Computer Communication & Collaboration (Vol. 6, Issue 4, 2018)
ISSN 2292-1028(Print) 2292-1036(Online)
DOIC:2292-1036-2018-11-001-95

Analysis and Classification of Programming Exercises by Graph Clustering for Recognition of Model Solutions

Márcia G. Oliveira

Reference Center in Formation and Distance Education (Cefor), Federal Institute Espirito Santo (Ifes)

30 Barão de Mauá st, Vitória, Espirito Santo, Brazil.

E-mail: marcia.oliveira@ifes.edu.br

Howard Roatti

FAESA Centro Universitário

2220 Vitória Avenue, Vitória, Espírito Santo, Brazil

E-mail: howardcruzroatti@gmail.com

Elias S. Oliveira

Graduate Program on Informatic (PPGI), Federal University of Espirito Santo (UFES)

514 Fernando Ferrai Avenue, Vitória, Espírito Santo, Brazil

E-mail: elias_oliveira@acm.org

Abstract: Computer programming is a cognitive and formal problem solving process that can involve many possible solutions. Thus, manual evaluation of programming exercises is an onerous task, in particular in the case of numerous exercises and programming classes with many students. Once the assessment is automated, the effort put forth by teachers can be reduced; however, he should consider all possible solutions for each exercise to create model solutions or to train automatic assessment systems. In order to assist teachers in analyzing programming exercise solutions, this paper proposes a strategy based on clustering and LSA (Latent Semantic Analysis) techniques to identify classes of solutions that represent rubrics and automatically sort based on score the majority of the sets of exercise solutions. The results of the first experiments indicate the ability of this strategy to identify solutions classes and to automatically classify the best solutions.

Keywords: Analysis of Exercises, Clustering, Programming, Rubrics.

1 Introduction

Computer programming is a complex process that formally specifies the steps involved in solving a problem in a logical sequence for machine processing. However, this process can result in several solutions to the same problem. Thus, manually evaluating a large amount of exercises, particularly in large classes, is an onerous activity, as teachers must devote a great deal of time and effort to

analyzing all possible solutions for each exercise.

Automatic assessment can help reduce the effort that is needed for evaluation; however, to imitate the evaluation patterns of a teacher, such a system requires examples that have been evaluated so that it is possible to compose model solutions or train systems for automatic assessment.

In the first case, to assess the effectiveness of the system, the teacher must know all possible solutions for each exercise. In the second case, in general, a teacher must manually evaluate for each exercise at least 50% of a solution set to train an automatic assessment system. Given these two possibilities for automatic assessment of programming exercises in MOOCs (Massive Open Online Courses), reducing the evaluation effort remains a major challenge.

In other areas, to reduce the human effort involved in training supervised learning systems, some strategies have been proposed to select a smaller and more representative set of training samples. Selective Sampling [1] and the Active Learning [2] are some examples. In the programming domain, a suggestion for recognizing a variety of solutions is to apply clustering algorithms [3]. Following this idea and applying the strategy of [2] to reduce the evaluation effort of teachers, this paper proposes a system of analysis, selection and classification of programming solutions.

The proposed system combines clustering and LSA techniques to select the most representative samples of each cluster for the teacher assigning scores to them and, based on these evaluated samples, automatically classifies the samples that are most similar to them to form the training set of an automatic assessment system.

The first results in real programming exercises indicate the effectiveness of the proposed strategy in identifying model solutions such as rubrics as well as precisely classifying the best solutions in a dataset of programming exercises. In conclusion, our strategy represents an important contribution to because it analyzes the relationship between the features of the codes developed by students and the similarity between patterns to recognize the best solutions in homogeneous clusters. By recognizing these solutions, we help teachers to identify good indicators of programming learning. Thus, the model solutions can also be used as references to inform good programming practices.

This paper is organized as follows. In Section 2, we present a literature review and related works. In Section 3, we describe the functional model of our strategy. In Section 4, we present and discuss the experimental results of the proposed method. In Section 5, we offer conclusions and ideas for future work.

