure or the presented purpose represents the first and the most important step. In this phase, all the possible actions that may be performed by a learner need to be presented. The resources that are delivered by the e-Learning system are also identified. Finally, there are identified the highly complex business logic components that are used when actions are performed by learners.

Each implemented action needs to have an assigned weight. In the prototyping phase, the assignment of weights is performed manually according with a specific setup. This assumes that we have an e-Learning system that is already set up. The main characteristics regard the number of learners, the number of disciplines, the number of chapters per discipline, the number of test/exam questions per chapter and the dimension of the document that is assigned to a chapter. The data that is obtained from analyzing a certain setup will represent the input data for the simulation procedure.

Another type of activities regarding learners are represented by the communication that takes place among parties. Each sending or reading of a message is assigned a computed average weight.

A sample e-Learning setup infrastructure may consist of 500 students, 5 disciplines, 5 to 10 chapters per discipline, 10 to 20 test/exam questions.

For this infrastructure here may be established a list of costs for all needed actions that may be performed by learners. The weight assigned to an action takes into consideration the complexity of the action and the dimension of the data that is obtained as response after the query is sent.

For obtaining reasonable weight, a pre-assessment procedure is performed. The simulation tool performs this procedure from a computer that resides in the same network as the server such that response times are minimal. Each request that is composed and issued to the e-Learning platform is measured in terms of time and space complexity. A scaling factor will assign each action a certain weight such that the scenarios that will be created when real time testing starts will have a sound basis.

The pre-assessment procedure firstly loads all the data regarding the analyzed e-Learning platform. This means the data about all managed resources (e.g., disciplines, chapters, quizzes, etc.) are loaded such that the simulation tool may build valid requests for the e-Learning environment.

B. Clustering with k-means algorithm

K-means is the most important flat clustering algorithm. Its objective is to minimize the average squared Euclidean distance of items from their cluster centers where a cluster center is defined as the mean or centroid #μ of the items in a cluster ω: centroid.

$$\bar{\mu}(\omega)=\frac{1}{|\omega|}\sum_{\bar{x}\epsilon\omega}\bar{x} \qquad (1)$$

The algorithm is:

```
procedure k-means ($x_1$, $x_2$, …, $x_N$; K)
 {$c_1$, $c_2$, …, $c_K$} ← Select Random Centroids
 for ( k=1, k<K )
   centroid$_k$ = $c_k$;//these are initial centroids
  while (#centroids are not same){
   for ( k=1, k<K ){
     for ( n=1, n<N ){
       j = index of corresponding cluster
       #put $x_n$ in corresponding cluster Cj
     }//end for
   }//end for
   for (k=1, k<K)
     # compute centroids for all clusters
  }//end while
```

In most cases, K-means quickly reaches either complete convergence or a clustering that is close to convergence. In the latter case, a few items would switch membership if further iterations are computed. This computation has a small effect on the overall quality of the clustering.

C. M Trees

The classical M Tree algorithm has been adapted such that is the final structure has two levels. The procedure for building the structure takes into consideration both the desired number of clusters and the filling factor of a leaf node.

```
procedure MTree ($x_1, x_2, \ldots, x_N$; K; F)
// K – the number of clusters
// F– filling factor
 for ( i=1, i<N ){
   $C_i$ = FindCentroid(centroids, $x_i$);
   if ( #Leaf[$C_i$] has F instances)
     if (#we have k clusters)
       #put $x_i$ in Leaf[$C_i$]
     else
       #split Leaf[$C_i$]
   else
     #put $x_i$ in Leaf[$C_i$]
   RecomputeCentroids(Leaf[$C_i$])
 }//end for
```

D. Information Gain

Information gain is calculated using a measure called entropy, which we first define for the case of a binary decision problem and then define for the general case.

Given a binary categorization, **C**, and a set of examples, **S**, for which the proportion of examples categorized as positive by **C** is *p+* and the proportion of examples categorized as negative by **C** is *p-*, then the **entropy** [5] of **S** is:

$$\mathrm{Entropy(S)} = -\mathrm{p}_+ \log_2(\mathrm{p}_+) - \mathrm{p}_- \log_2(\mathrm{p}_-) \quad (2)$$

The reason we defined entropy first for a binary decision problem is because it is easier to get an impression of what it is trying to calculate.

