Reconhecimento Automático de Representações de Rúbricas em Agrupamentos de Soluções de Exercícios de Programação

Márcia G. de Oliveira [1], Leonardo Leal Reblin [2], Mateus Batista de Souza [2], Elias Oliveira [3]

[1] Centro de Referência em Formação e EaD (Cefor)
Instituto Federal do Espírito Santo

[2]Departamento de Engenharia Elétrica
Universidade Federal do Espírito Santo

[3]Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo

clickmarcia@gmail.com, elias@acm.org

Abstract. *The assessment of programming exercises is a complex process because for each exercise that a teacher applies, it is common to have several potential solutions. As not always the teacher knows all the possible solutions of an exercise, it is always a challenge for him to justify all the criteria of their assessment. In order to support the program evaluation process, this paper proposes a strategy based on clustering techniques and Principal Component Analysis (PCA) to recognize, from solutions developed by students, examples of solutions that represent, in a rubrics scheme, the scores assigned by a teacher. The results of experiments on real solutions of programming exercises indicate that our method recognizes solutions with high, medium and low scores requiring a little assessment effort of teachers.*

Keywords*: Clustering, PCA, Programming, Rubrics.*

Resumo. *A avaliação de exercícios de programação é um processo complexo porque para cada exercício que um professor aplica é comum haver várias possibilidades de soluções. Como nem sempre o professor conhece todas as possíveis soluções de um exercício, é sempre um desafio para ele justificar todos os critérios de sua avaliação. Com o objetivo de apoiar o processo de avaliação de programação, este trabalho propõe uma estratégia baseada em técnicas de clustering e de Análise de Componentes Principais (PCA) para reconhecer, a partir de soluções desenvolvidas por alunos, exemplos de soluções que representem, em um esquema de rúbricas, os escores atribuídos por um professor. Os resultados dos experimentos em soluções reais de exercícios de programação indicam que o nosso método reconhece soluções com escores altos, médios e baixos demandando pouco esforço de avaliação de professores.*

Palavras-chave*: Clustering, PCA, Programação, Rúbricas.*

1. Introdução

A avaliação de exercícios de programação é um processo complexo que demanda muito esforço de análise de professores porque para cada exercício pode haver várias possibilidades de soluções. Como nem sempre é possível o professor conhecer todas as possíveis

DOI: 10.5753/cbie.sbie.2016.1106

soluções, ele quase sempre precisa analisar minuciosamente cada solução, compreender como o aluno a desenvolveu e atribuir-lhe um escore informando a corretude da solução. Dessa forma, sendo baseada em decisões subjetivas, a avaliação de programação torna-se um desafio para o professor, especialmente quando ele precisa justificar seus critérios de avaliação e evitar muitas reclamações de alunos na entrega de resultados de avaliação.

Para melhor justificar seus critérios de avaliação, o professor poderia selecionar exemplos representativos de cada classe de escore a partir das soluções já corrigidas. Mas essa tarefa torna-se onerosa quando há muitas soluções de muitos alunos a serem avaliadas, o que torna necessário um tratamento automático do problema.

Em muitos trabalhos científicos, problemas similares ao problema de selecionar exemplos representativos de classes de desempenhos a partir de um conjunto de soluções desenvolvidas por alunos são tratados como problemas de amostragem seletiva [Lindenbaum et al. 2004], de aprendizagem ativa [Tuia et al. 2011, Oliveira et al. 2014] e de composição de rúbricas [Kwon and Jo 2005].

Com o objetivo de auxiliar professores na avaliação de exercícios de programação através da composição de rúbricas que informem critérios de avaliação, propomos uma estratégia de reconhecimento automático de modelos de soluções de programação a partir de soluções desenvolvidas por alunos.

Nessa estratégia, o reconhecimento de exemplos que representam gabaritos e a diversidade de soluções desenvolvidas para um exercício de programação é realizado através da combinação das técnicas de Análise de Componentes Principais (PCA) e de *clustering*. Através de um algoritmo de PCA, reduzimos a dimensionalidade da matriz que reúne os vetores de cada solução de um exercício de programação, tornando possível uma melhor caracterização da diversidade de soluções. Após os processos de *clustering*, identificamos dentro dos *clusters* mais homogêneos soluções com os escores mais altos ou mais baixos e, dentro dos mais heterogêneos, a diversidade de soluções.