2 Literature Review

A tendency to add more reliability to automatic evaluation mechanisms has been demonstrated by the use of rubrics [4]. According to [5], a rubric is a scoring tool for qualitative evaluation of various dimensions of student performance in learning activities. A scoring rubric contains several components including one or more dimensions for receiving scores and examples that illustrate the score scale for each dimension of performance [6].

Given that the reliability of performance evaluations can be improved by the use of rubrics [5], automatic evaluation mechanisms have adopted anchors, which complement rubrics with examples that illustrate various levels of learning [6]. Thus, anchors can be selected from students exercise solutions to set score criteria.

In order to address problems such as automatically selecting anchors or model solutions from the broad set of solutions, many technologies based on selective sampling [1] and active learning [7]

have been developed.

In the case of selecting representative samples, the objectives are to reduce the efforts of human experts to label training examples of supervised learning systems and to generate the largest and best representation of the domain to be learned by such systems.

Taking these objectives into consideration, [2] have proposed an active learning strategy that selects a minimum number of examples of Twitter texts for classification by a human specialist and then automatically classifies a large number of other tweets based on them.

With respect to identifying rubrics, the goal is to inform both the teacher and the student about the criteria of an assessment once a rubric is an instrument of scoring to qualitatively evaluate multiple dimensions of student performance [5].

An example of rubric composition presented by Olmos [8] is an automatic evaluation method for textual documents that applies the LSA (Latent Semantic Analysis) technique to identify and evaluate conceptual axes from samples of texts written by students.

Learning Analytics has also contemplated the field of computer programming and tools to support the assessment has been developed. An example is the technique to assess, analyze and visualize students learning computer programming proposed by [9]. This work presents developing metrics to analyze programming process that could support the formative assessment actions.

In order to evaluate programming learning, an initial step to identify models of solutions from programs developed by students in the field of computer programming is proposed by [10]. This proposal is based on clustering and classifies two representations into two clusters of model solutions: a representation of the best solutions and a representation of the diversity of solutions. The first representation is obtained from the more homogeneous cluster, and the second representation is obtained from a heterogeneous cluster that brings together more examples of programming solutions.

The two representations generated by the strategy of [10] may be used for the composition of items to inform evaluation criteria based on the classes of scores assigned to samples of these representations. The first representation of the samples, however, may be used as models for automatic evaluation templates based on jigs. Samples from the second representation, in turn, may provide a training set with higher representation of domain diversity for a semi-automatic evaluation system based on supervised learning.

However, the method proposed by [10] is limited because it performs various clustering settings to define an appropriate number of clusters, which offers a greater number of possibilities to form clusters with the best representations of the model solutions. Using the Graph Clustering Method in this work, we can overcome this limitation; we can set a minimum fixed number of clusters (1, for example), and the clustering algorithm is able to adjust this number as its perception of diversity evolves and as the need to generate more clusters to group different classes of solutions grows.

In summary, our method combines the strategy of [2] and the concept of [10] to recognize model solutions; it uses the LSA technique to identify latent factors from the relationship between the features from C Language Source codes developed by programming students and clusters the new representation of these codes to recognize model solutions to programming exercises, which indicates successes in programming learning.

In the future, our strategy to represent diversity could be improved to provide examples of pre-rated codes, which inform the evaluation criteria of teachers and train supervised learning systems with less possibility of overfitting in a classification process for programming solutions.

3 Selection of Solutions Model

Figure 1 shows the strategy for selecting model solutions for an automatic assessment system of supervised learning and for composing rubrics.

