Learning Task-related Strategies from User Data through Clustering

Mihaela Cocea*†, George D. Magoulas†
*School of Computing, University of Portsmouth, Portsmouth, UK
Email: mihaela.cocea@port.ac.uk
†London Knowledge Lab, Birkbeck College, University of London, London, United Kingdom
Email: gmagoulas@dcs.bbk.ac.uk

***Abstract*—In exploratory learning environments, learners can use different strategies to solve the same problem. Not all these strategies, however, are known to the teacher and, even if they were, they need considerable time and effort to introduce them in the knowledge base. In this paper we propose a learning mechanism that extracts strategies from user data and presents them to the teacher for further authoring. To this end, a clustering approach is used in which the strategies of learners are grouped into clusters and the teacher is presented with a representative strategy for each cluster. The teacher can then decide whether to store the proposed strategies or to author them further. This approach allows populating the knowledge base using user data, thus saving authoring time for the teacher.**

***Keywords*-clustering, learning from user data, exploratory learning environments**

I. Introduction

Exploratory learning environments (ELEs) present learners with constructionist activities [1] in which learners vary the parameters of their constructions/models and observe the implications on the behaviour of their models. ELEs are associated with the so-called ill-defined domains [2], in which the problems are less structured and the boundaries between correct and incorrect approaches to solve a task are not clearcut. Moreover, problems in these domains are characterised by having several equally valid solutions. ELEs that are equipped with guidance and support mechanisms have been proven to have positive impact on learning when compared with other structured learning environments [3]. Lack of support, however, may hinder learning [4] and outstrip the advantages of ELEs. Therefore, to make ELEs more effective, intelligent support is needed, despite the difficulties arising from their open nature.

To address this, [5] proposed a learner modelling mechanism for monitoring learners' actions when constructing and/or exploring models by modelling sequences of actions reflecting different strategies in solving a task. An important problem, however, remains: only a limited number of strategies are known in advance and can be introduced by the teacher. In addition, even if all strategies were known, introducing them in the knowledge base would take considerable time and effort. To reduce this time and effort, we propose a mechanism for learning strategies from user data through clustering in the context of an ELE for mathematical generalisation called *eXpresser* [6], which results in a number of representative strategies that can be presented to the teacher for further authoring or storing in the knowledge base.

The next section briefly introduces *eXpresser* and gives examples of mathematical generalisation tasks. Section 3 presents the mechanism we propose for learning strategies from user data. Experimental results using data from a classroom session are presented in Section 4. Section 5 discusses the results and concludes the paper.

II. The Exploratory Learning Environment

eXpresser [6] is an ELE for the domain of mathematical generalisation; it is designed for 11-14 year olds and for classroom use. The tasks involve building a construction and deriving an algebraic-like rule from it.

Two typical tasks, 'pond tiling' and 'footpath' are described below. 'Pond tiling' requires to find a general rule for surrounding any rectangular pond. The construction, several ways of building it and their corresponding rules are displayed in Figure 1; the variables w and h refer to the width and the height of the pond. The 'footpath' task requires to build a construction such as in Figure 2(a) and to find a rule for the green (lighter colour) tiles in relation to the red (darker) tiles, i.e. the footpath; some blocks of the construction are expanded for ease of visualisation; the variable red refers to the number of red tiles. In these figures, the internal structure of the constructions has been highlighted for clarity. In *eXpresser* all constructions would look the same in the normal course of the task.

Each construction is called a *strategy* and is made of several *patterns*. For example, the construction in Figure 2(c) is made of 4 patterns: two green (lighter colour) patterns made of 7 tiles with no gaps between them, which are placed at the top and the bottom of the construction; one green pattern made of 4 tiles with gaps of one tile between them, and one red (darker colour) pattern made of 4 tiles with gaps of one tile between them.

The strategies above are illustrated using one particular instance for each task, i.e. a 'pond' of width 5 and height 3, and a 'footpath' of 3 tiles; however, learners build constructions of various dimensions corresponding to different instances of the task. The goal is to build a construction that is *general*, i.e. it is correct for *any* instance of the task. To verify if their constructions is general, the system allows the learners to animate their construction by varying the values of the

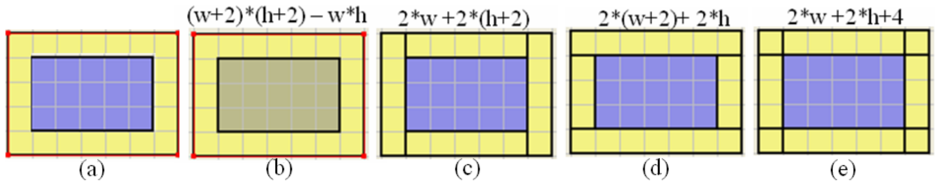

Fig. 1. ‘Pond tiling’ task, constructions and associated rules.