Para capturar interativamente a diversidade de soluções, o professor deve atribuir escores às soluções selecionadas até que tenham sido avaliados, em um esquema de rúbricas, exemplos representativos de escores baixos, médios e altos que melhor representem os critérios de avaliação desse professor.

A grande vantagem da estratégia proposta é, portanto, oferecer aos estudantes de programação maior clareza do processo avaliativo fornecendo-lhes um conjunto de soluções explicadas a partir de diferentes escores atribuídos por professores.

Este trabalho está organizado conforme a ordem a seguir. Na Seção 2, destacamos os trabalhos relacionados que oferecem mecanismos de avaliação baseados em rúbricas. Na Seção 3, descrevemos a estratégia proposta de reconhecimento de representações de rúbricas bem como as técnicas e algoritmos utilizados. Na Seção 4, apresentamos os experimentos realizados e os resultados alcançados. Na Seção 5, concluimos com as considerações finais e trabalhos futuros.

2. Trabalhos Relacionados

A estratégia de composição de rúbricas para informar critérios de avaliação de professores de programação baseia-se nas ideias de *scoring rubric* de [Perlman 2003], na combinação das técnicas de *clustering* e de redução de dimensionalidade para avaliar exercícios de

programação de [Oliveira et al. 2015a] e no método de seleção de modelos de soluções de programação de [Oliveira et al. 2016].

Uma tendência para agregar mais confiabilidade a mecanismos de avaliação automática tem sido o uso de rúbricas [Kwon and Jo 2005]. De acordo com [Jonsson and Svingby 2007], uma rúbrica é uma ferramenta de *scoring* para avaliação qualitativa de várias dimensões de desempenhos de estudantes em atividades de aprendizagem. Um *scoring rubric* contém vários componentes que incluem uma ou mais dimensões que recebem escores e exemplos que ilustram a escala de escores para cada dimensão de performance [Perlman 2003].

Em resumo, uma rúbrica informa tanto o professor quanto o aluno sobre os critérios de uma avaliação, o que facilita o *feedback* do professor e a auto-avaliação dos alunos [Jonsson and Svingby 2007].

Um exemplo de composição automática de rúbricas é o método de [Olmos et al. 2016] baseado na técnica de redução de dimensionalidade *Latent Semantic Analysis (LSA)* que consiste em avaliar documentos textuais e compor rúbricas a partir de conceitos com a finalidade de identificar e avaliar eixos conceituais de um texto.

Uma outra abordagem de composição de rúbricas aplica a ideia de *scoring rubric* [Perlman 2003] ao informar critérios de avaliação a partir de exemplos de escores atribuídos a soluções de exercícios. Um exemplo mais recente de utilização dessa abordagem é o método de seleção de modelos de soluções de exercícios de programação proposto por [Oliveira et al. 2016]. Nesse método, é utilizado o algoritmo de *clustering Bissecting K-means* com indicação estratégica do número de *clusters* para descobrir modelos de soluções representando gabaritos.

O método de [Oliveira et al. 2016] também dá um passo inicial na tentativa de selecionar a diversidade de soluções para composição de rúbricas. Caminhando nessa direção, nossa estratégia avança mais um passo aplicando a ideia de combinar técnicas de *clustering* e de redução de dimensionalidade proposta por [Oliveira et al. 2015a] para predizer desempenhos em uma nova função, que é a de melhor selecionar a diversidade de soluções de exercícios de programação.

3. Reconhecimento Automático de Representações de Rúbricas

A estratégia tecnológica proposta neste trabalho combina as estratégias de *clustering* e *PCA* para selecionar representações de soluções de gabarito e da diversidade de escores de soluções de exercícios de programação.

O processamento dessa estratégia se inicia com o recebimento de uma Matriz *S* gerada pelo software *PCodigo*, que é um sistema de apoio à prática assistida de programação por execução em massa e análise de exercícios de programação [Oliveira et al. 2015b]. A Matriz *S* reúne as soluções de exercícios de programação submetidas via *Moodle* e representadas em vetores de 60 dimensões, onde cada dimensão é representada pela frequência de ocorrência de palavras reservadas, símbolos, operadores, funções e pelos valores lógicos (0=Falso; 1=Verdadeiro) dos indicadores de funcionamento (compilação e execução) da Linguagem C [Oliveira et al. 2015b].

A Figura 1 apresenta a arquitetura da estratégia de reconhecimento de representações de rúbricas bem como a sua integração ao sistema *PCodigo*.