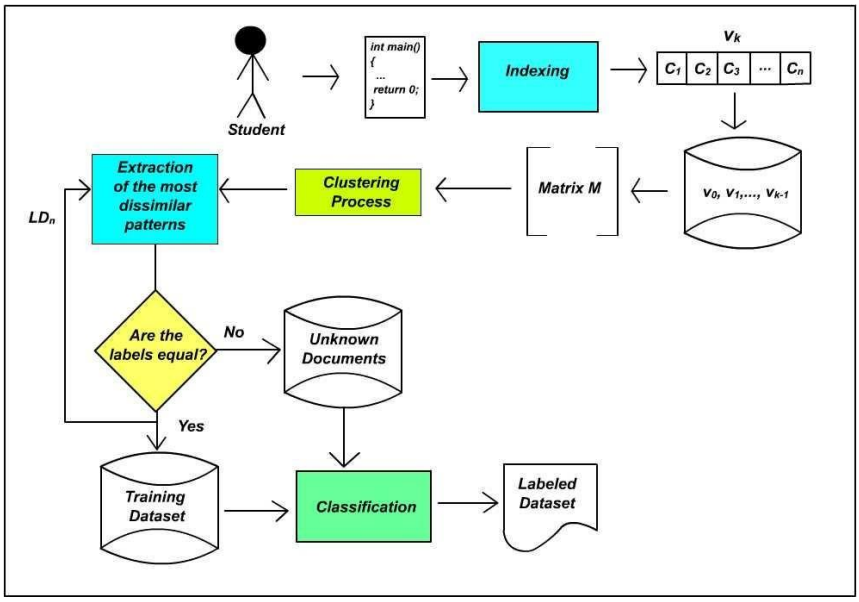

Figure 1 Selection of Model Solutions Strategy

According to Figure 1, the proposed method is based on the proposal of [2] and occurs in two phases. In the first phase, the documents are introduced into the algorithms that perform the Clustering Process. This process consists of grouping solutions of a programming exercise and mapping them in vectors in the Indexing phase. Each dimension of these vectors has C_i values representing the frequency of occurrence of keywords, symbols, conditional structures, loop structures, operators, functions of C language and logic variables, with values 1 (true) and 0 (False) indicating whether a program compiles and runs. In summary, we have mapped each C program in 60 features or C_i values.

In the next phase, Indexing, the vectors representing all the developed solutions for a programming exercise are combined in a Matrix M that is transformed by the LSA technique, which reduces its dimensionality. In this case, sets of related features are associated with 12 factors representing the relationship between these features.

In the Clustering Process, the vectors of M – which represent samples of programming solutions that are analyzed and the samples more similar to each other – are gathered in clusters. This is performed by the Graph Clustering Algorithm from the Cluto 2.1.2 software [11]. We have chosen this algorithm because the number of clusters is set by the clustering algorithm itself according to the following configuration parameters:

1. *-method=graph (Clustering Graph Algorithm)*
2. *-edgePrune=0.6 (Cut of patterns between two clusters with similarity index less than 60%)*
3. *-vtxPrune=-1 (Cut outliers. -1 means don't cut outliers)*
4. *-sim=cos (Cosseno similarity metric)*

After the Clustering Process, the vectors are classified in descending order for each cluster based on the average of internal similarities. In each cluster, the two most dissimilar solutions are

presented to the teacher who is assigning them scores. If the scores of these dissimilar solutions are identical, all other solutions in the cluster receive the same scores. At this point, there is a decision level 0, i.e., LD_0, as shown in Figure 1.

If the scores of the two selected solutions are different, these solutions are discarded, and within the same cluster, the second most dissimilar solution is selected. At this time, two solutions are presented to the teacher, who then gives them scores. If the assigned scores are identical, the entire cluster receives the same score. Otherwise, the entire cluster is discarded and inserted in the set Unknown Documents. This decision level is called LD_1.

It is important to observe that the process of Extraction of the Most Dissimilar Patterns can go farther the LD_1 level, but it requires more human effort to assign scores.

After the distribution of model solutions between the Training Dataset and the Unknown Documents set, the automatic assessment system is trained with the Training Dataset, and the solutions of the Unknown Documents receive scores in a process of Classification that generates as an output a Labeled Dataset.

4 Experiments and Results

In our experiments, we used a dataset containing 100 solutions for a programming exercise written by students from different programming classes at a university. This dataset, called DS-A, was indexed on 60 assessment variables C_i representing the Language C domain.