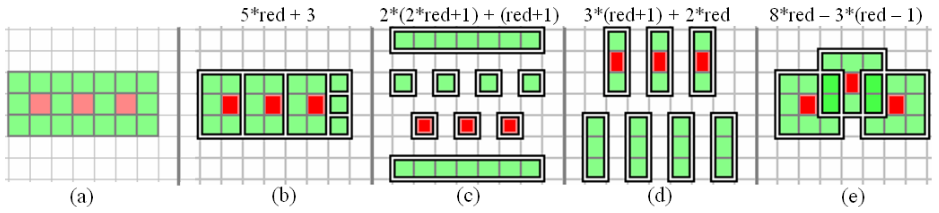

Fig. 2. ‘Footpath’ task, constructions and associated rules.

variables involved in building the construction (e.g. w and h for the pond tiling task, and red for the footpath task).

To enable intelligent support while learners work on mathematical generalisation tasks, a knowledge base of strategies (i.e. correct constructions) for each task is needed to diagnose the students' progress when solving a task [7]. The next section outlines our approach for learning these strategies from user data to reduce the teachers' effort on authoring.

III. Learning Strategies from User Data through Clustering

The proposed mechanism for learning strategies from user data based on clustering [8] involves several steps, which are detailed below:

1) Calculate a similarity distance between the constructions of all users. For this purpose, the constructions are represented as a sequence of vectors, where each vector corresponds to a pattern;
2) Create a vector with the distances between the constructions of all users;
3) Perform clustering using the vector from Step 2;
4) Choose a representative for each cluster.

Each pattern is represented as a vector of numerical values corresponding to the following properties: units, move right, move down, colour allocations and a group flag. For example, in Figure 2(b), the pattern that looks like a C-shape with a red tile in the middle, is represented as the following vector: (3, 2, 0, 15, 3, 1). More specifically, the pattern is repeated 3 times, for each repetition it moves 2 tiles to the right and 0 down (i.e. it stays on the horizontal axis), it has 15 green tiles and 3 red tiles, and it is a group because the 5 green tiles in the form of a C-shape were grouped together with one red tile. This group which is repeated 3 times is called the basic unit of a pattern. A pattern, however, does not necessarily need to be repeated, i.e. a single tile is still a pattern and in this case the pattern is the same as its basic unit.

A construction typically includes several patterns and is represented as a sequence of vectors corresponding to the patterns. Using the example started above, the construction in Figure 2(b) will be represented as a sequence of 4 vectors:(3, 2, 0, 15, 3, 1), (1, 0, 0, 1, 0, 0), (1, 0, 0, 1, 0, 0) and (1, 0, 0, 1, 0, 0) (the last 3 vectors each correspond to a single green tile). The sequence is represented in the order in which the patterns were constructed by the user.

To calculate the similarity between each construction, the following procedure is used:

Reduce each construction to the smallest instance of the task by adjusting the values for the units and the colour allocations. This is necessary because users use different instances of the task and we are interested in the structural similarity rather than the exact dimensions of the construction. The structure of a construction is important because it defines the strategy used regardless of the instance of the task. The algorithm for reducing a construction to the smallest instance of the task for the ‘footpath task’ is given below (Algorithm 1); some examples are given in the next section. This algorithm depends on the task and we chose to present the one for the ‘footpath task’ because the experimental results use data related to this task.