Given an arbitrary categorization, **C** into categories **c1, ..., cn**, and a set of examples, **S**, for which the proportion of examples in $\mathbf{c_i}$ is $\mathbf{p_i}$, then the entropy of **S** is:

$$\mathrm{Entropy(S)} = \sum_{i=1}^{n} -\mathrm{p_i} \log_2(\mathrm{p_i}) \quad (3)$$

We now return to the problem of trying to determine the best attribute to choose for a particular node in a tree. The following measure calculates a numerical value for a given attribute, **A**, with respect to a set of examples, **S**. Note that the values of attribute **A** will range over a set of possibilities which we call **Values(A)**, and that, for a particular value from that set, ***v***, we write ***Sv*** for the set of examples which have value ***v*** for attribute ***A***.

The information gain [5] of attribute ***A***, relative to a collection of examples, ***S***, is calculated as:

$$\mathrm{Gain\ (S, A)} = \mathrm{Entropy(S)} - \sum_{v \in \mathrm{Values(A)}} \frac{|S_v|}{|S|} \mathrm{Entropy}(S_v) \quad (4)$$

The information gain of an attribute can be seen as the expected reduction in entropy caused by knowing the value of attribute ***A***.

E. Assessing Clustering Quality

The indicators [12] that are taken into considerations are:

Tightness Indicator:

$$Q = \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{x \epsilon C_i} d(x, \mu_i) \quad (5)$$

where $|C_i|$ is the number of points from cluster i. The value for Q will be small if the data points from the cluster are close. Thus, in the comparison analysis procedure the clusters with smaller computed values of Q have higher quality.

Homogeneity Indicator:

If the centroids of clusters are computed with formula: $r_k = \frac{1}{n_k} \sum_{x \epsilon C_k} x$, where x are the instances from cluster C_k than homogeneity indicator is:

$$H(C) = \sum_{k=1}^{K} \sum_{x \epsilon C_k} d(x, r_k)^2 \quad (6)$$

The value for H will be small if a cluster has homogeneous structure. This, in the comparison analysis procedure the clusters with smaller computed values of H, have higher quality.

Cluster Distance:

$$CD = \sum_{1 \le j \le k \le K} d(r_j, r_k)^2 \quad (7)$$

where j and k are indexes of clusters whose centroids r are taken into consideration. The value for CD will be big if the similarity among clusters themselves is low. Thus, in the comparison analysis procedure the method with bigger computed values of CD have higher quality.

Weakest Link between Points:

The weakest link for a cluster is the maximal value of all pairs of points belonging to the same cluster.

$$WL = \max(d(x_i, x_j)) \quad (8)$$

for all x_i and x_j belonging to the same cluster.

IV. Analysis Process

The analysis process uses a dataset of 150 students represented by seven attributes.

The parameters that characterize each instance are:

positiveCount – represents the number of correctly answered questions;

correctPercent – represents the percentage of correctly answered questions from the total number of questions;

totalTries – represents the total number of tries (answered questions);

avgTries – represents the medium number of tries per question;

avgQuestionTime – represents, on average, how long (in minutes) it takes for a student to answer a question;

totalTime – represents the total time spent on testing;