The scores were assigned by a teacher of programming to each sample of DS- A and were distributed in five classes, as shown in Table 1.

Table 1 Classes of Score

Class	Number of samples	Score range (%)
A	20	]80, 100]
B	46	]70, 80]
C	22	]60, 70]
D	10	]50, 60]
E	**2**	[0, 50]

Dataset DS-A

The M matrix was transformed through the LSA technique into a matrix M' of 12 dimensions that map the relationships between the variables of the Matrix M in factors.

Figures 2 and 3 show the graphics of clustering with the configuration parameters presented in Section 3. Figure 2 illustrates how the six clusters were formed, presenting the relations of similarity among their samples. The stronger the red color, the more similar to each other are the two samples of a cluster. The stronger the red color of a cluster, the more homogeneous it is.

According to Figure 2, the clusters were organized from the bottom up in descending order of similarity. Cluster 0, as in the graphs in Figures 2 and 3, is presented as the most homogeneous cluster. It is also observed that the samples of Class A predominate in Cluster 0. This confirms the assertion of Naudé [4], which states that programming exercise solutions with higher scores tend to be more similar.

In the clusters of Figure 3, the red color represents positive values, white color values equal to

zero and the green color represents negative values. Bright red/green indicate large positive/negative values [11]

Graph (B) in Figure 3 shows that Cluster 1 is the second most homogeneous cluster, and Class B lightly predominates in this cluster. In the remaining clusters, the classes appear more scattered. We therefore conclude it is most likely to find models of the best solutions in the most homogeneous clusters. In addition, through automatic classification, it is possible to set the classes of other samples of these clusters in LD_0 or LD_1 levels, which will require less evaluation effort by the human expert.