Algorithm 1 ConstructionResize($Constr$)

```
$instance$ = no of red tiles
for all vectors in $Constr$ do
    if the vector contains red tiles then
        $units$ = initial units value
        $allocations$ = initial colour allocations value
        change units value to 1
        change colour allocations value to $allocations/units$
    end if
    if the vector does not contain red tiles then
        if units value $\neq 1$ then
            $units$ = initial units value
            $allocations$ = initial colour allocations value
            change units value to $newUnits = units/instance + remainder(units/instance)$
            change colour allocations value to $allocations/units \times newUnits$
            {$allocations/units$ defines the value of the allocations per unit, which is then multiplied with the new value for the units}
        end if
    end if
end for
return $Constr$
```

Use a greedy approach to determine the order in which the vectors are compared. This is important because users do not construct the patterns in their construction in the same order even if they use the same strategy. For example, the construction in Figure 2(c) can be build in 24 different ways. If the patterns order is ignored, the similarity between 2 constructions using the same strategy would not be accurate unless the order in which they were build is the same. To avoid this problem, we use a greedy approach using Algorithm 2 illustrated bellow.

As different constructions have different numbers of vectors, the first step in Algorithm 2 is to find which is the construction with the fewer vectors. Let's consider two constructions $ConstrA$ and $ConstrB$, and that $ConstrA$ has fewer vectors, and more specifically z vectors. The algorithm will return two ordered constructions of size z. In other words, the ordered $ConstrA$ has the same z vectors, while the ordered $ConstrB$ will have its number of vectors capped at z.

To compare the vectors, the Euclidian distance is used: $D = \sqrt{\sum_{j=1}^{6}(\alpha_{V_j} - \alpha_{W_j})^2}$. V and W are vectors from two users' constructions (V from $ConstrA$ and W from $ConstrB$) and α represents an attribute of the vector, e.g. α_{V_j} stands for the jth attribute in vector V.

Algorithm 2 Order($Constr1$, $Constr2$)

```
Find which construction has fewer vectors
$ConstrA \leftarrow$ construction with the fewer vectors;
$ConstrB \leftarrow$ construction with more vectors;
repeat
    Compare 1st vector of $ConstrA$ with all vectors of $ConstrB$ and select the most similar one
    Store 1st vector of $ConstrA$ in $ConstrX$
    Store the most similar vector of $ConstrB$ in $ConstrY$
    Remove 1st pattern of $ConstrA$ and the most similar pattern from $ConstrB$
until $ConstrA$ is empty
return $ConstrX$, $ConstrY$
```

Calculate the similarity between the constructions returned by Algorithm 2. For this purpose, the following aggregated measure is used:

$$Sim = \begin{cases} \frac{z}{\sum_{i=1}^{z} D} & \text{if } \sum_{i=1}^{z} D \neq 0 \\ 0 & \text{if } \sum_{i=1}^{z} D = 0, \end{cases}$$

where z is the number of vectors in the construction with the fewer vectors from the two constructions that are compared. In this way the number of vectors compared is included in the similarity metrics. As the number of vectors of a strategy is part of its structure, it is important to have it included in the similarity metric.

The similarity metric is calculated for all pairs of constructions, i.e. $(n-1) \times n/2$ pairs, where n is the number of constructions (which also corresponds to the number of users).

In the next step, a vector Y of length $(n-1) \times n/2$ is created that includes the distances between constructions of all users in the following order: $(1,2), (1,3), ..., (1,n), (2,3), ..., (2,n), ..., ..., (n-1,n)$. This vector is then used to perform clustering. For this purpose Matlab is used and a hierarchical cluster tree is created using the single linkage algorithm: $Z = linkage(Y)$. Clusters are then created based on the natural division in the data (i.e. no pre-specified number of clusters), using $T = cluster(Z, \text{'cutoff'}, \text{cutoff})$, where $0 < \text{cutoff} < 2$. To verify the cluster tree, the $cophenet(Z,Y)$ function is used; it outputs a value below 1 and its meaning is that its value should be close to 1 for a high quality solution.

To choose a representative for each cluster the procedure described in Algorithm 3 is followed. Consequently, the construction that is most similar to all constructions in the cluster is chosen as a representative. This approach was used because unlike other applications where the centroid of a cluster (the average of all the points in the cluster) is used as a representative, in our case, the centroid does not reflect the nature of the task as it would represent a synthetic construction that none of the users has built.

Algorithm 3 Representative($Cluster$)

```
for all $Construction_i$ in $Cluster$ do
    calculate the distance between $Construction_i$ and all other constructions in $Cluster$
    calculate the average of the distances computed at the previous step
end for
$Constr$ = the construction with the smallest average
return $Constr$
```

IV. Experimental Results

The approach presented in the previous section was tested using data from a classroom session where 18 students used *eXpresser* to solve the 'footpath' task. Out of the 18 learners, 14 completed a construction while the other 4 did not. This was verified by matching the learners' constructions to the mask of the task. Therefore, our test was performed on 14 constructions, with one construction per user; their distribution and the corespondent figure references are displayed in Table I. The learners' constructions were identified as following the strategies in Table I by experts in mathematical generalisation.