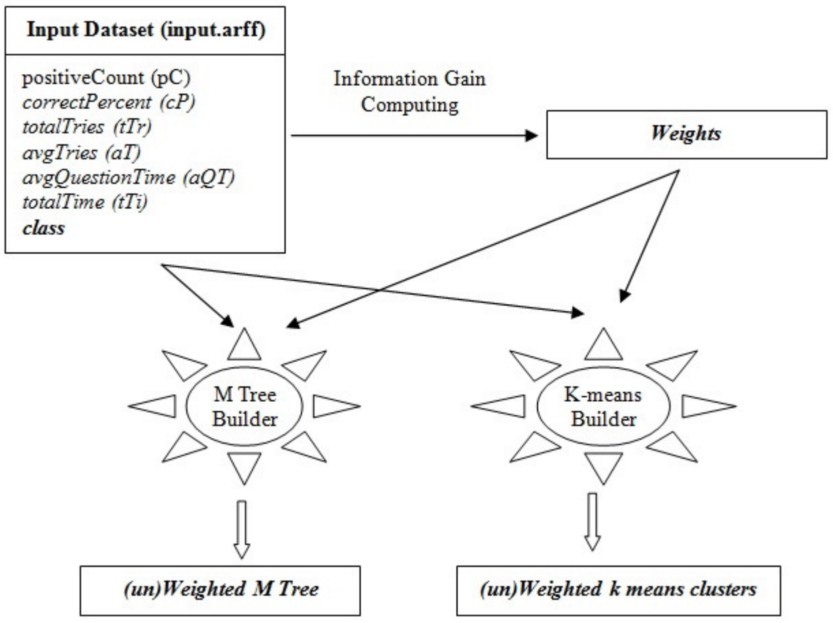

Figure 1. The architecture of the analysis framework

The last attribute represents the class and is not used during the clustering procedures. The class of the student may be *low*, *average* or *high*. The *class* attribute makes possible the usage of input data in an supervised learning context. This is necessary for computing the information gain brought by each attribute. This value is set by the data analyst which has also the domain knowledge. Still, the k-means and M Tree algorithms are unsupervised methods and thus the class of each item is not taken into consideration.

Figure 1 presents the analysis process. It may be observed that the input dataset is represented by *input.arff* file. In this file resides the data regarding the activity performed be learners that is used for computing the information gain for each attribute. After the information gain is determined for each attribute the weight of each attribute may be determined. Once the weights are determined, they may be used to build the weighted M Tree and weighted k-means clustering. These weighted models along with un-weighted models are thereafter analyzed by the clustering quality metrics.

The overall idea of the knowledge discovery process is to assess the quality of the obtained model in each analyzed situation.

This setup uses only normalized and continuous type parameters and a final nominal class attribute. The appearance of the class attribute allows a supervised approach on input data with the possibility of computing the information gain for each attribute. Once the class attribute is removed the learning becomes unsupervised and a clustering procedure (e.g., k-means, M Tree) may be used. The class variable is set by a domain knowledge data analyst for real life examples and thus there is no clear (mathematical) prior dependency between this variable and the rest of variables.

This approach makes the experiment to have real consistency regarding the learning process.

The analysis process is used in an e-Learning application for clustering students. All the students that have followed and finished courses represent the training set for the analysis process. The data coming from a new student is used as test data. The new student is clustered, which means he is assigned to the cluster which has the minimum distance from its centroid to the instance itself. Once the student in clustered the target cluster may be determined, as the cluster with the next increasing knowledge weight. A recommender system may use these data to determine the features in which the current student needs improvements. In a more general approach, each educational resource may represent a feature and thus the educational resources that need more attention may be determined.

V. SAMPLE EXPERIMENT

The goal of the experiment is to prove the concept and to objectively describe the results. The experiment uses an input dataset which there data for 50 students. The *input.arff* file has the following structure:

```
@RELATION activity

@ATTRIBUTE positivCount NUMERIC
@ATTRIBUTE correctPercent NUMERIC
@ATTRIBUTE totalTries NUMERIC
@ATTRIBUTE avgTries NUMERIC
@ATTRIBUTE avgQuestionTime NUMERIC
@ATTRIBUTE totalTime NUMERIC
@ATTRIBUTE class {low, avg, high}
```

```
@DATA
75,90,80,19,45,87,high
85,65,71,10,25,92,high
41,59,67,75,31,56,avg
…
```

All numeric attributes have real continuous values that are normalized in the range of 0 to 100. The normalization of the numeric values is performed with the following formula:

$$x_i = (\text{value}_i - \text{mean})/\text{range} \quad (9)$$

where:

- *value* is the initial value of the feature;
- *mean* is the average value from all values of the feature in the training set;
- *range* is the difference between maximum value of the feature and minimum value of the feature;
- x_i is the normalized computed value of the feature;

The first step is to build the k-means and M Tree models. At this step there are taken into consideration only the six numeric attributes. The obtained clusters by k-Means clustering have the following centroids and composition:

C1 (3, 5, 10, 28, 32, 45) *//Cluster 1's Centroid*
{12 instances}
C2 (34, 42, 56, 78, 62, 58) *//Cluster 2's Centroid*
{18 instances}
C3 (61, 75, 85, 69, 88, 69) *// Cluster 3's Centroid*
{20 instances}

The obtained clusters by M Tree clustering have the following centroids and composition:

C1 (4, 6, 9, 31, 28, 41) *//Cluster 1's Centroid*
{14 instances}
C2 (37, 40, 52, 80, 63, 62) *//Cluster 2's Centroid*
{19 instances}
C3 (65, 77, 82, 83, 89, 72) *// Cluster 3's Centroid9*
{17 instances}

For each clustering procedure there were computed the evaluation metrics presented in section three. The results are presented in the following table:

TABLE I. TIGHTNESS, HOMOGENEITY AND CLUSTER DISTANCE INDICATORS FOR K-MEANS AND M TREE DISTRIBUTIONS

Indicator	Clustering Procedure	
	k-means	*M Tree*
Tightness	7.80	8.85
Homogeneity	105.29	138.55
Clusters Distance	220.32	203.11

The link analysis for both distributions is presented in the following table:

TABLE II. WEAKEST LINK VALUES OBTAINED FOR K-MEANS AND M TREE DISTRIBUTIONS

Indicator	Clustering Procedure	
	k-means	*M Tree*
Weakest Link Cluster 1	0.92	1.25
Weakest Link Cluster 2	0.88	1.25
Weakest Link Cluster 3	0.89	0.57

The k-means results are obtained using Weka [6]. Weka is a collection of machine learning algorithms for data mining tasks which has implemented the k-means clustering algorithm.

All results presented so far do not take into consideration any weight of parameters. Thus, information gain is computed for each parameter and a normal distribution of weights is determined.

Firstly, the entropy is computed:

Entropy(S) = $-p_{high}\log_2(p_{high}) - p_{avg}\log_2(p_{avg}) - p_{high}\log_2(p_{high}) = -(14/50) * \log_2(14/50) - (20/50) * \log_2(20/50) - (16/50) * \log_2(16/50) = -(14/50) * -1.83 - (20/50) * -1.32 - (16/50) * -1.64 = 0.51 + 0.52 + 0.52 = 1.55$

For computing the information gain of each parameter all continuous values are transformed into nominal values of *low*, *average* and *high* using a normal distribution.

Gain(S, positiveCount) = $1.55 - (|S_{low}|/50)*\text{Entropy}(S_{low}) - (|S_{avg}|/50)*\text{Entropy}(S_{avg}) - (|S_{high}|/50)*\text{Entropy}(S_{high}) = 1.55 - (0.3)*\text{Entropy}(S_{low}) - (0.4)*\text{Entropy}(S_{avg}) - (0.3)*\text{Entropy}(S_{high}) = 1.55 - (0.3)*(0.91) - (0.4)*(0.81) - (0.3)*(0.92) = 0.677$

In a similar way, there is computed the information gain for all other parameters.

Gain(S, correctPercent) = 0.57
Gain(S, totalTries) = 0.88
Gain(S, avgTries) = 0.56
Gain(S, avgQuestionTime) = 0.38
Gain(S, totalTime) = 0.77

The formula for computing the corresponding weight for a feature takes into account the overall gain brought by all features. In general, if there are defined m features, the overall gain is defined as the following sum:

$$AllGain = \sum_{i=1}^{m} Gain\,(S, f_i) \quad (10)$$

The formula for computing the value of the weight for a certain feature f_i is:

$$W_{f_i} = \frac{Gain(S,f_i)*100}{AllGain} \quad (11)$$

According to the values obtained by computing the information gain the weight for each parameter is:

$W_{positiveCount} = 17.4$
$W_{correctPercent} = 14.8$
$W_{totalTries} = 22.97$
$W_{avgTries} = 14.62$
$W_{avgQuestionTime} = 9.92$
$W_{totalTime} = 20.10$

The obtained weights are used when computing the Euclidian distances between items in the process of building the M Tree structure and the k-means clusters.