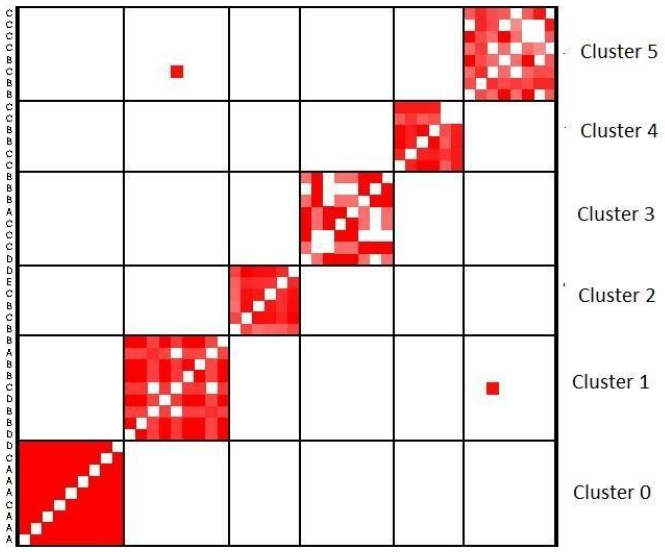

Figure 2 Analysis of Samples in Clusters

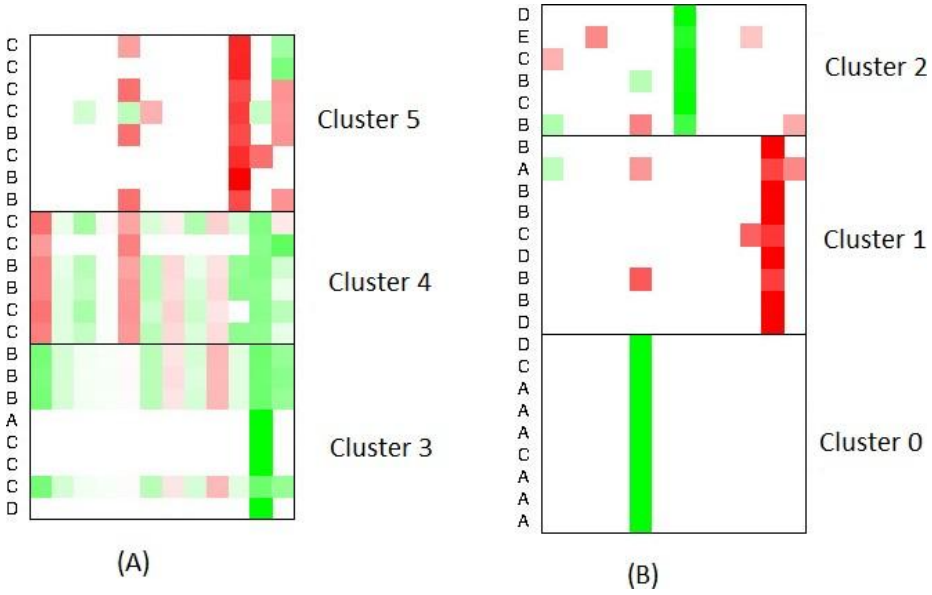

Figure 3 Analysis of Features in Clusters

4.1 Results

Tables 2 and 3 show the results obtained for the three classes with higher scores, i.e., the A, B and C classes in the LD0 and LD1 levels. In these tables, TP, FP and FN indicate the number of True

Positive, False Positive and False Negative classified samples. For each class, based on these values, error (err) and accuracy (acc) values are calculated according to the following mathematical expressions:

$$err(h) = \frac{1}{n}\sum_{i=1}^{n} ||y_i \neq h(x_i)|| \quad (1)$$

$$acc(h) = 1 - err(h) \quad (2)$$

It is worth emphasizing that after the process of reducing the specialist's efforts, the dataset has its elements separated in the training and testing sets. Therefore, each class is replaced by part of its sorted elements, and the remainder is carried to the classifier that labels them. Thus, the class is replaced by 12 elements to be sorted in LD_0 and 13 in the LD_1 level. In Class B, there are 31 in LD_0 and 32 in LD_1. Class C is replaced by 5 elements in LD_0 and 6 in LD_1.

In tables 2 and 3, TP (True Positive) is the number of documents that the human expert labeled as pertinent to a class and the algorithm also labeled as pertinent. FN (False Negative) is the number of documents that the human expert labeled as pertinent to a class and that the algorithm labeled as not pertinent. FP (False Positive) is the number of documents that the human expert labeled as not pertinent to a class and the algorithm labeled as pertinent.

Finally, the error (h), i.e., the error value in a class h, is calculated as (FP + FN) / N, where N is the total number of items in all classes.

In LD_0, for example, we have 12 items in Class A, 31 in Class B and 5 in Class C, totaling 48 items. Thus, for Class A, according to the results of Table 1, we have: FP = 1 and FN = 12, with a total of 13 errors. Thus, error(A) = 13/48 = 0.270833 and acc(A) = 1 - error(A) = 0.729167.

This same procedure is applied to the other classes to obtain the accuracy value in the LD_0 and LD_1 levels.

Table 2 Results of Automatic Assessment in the LD_0 Level

Class	TP	FP	FN	Accuracy	Recall	Precision
A	0	1	12	0.7291	0	0
B	20	4	11	**0.6875**	**0.6451**	**0.8333**
C	2	21	3	0.5	0.4	0.0869

Results per Class – Level LD_0

According to Table 2, in the LD_0 level, the patterns of Class B had better results according to the Recall and Precision metrics. There were no TP classifications in LD_0 for Class A. Thus, if there were different samples of classes in the same cluster, they should be selected as one of the most dissimilar samples, and possibly the best classification would occur in LD_1, as we can see in the TP for class A in the LD_1 level in Table 3.

Table 3 Results of Automatic Assessment in the LD_1 Level

Class	TP	FP	FN	Accuracy	Recall	Precision
A	9	6	3	**0.82**	**0.75**	**0.6**
B	14	4	18	0.56	0.4375	0.7777
C	3	14	3	0.66	0.5	0.17

Results per Class – Level LD_1

The classification results can be improved if the selection of training samples of the automatic assessment system have a better representation of the diversity of solutions. In our work, this did not occur because in the LD_0 and LD_1 levels, in general, the internal samples of clusters are labeled with the classes that have the best scores. As the solutions labeled with low scores tend to differ, they rarely enter the training set. Thus, the ability to generalize decisions is reduced in the automatic

evaluation system.