TABLE I
User distribution in terms of strategies used

No of users	Strategy name	Figure reference
6	'C' strategy	Figure 2(b)
4	'HParallel' strategy	Figure 2(c)
2	'VParallel' strategy	Figure 2(d)
1	'Squares' strategy	Figure 2(e)
1	Combination of 'HParallel' and 'VParallel' strategies	Figure 3

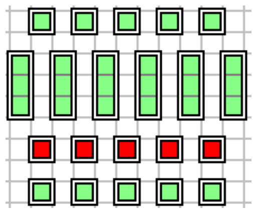
Fig. 3. Combination of 'HParallel' and 'VParallel' strategies.

The first step in our procedure is to reduce all constructions to the smallest instance of the task. For the 'footpath' task that the learners solved in this session, the smallest instance corresponds to one red tile. To reduce all constructions to this instance Algorithm 1 is used. To illustrate how the algorithm works, we will explain in detail the modifications for the constructions illustrated in Figure 2(b) and Figure 2(d).

The construction in Figure 2(b) is made of four vectors. The first one is (3, 2, 0, 15, 3, 1) and it will be modified to (1, 2, 0, 5, 1, 1). This is done by replacing the initial value of the units, i.e. 3, with value 1, and by replacing the colour allocations with $15/3 = 6$ and $3/3 = 1$, where 15 and 3 are the initial values of the colour allocations, and the divisor 3 is the initial value of the units. The other three vectors stay the same as their units value was already 1.

The construction in Figure 2(d) is made of two vectors, one that includes red and green tiles and one that has only green tiles. The vector with the red tiles is modified from (3, 2, 0, 6, 3, 1) to (1, 2, 0, 2, 1, 1) by changing the value of the units from 3 to 1 and the value of the colour allocations from 6 and 3, to $6/3 = 2$ and $3/3 = 1$, respectively. The vector with only green tiles is modified from (4, 2, 0, 12, 0, 1) to (2, 2, 0, 6, 0, 1). The units are changed from 4 to 2 using $4/3 + remainder(4/3) = 1 + 1 = 2$ and the colour allocations are modified from 12 to 6 using $12/4 \times 2 = 3 \times 2 = 6$.

The next step is to find the optimal order for comparing the constructions. For this purpose, Algorithm 2 was used for all pairs of constructions, i.e. 91 pairs. The next step was to construct the vector Y which includes the distances between constructions of all users. This is displayed in Table II in a matrix format rather than vector format for ease of visualisation. For the same purpose, the values correspond to the constructions in the order they were mentioned in the distribution table (Table I), i.e. the first 6 constructions follow the 'C' strategy, the next 4 follow the 'HParallel' strategy, the next 2 follow the 'VParallel' strategy, the next one follows the 'Squares' strategy and finally, the last one follows a combination of 'HParallel' and 'VParallel' strategies.

A hierarchical cluster tree is created using the single linkage algorithm: $Z = linkage(Y)$. A dendogram plot of the binary cluster tree was generated which is displayed in Figure 4.

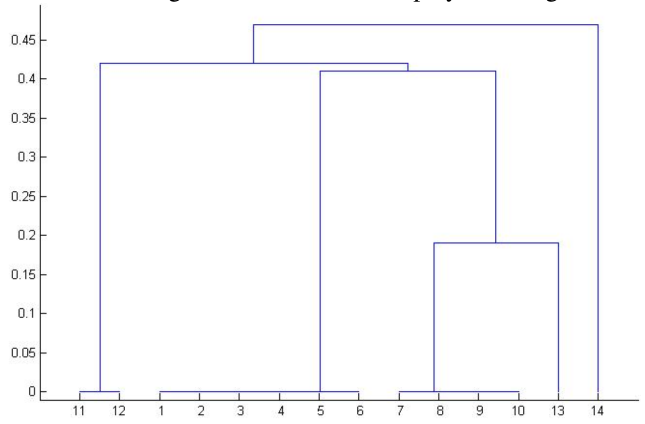

Fig. 4. The dendogram of the hierarchical clustering.