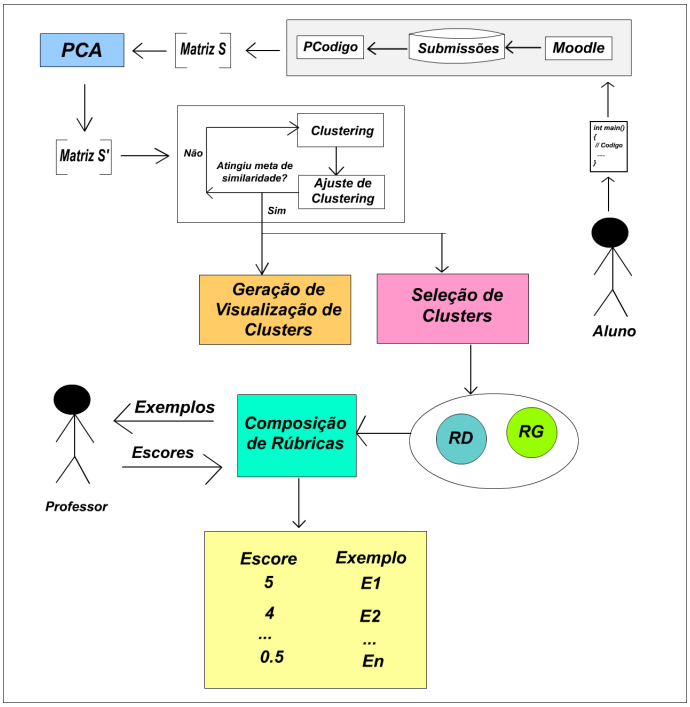


Figura 1. Estratégia de reconhecimento de representações de rúbricas

De acordo com a Figura 1, a Matriz *S* tem sua dimensionalidade reduzida através do método *PCA*, que dá origem à Matriz S', cujas dimensões são mapeadas em fatores que representam as relações lineares entre as variáveis das dimensões da Matriz *S*.

De acordo com [Oliveira et al. 2016], na Figura 1, os processos de *Clustering* e de *Ajuste de Clustering* são realizados repetidamente até que se formem *clusters* com os maiores índices de similaridade. Em seguida, é iniciado o processo de *Geração de Visualização de Clusters* e são selecionados os *clusters* mais adequados para a *Representação de Gabaritos* (*RG*) e para a *Representação da Diversidade* (*RD*) de soluções. Através desses *clusters*, inicia-se o processo de *Composição de Rúbricas*.

Durante o processo de *Composição de Rúbricas* da Figura 1, são selecionados exemplos do *cluster RG* e do *cluster RD* e esses exemplos são apresentados ao professor para que ele atribua-lhes escores. Essa interação ocorre até que se obtenha um conjunto bem representativo de altos, médios e baixos escores a partir de soluções desenvolvidas por alunos de programação com menor esforço de avaliação do professor.

3.1. O Algoritmo Bissecting K-means

Para realizar os processos de *clustering*, utilizamos o algoritmo *Bissecting K-Means* [Karypis 2003] e a visualização gráfica 3D *Mountain Visualization* (*Visão Montanha*), ambos do software *gCluto* [1]. O algoritmo *Bissecting K-means* foi utilizado para formar os agrupamentos dos vetores da Matriz S' que representam soluções de programação.

Os principais parâmetros de *clustering* configurados para formar esses agrupamentos foram a medida de similaridade *cosseno* e o número de *clusters* igual a dez. Es-

[1] Software e manual disponíveis em: http://glaros.dtc.umn.edu/gkhome/cluto/gcluto/overview

colhemos um número de *clusters* maior porque assim formamos *clusters* com maiores índices de similaridade interna [Oliveira et al. 2016].

Após os processos de *clustering*, foram gerados alguns arquivos de saída, dentre eles um relatório que apresenta os *clusters* com os índices de similaridade e desvio interno. As Figuras 3 e 6 são exemplos desses relatórios.

O gráfico 3-D gerado pelo *gCluto* traz uma leitura dos índices dos relatórios de *clustering* em um gráfico do tipo *Mountain Visualization*. Nesse gráfico, a forma de cada relevo é uma curva gaussiana que é utilizada como uma estimativa aproximada da distribuição de dados dentro de cada *cluster*. Dessa forma, a altura é proporcional ao índice de similaridade interna do *cluster*. O volume é proporcional à quantidade de elementos contidos dentro do *cluster*. A cor de cada pico vai de acordo com o desvio interno do *cluster*: vermelho indica alto desvio interno, enquanto azul indica baixo desvio. As Figuras 2 e 5 são exemplos de gráficos *Mountain Visualization* gerados pelo *gCluto*.