The advantages of our strategy for the automatic assessment systems are not in the automatic classification; once the good solutions are properly classified, there is still the possibility of model overfitting because of the low amount of diversity in the selected samples for the training set. However, the samples that are selected for evaluation by a teacher can be used as model solutions because they are chosen in each cluster as being the most dissimilar samples. These solutions can, therefore, form representations of rubrics and explain scores based on solved examples of good solutions; this can help students in their programming learning process.

5 Conclusion

This paper presented a strategy of analysis and classification of solutions for programming exercises by identifying model solutions and by training automatic assessment systems.

The results indicate that the proposed system effectively recognizes model solutions and automatically classifies the best solutions, requiring less effort in analyzing solutions and assigning scores. However, the system is still incapable of forming sufficiently heterogeneous clusters to train an automatic assessment system and to classify the solutions associated with medium and low scores.

For future work, we propose improving the formation of clusters to make them more heterogeneous and to select training samples to represent the variety of solutions for programming exercises, which increases the ability of an automatic assessment system to generalize, thus preventing model overfitting. In conclusion, the main contributions of this work are a mechanism of analysis of C Language Source codes that recognizes the best solutions, which can help teachers to inform their assessment criteria and help students to obtain more qualitative feedback.

Acknowledgments: We thank the Foundation for Support to Research and Innovation of Esp ŕito Santo (FAPES) for the support given to the project “Semi-Automatic Learning Assessment Technologies” (from the EDITAL 006/2014 - Universal Individual Research Project) and to the Graduate Research and Graduate School of Ifes for their support of the project "Analysis of Programming Learning by Pattern Recognition Technologies". This publication is the result of the execution of these projects.

References

[1]. Lindenbaum M, Markovitch S, Rusakov D. Selective Sampling for Nearest Neighbor Classifiers. Mach Learn. 2004;

[2]. Oliveira E, Gomes Basoni H, Rodrigues Sa úde M, Marques Ciarelli P. Combining Clustering and Classification Approaches for Reducing the Effort of Automatic Tweets Classification. In: Proceedings of the International Conference on Knowledge Discovery and Information Retrieval. 2014.

[3]. Naud é KA, Greyling JH, Vogts D. Marking student programs using graph similarity. Comput Educ. 2010;

[4]. Gwon H-G, Jo M-H, Lee E-J. Design and implementation of the automatic rubric generation system for the neis based performance assessment using data mining technology. Educational research review 2005;9.

[5]. Jonsson A, Svingby G. The use of scoring rubrics: Reliability, validity and educational

consequences. Educational Research Review, 2(2):130–144.2007.

[6]. Perlman C. Performance Assessment : Designing Appropriate Performance Tasks and Scoring Rubrics. In: Report - US Department of Education. 2003.

[7]. Tuia D, Pasolli E, Emery WJ. Using active learning to adapt remote sensing image classifiers. Remote Sens Environ. 2011;

[8]. Olmos R, Jorge-Botana G, Luzón JM, Martín-Cordero JI, León JA. Transforming LSA space dimensions into a rubric for an automatic assessment and feedback system. Inf Process Manag. 2016;

[9]. Blikstein P. Using learning analytics to assess students' behavior in open-ended programming tasks. In: Proceedings of the 1st International Conference on Learning Analytics and Knowledge - LAK '11. 2011.

[10]. Oliveira MG De, Reblin LL, Oliveira E. Sistema de Apoio à Avaliação de Atividades de Programação por Reconhecimento Automático de Modelos de Soluções. XXIV Work sobre Educ em Comput - WEI An do XXXVI Congr da Soc Bras Comput – CSBC. 2016;

[11]. Karypis G. CLUTO - A Clustering Toolkit. University of Minnesota, Department of Computer Science. 2003.