The dendogram clearly displays the five categories that were mentioned in the distribution table, i.e. Table I. To test the validity of the solution obtained, the *cophenet* function is used. This function indicates how the data fits in the structure suggested by the cluster tree. The value obtained was 0.9182, which indicates that the data fits the structure well and that that the solution obtained is of high quality.

The next step is to obtain the clusters. Depending on the value of cutoff, a different number of clusters is produced. For example, using the value of 0.5 for cutoff, five clusters are obtained as displayed in Table III. These five clusters correspond to the categories presented at the beginning as identified by experts.

When a value of 0.8 is used for cutoff, four clusters are obtained. They are displayed in Table IV. The difference from the previous clustering results is that construction 13 instead of constituting a cluster by itself has been attached to the cluster formed by constructions 7 to 10.

TABLE III
CLUSTERS USING CUTOFF = 0.5

Cluster number	Members
1	13
2	1, 2, 3, 4, 5, 6
3	11, 12
4	14
5	7, 8, 9, 10

TABLE IV
CLUSTERS USING CUTOFF = 0.8

Cluster number	Members
1	1, 2, 3, 4, 5, 6
2	11, 12
3	14
4	7, 8, 9, 10, 13

From a teacher's point of view the clusters still reflect the learners' data. One particular approach (corresponding to construction 13 and to the 'Squares' strategy), however, is classified as part of a cluster that includes one other approach (the 'HParallel' strategy) that was used by four learners (7 to 10 in our distribution). In other words, if these clusters were used further and a representative for each cluster is presented to the teacher, s/he would not be aware of the fact that one learner has used an approach that is not reflected in the results. Some would not consider this as a problem, as only one learner used that approach, therefore, not being representative for the whole group of learners. On the other hand, it could be argued that the approach is still a valid one and should be presented to the teacher and stored in the knowledge base or that the decision to store it should be left for the teacher to make. This aspect is further discussed in Section V.

To calculate the representatives for clusters, the minimal average distance between all constructions within the cluster is used, as in Algorithm 3. For example in the last situation with the 4 clusters, the last cluster has the constructions of users 7, 8, 9, 10 and 13 and the average distances are as displayed in Table V.

TABLE V
AVERAGE DISTANCES IN CLUSTER 4

Construction	Average distance
7	0.0475
8	0.0475
9	0.0475
10	0.0475
13	0.19

Therefore, the representative could be any of the constructions 7, 8, 9 or 10. In fact, users 7, 8, 9 and 10 use the same strategy, while user 13 uses a different one. This explains why constructions 7 to 10 have the same average distance to all other constructions in the cluster.

V. DISCUSSION AND CONCLUSIONS

The experimental results presented above show that the clustering mechanism performs well, leading to clusters that are relevant in the context of the task. Depending on the cutoff, a different number of clusters is obtained and both a lower or a higher value could be argued as advantageous. The lower value for the cutoff leads to a higher number of clusters, allowing

TABLE II
THE DISTANCED BETWEEN THE CONSTRUCTIONS OF ALL USERS

	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.00	0.00	0.00	0.00	0.00	0.41	0.41	0.41	0.41	0.67	0.67	0.50	0.67
2		0.00	0.00	0.00	0.00	0.41	0.41	0.41	0.41	0.67	0.67	0.50	0.67
3			0.00	0.00	0.00	0.41	0.41	0.41	0.41	0.67	0.67	0.50	0.67
4				0.00	0.00	0.41	0.41	0.41	0.41	0.67	0.67	0.50	0.67
5					0.00	0.41	0.41	0.41	0.41	0.67	0.67	0.50	0.67
6						0.41	0.41	0.41	0.41	0.67	0.67	0.50	0.67
7							0.00	0.00	0.00	0.42	0.42	0.19	0.47
8								0.00	0.00	0.42	0.42	0.19	0.47
9									0.00	0.42	0.42	0.19	0.47
10										0.42	0.42	0.19	0.47
11											0.00	0.50	0.89
12												0.50	0.89
13													0.50

to distinguish constructions even when only one of that type has been produced. On the other hand, if there are many such unique constructions, a higher value for the cutoff will ensure that fewer clusters are produced and the teacher does not need to spend time on strategies that are not frequently used by learners. This issue could be solved by allowing the teacher to manipulate the value of cutoff, observe the difference in output, and choose a particular set of results as a starting point for his/her decision to author the strategies further and/or store them in the knowledge base.