3.2. A Análise de Componentes Principais

A fatoração de uma matriz por meio de estatística multivariada tem o intuito de reduzir a dimensionalidade e facilitar a análise e representação dos dados obtidos. A motivação para isso foi devido à possibilidade de se sair de uma matriz de 60 colunas para matrizes de 8, 10, 12 ou 15 colunas, de forma que as informações relevantes dos dados fossem preservadas em novas variáveis formadas a partir de uma combinação linear das variáveis antigas. Visto isso, o fato de lidar com matrizes de menor dimensão sem perda de informação mostrou que o método *PCA* é um poderoso instrumento de extração de dados.

O *PCA* é um método de estatística multivariada que, fazendo uso da decomposicao *SVD*, gera novas variaveis (no caso indicadas pelas colunas da matriz) a partir de combinações lineares das variáveis antigas [Wall et al. 2003].

Ao aplicar o método *PCA*, é obtido como saída do processo uma matriz de covariancia, uma matriz de componentes principais e uma matriz de componentes latentes. Observa-se que a matriz de componentes principais possui as novas variaveis organizadas em ordem decrescente das respectivas componentes latentes a elas associadas, de tal forma que, a primeira componente principal possui a maior componente latente assim como a última componente principal possui a menor componente latente. Dessa forma, a matriz de componentes latentes é organizada em ordem decrescente de latência.

Sabe-se que a componente latente é diretamente proporcional à variancia acumulada. Podemos então concluir que a componente principal com a maior componente latente é a variavel com a maior quantidade de informação. Explorando esse fato, obtemos um criterio subjetivo para a seleção do melhor número de variáveis para a nova matriz S'.

Depois dos experimentos, concluiu-se que algumas componentes principais tinham componentes latentes de magnitude insignificante comparadas à variável de maior latência. Adotou-se então para o método que componentes principais com componente latente de valor absoluto menor do que um não seriam utilizadas. Dessa forma, a matriz de componentes principais gerada apresenta-se com um número de variáveis menor.

4. Experimentos e resultados

Para a experimentação da estratégia de reconhecimento de representações de rúbricas, utilizamos duas bases de soluções de um exercício de programação obtidas em turmas reais de programação de uma universidade e também utilizadas nos experimentos de [Oliveira et al. 2015a] e [Oliveira et al. 2016].

A primeira base, que chamaremos de *Base-A*, reúne 100 amostras de soluções de um exercício coletadas em diferentes turmas de programação. Já a segunda base, que chamaremos de *Base-B*, é uma base formada por 39 amostras de soluções do mesmo exercício utilizado na *Base-A*. No entanto, essa base é considerada mais difícil porque foi obtida em condições controladas de aplicação de prova. Desse modo, essa base apresenta uma maior diversidade de soluções e menor possibilidade de conter amostras plagiadas.

A *Base-A* e a *Base-B* foram mapeadas em vetores de 60 dimensões ou componentes de habilidades, onde cada dimensão, conforme já informamos, é quantificada pela frequência de ocorrência de palavras-chave, funções e indicadores de funcionamento (*compila, executa*) da Linguagem C [Oliveira et al. 2015b, Oliveira et al. 2015a].

Os escores da *Base-A* e da *Base-B* atribuídos por professores variam de 0 a 5, categorizados nas classes de A a E da seguinte forma: a Classe A reúne as amostras de maiores escores (4.1 a 5.0); a Classe B, as amostras de escores médios-altos (3.6 a 4.0); a Classe C, as amostras de escores médios (3.1 a 3.5); a Classe D, as amostras de escores médios-baixos (2.6 a 3.0); e a Classe E, as amostras com escores baixos (0 a 2.5).

Após a aplicação da técnica *PCA*, a *Base-A* teve sua dimensionalidade reduzida a doze fatores (ou componentes latentes) e a *Base-B*, a cinco fatores.

Para realizar o processo de *clustering*, escolhemos a medida de similaridade *cosseno* e o algoritmo *Bissecting K-means*. Após os *Ajustes de Clustering*, fixamos o número de *clusters* em 10 com o propósito de formar *clusters* mais distintos entre si e com padrões internos mais semelhantes entre si. No entanto, podemos ter uma melhor seleção desse número de *clusters*. Uma sugestão para avançar nessa escolha, em trabalhos futuros, é utilizar o algoritmo de *Clustering em Grafo*, conforme o fez [Oliveira et al. 2015a].