Now, it time to rebuild the k-means and M Tree models taking into consideration the above computed weights. The obtained clusters by k-Means clustering have the following centroids and composition:

C1 (4, 5, 9, 30, 31, 7) *//Cluster 1's Centroid*
{14 instances}
C2 (37, 45, 52, 75, 65, 62) *//Cluster 2's Centroid*
{17 instances}
C3 (60, 72, 87, 68, 81, 72) *// Cluster 3's Centroid*
{19 instances}

The obtained clusters by M Tree clustering have the following centroids and composition:

C1 (5, 7, 10, 17, 26, 37) *//Cluster 1's Centroid*
{13 instances}
C2 (34, 40, 52, 77, 63, 59) *//Cluster 2's Centroid*
{16 instances}
C3 (62, 78, 69, 87, 81, 79) *// Cluster 3's Centroid9*
{21 instances}

For each clustering procedure there were computed the evaluation metrics presented in section three. The results are presented in the following table:

TABLE III. TIGHTNESS, HOMOGENEITY AND CLUSTER DISTANCE INDICATORS FOR K-MEANS AND M TREE DISTRIBUTIONS

Indicator	**Clustering Procedure**	
	k-means	***M Tree***
Tightness	7.85	9.25
Homogeneity	107.35	141.55
Clusters Distance	225.32	201.15

The link analysis for both distributions is presented in the following table:

TABLE IV. WEAKEST LINK VALUES OBTAINED FOR K-MEANS AND M TREE DISTRIBUTIONS

Indicator	**Clustering Procedure**	
	k-means	***M Tree***
Weakest Link Cluster 1	0.95	1.22
Weakest Link Cluster 2	0.91	1.33
Weakest Link Cluster 3	0.93	0.78

The M Tree results are obtained using a custom Java implementation of the algorithm. The main differences of this implementation compared with classical M Tree algorithm regards two aspects. One regards the general structure of the tree that is restricted to two levels. This means there is only one root node where centroids along with covered radius are placed. The second issue regards the way k (the number of clusters) and f (the filling factor) are managed. If the algorithm is required to produce a certain number of clusters, the instances are placed into appropriate clusters until a filling factor is reached. When this happens, a split is performed. Splitting is no longer performed when the desired number of clusters is reached. In this way, the clustering process is directly managed by the values k and s.

The comparison of the two obtained distributions reveals the fact that the M Tree distribution clusters have lower quality than the ones obtained by usage of k-means. Still, the results obtained by M Tree are very different from the ones obtained by k-means. All indicators presented in table 1 have better results for k-means than the ones obtained for M Tree. It can be observed that the tightness and homogeneity are better (because they have smaller values) for k-means than for M Tree.

The results obtained when the attributes are weighted show a similar quality with the ones obtained without weights. Still, the models are quite distinct and we think the one in which attributes are weighted is a more realistic one.

The clustering process belongs to the class of unsupervised learning schemes that tries to obtain patterns in the training dataset. One common approach used to enhance the learning scheme is feature scaling and/or mean normalization. The presented approach of assigning weights features is custom to e-Learning domain but may adapted in any learning process. Intuitively, the main reason for this approach is that features that characterize an instance may not all have the same importance or significance. One approach might be to have a domain expert assign a weight value for each feature. This may be regarded as a manual configuration of the learning scheme. In this paper, we use an automatic approach. This means that the weights are set according with the information gain provided by each feature. That is why the obtained patterns have the chance of being more realistic since they are determined in an objective way in correspondence with the provided dataset. The fact that the quality of the both clustering schemes, weighed and un-weighed, are similar means that both obtained models may be used with confidence. Still, the weighted model is different from the un-weighted model in the way that the obtained values of coordinates for centroids are different. Taking into account that the weights were automatically and objectively determined, the weighted model is a more realistic one.

VI. CONCLUSION AND FUTURE WORK

This paper presented a procedure that measures the degree in which the effectiveness of on e-learning process has improved. The analysis process is data centered. The data represents experiences provided by learners. In this study, six features (attributes) characterize each learner.

The study is repeated with weighted attributes. The weights are proportional with the information gain produced by each attribute. The information gain is computed as the difference between system's entropy and the entropy of the system when each attribute is taken into consideration.