Besides judging the clusters according to the known distribution of the constructions build by the users, another measure that validates the results is the *cophenet* function [9], which measures how faithfully a dendrogram preserves the pairwise distances between the original unmodeled data points. The value that was obtained, i.e. 0.9182, indicates that the data fits well into the structure suggested by the cluster tree. This value is independent of the number of clusters and is calculated for the cluster tree that results from the single linkage algorithm. Therefore, it is an indication of the quality of the cluster tree rather than the quality of the different clusters obtained for different values of cutoff.

The representative of each cluster is calculated using the minimal average distance, as mentioned in Section III, because the average of all constructions in the cluster does not make sense for our particular application. In the experimental results, most clusters contained only constructions that were built using the same strategy. For example, when 0.5 was used for the value of cutoff, five clusters were obtained each with constructions following the same strategy, i.e. the construction in cluster 1 followed the 'Squares' strategy, the six constructions in cluster 2 followed the 'C' strategy, the two constructions in cluster 3 followed the 'VParallel' strategy, etc. Similarly, when using 0.8 for the value of cutoff, three out of the four clusters were composed of constructions following the same strategy. The only cluster that had constructions belonging to more than one strategy was the fourth cluster obtained when using 0.8 for the value of cutoff, which was used to illustrate the algorithm for selecting a cluster representative.

The cluster representative is important because it is presented to the teacher for further authoring or for storing in the knowledge base. Consequently, in the context of our particular application, the teacher should be presented with actual constructions built by learners rather than the constructions that were numerically transformed for the purpose of performing the clustering.

Apart from the fact that the centroids of clusters are not relevant for our application, there is another aspect that is different in our case compared with other clustering applications. This refers to the dimensions of the data used for clustering. Typically, all data points have the same dimensionality. In our case, however, the data varies in dimensionality, i.e. different constructions have different numbers of vectors or patterns. This is due to the domain we are working with and in particular, to the fact that each generalisation task can be solved using several different strategies.

The research presented in this paper shows that clustering can be used to learn strategies from user data, which could reduce the effort needed from teachers to introduce task-specific strategies in the knowledge base. Future work includes developing an interface where the teacher can easily check the output of the clustering mechanism and try different outputs by varying the value of cutoff.

REFERENCES

[1] S. Papert, *Mindstorms: children, computers and powerful ideas*. BasicBooks, New York, 1993.

[2] C. Lynch, K. Ashley, V. Aleven, and N. Pinkwart, "Defining "ill-defined domains"; a literature survey," in *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th ITS Conference*, 2006, pp. 1–10.

[3] T. de Jong and W. R. van Joolingen, "Scientific discovery learning with computer simulations of conceptual domains," *Review of Educational Research*, vol. 68, no. 2, pp. 179–202, 1998.

[4] P. Kirschner, J. Sweller, and R. E. Clark, "Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential and inquiry-based teaching," *Educational Psychologist*, vol. 41, no. 2, pp. 75–86, 2006.

[5] M. Cocea and G. D. Magoulas, "Hybrid model for learner modelling and feedback prioritisation in exploratory learning," *International Journal of Hybrid Intelligent Systems*, vol. 6, no. 4, pp. 211–230, 2009.

[6] R. Noss, A. Poulovassilis, E. Geraniou, S. Gutierrez-Santos, C. Hoyles, K. Kahn, G. D. Magoulas, and M. Mavrikis, "The design of a system to support exploratory learning of algebraic generalisation," *Computers & Education*, vol. 59, no. 1, pp. 63 – 81, 2012.

[7] M. Cocea, S. Gutierrez-Santos, and G. Magoulas, *Innovations in Intelligent Machines - 2*, ser. Studies in Computational Intelligence. Springer, Berlin, 2011, vol. 376, ch. Case-based Reasoning Approach to Adaptive Modelling in Exploratory Learning, pp. 167–184.

[8] C. Romesburg, *Cluster Analysis for Researchers*. Lulu Press, North Carolina, 2007.

[9] R. R. Sokal and F. J. Rohlf., "The comparison of dendrograms by objective methods," *Taxon*, vol. 11, pp. 33–40, 1962.