The goal of the procedure is to produce clusters of users using two different techniques: standard k-means algorithm implemented in weka and a custom flavor of M Tree algorithm with a custom implementation.

The input dataset is restricted to a sample of 50 learners. An automated analysis of the obtained clusters is performed by computing some basic clustering quality metrics: Tightness, Homogeneity, Clusters Distance and link analysis. The obtained results show an acceptable quality of the M

Tree clusters although the computational complexity of the algorithm is much lower than complexity of k-means.

The main goal of the paper is to find a more realistic knowledge discovery process that obtains acceptable results with complexity much smaller than a classical procedure.

The quality of the obtained clusters has a direct influence over the degree in which the e-learning process has been performed. Unsupervised classification (clustering) is one of the main methods for making evidence regarding the knowledge acquisition of learners. Once a high quality distribution has been discovered a learner may by clustered at certain moments and progress may be evaluated. Of course, the process needs to be well defined and needs to be based on a high quality clustering procedure.

The future works regard different aspects. A first issue would be to replicate the procedure with more data. This may be accomplished on hundreds or even thousands of learner, if data is available. The clustering procedure is highly influenced by the initial centroids. In custom initialization is advisable. A good starting point may be obtained by using a k-means clustering on a sample dataset from the entire dataset. The quality of the clustering process is directly influenced by the choices made regarding k and f values. Thus, an initialization step may also refer to prior computation of the optimal number of clusters and optimal filling factor. The computation of these parameters may be delegated to other high quality clustering procedure that works on a data sample.

Finally, there may be defined procedures for assessing progress in time and even recommendations. The progress in time may be computed classifying the learner from time to time. This may yield to a learning path that has been followed by the learner. More than this, there may be obtained recommendations for the learner. The recommendations may regard necessary actions necessary to be taken by the learner in order to improve his learning curve.

REFERENCES

[1] C. Romero, S. Ventura, A. Zafra, and P. de Bra, “Applying Web Usage Mining for Personalizing Hyperlinks in Web-based Adaptive Educational Systems”, Computer&Education, Elsevier, Volume 53, Issue 3, pp. 828-840, 2009.

[2] A. El-Halees, “Mining Students Data to Analyze e-Learning Behavior: A Case Study”, Department of Computer Science, Islamic University of Gaza, 2009.

[3] K.S. Qaddoum, “Mining student evaluation using associative classification and clustering”, Communications of the IBIMA vol. 11 IISN 1943-7765, 2009.

[4] G. Ben-Zadok, A. Hershkovitz, R. Mintz, and R. Nachmias, “Examining online learning processes based on log files analysis: a case study”, Research, Refelection and Innovations in Integrating ICT in Education, 2007.

[5] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", 2nd edition, The Morgan Kaufmann Series in Data Management Systems, Jim Gray, Series Editor, 2006.

[6] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update”, SIGKDD Explorations, Volume 11, Issue 1, 2009.

[7] http://www.ibm.com/developerworks/java/library/j-mahout/ "Introducing Apache Mahout", ibm.com. 2011 [last update]. Retrieved 13 September 2011.

[8] P. Ciaccia, M. Patella, and P. Zezula, “M-tree: An efficient access method for similarity search in metric spaces”, in Jarke, M., Carey, M. J., Dittrich, K. R., Lochovsky, F. H., Loucopoulos, P., and Jeusfeld, M. A., editors, Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB 1997), Morgan Kaufmann, pp. 426-435, 1997.

[9] P. Ciaccia and M. Patella, “The M2-tree: Processing complex multi-feature queries with just one index”, In Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries, 2000.

[10] W. Baobao, M. Jinsheng, and S. Minru, “An enhancement of K-Nearest Neighbor algorithm using information gain and extension relativity”, in International Conference on Condition Monitoring and Diagnosis (CMD2008), pp.1314-1317, 2008 .

[11] M. H. Dunham, “Data Mining: Introductory and Advanced Topics”, Prentice Hall, 2003.

[12] P. Zezula, G. Amato, V. Dohnal, and M. Batko, “Similarity Search-The Metric Space Approach”, Advances in Database Systems, Vol. 32, Springer, 2006.