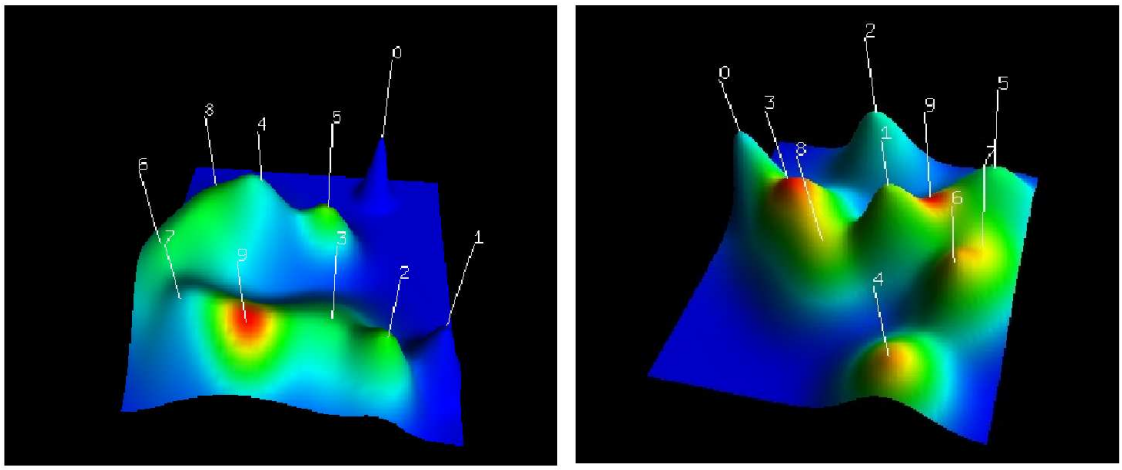
4.1. Resultados

Os resultados alcançados demonstram que é possível selecionar exemplos representativos de escores altos, médios e baixos a partir de um conjunto de diversas soluções de programação desenvolvidas por alunos demandando pouco esforço de avaliação do professor.

A Figura 2 apresenta visualizações 3D dos *clusters* formados a partir da *Base-A* sem aplicação do PCA (A) e com aplicação do PCA (B), sugerindo maior diferenciação entre os *clusters* no Gráfico (B), o que permite uma melhor separação das soluções de exercícios por classes de escores.

Nos *clusters* 0 e 3 do Gráfico (B) da Figura 2, observamos uma maior proximidade entre ambos, sendo o primeiro mais homogêneo e o segundo mais heterogêneo, conforme pode ser confirmado na Figura 3 pelos valores de similaridade interna (ISIM), que é maior no *Cluster 0*, e desvio interno (ISDev), que é maior no *Cluster 3*.

Analisando os *clusters* 0 e 3 da *Base-A* (Figura 2 - (B)), observamos, conforme Figura 4, que esses *clusters* se aproximam mais pelas notas mais altas (Classe A) e se



(A) *Clusters da Base-A – Sem PCA* **(B) *Clusters da Base-A – Com PCA***

Figura 2. Os clusters da Base-A

10-way clustering: [100 of 100]

Cluster	Size	ISim	ISdev
0	5	0.872	0.023
1	7	0.809	0.098
2	10	0.739	0.062
3	15	0.717	0.173
4	13	0.665	0.166
5	13	0.633	0.077
6	7	0.669	0.112
7	13	0.465	0.124
8	10	0.450	0.120
9	7	0.415	0.153

Go to Top

Figura 3. Resultados de Clustering da Base-A

diferenciam mais pelas notas baixas (Classe E). Esses clusters podem conter, portanto, a melhor representação das notas altas.

A Figura 4 apresenta como as amostras de soluções de exercícios da *Base-A* foram organizadas em alguns *clusters*. Nessas tabelas, *Cluster* informa o *id* dos *clusters*, *Índice* é um rótulo de amostra, *Escore* é a nota atribuída por um professor a uma amostra de solução de exercício, *Classe* indica em que faixa de escores está esse escore e cada uma dessas classes é representada por uma cor.

De acordo com as Figuras 2 e 4, podemos observar também que outra representação da diversidade de soluções pode ser encontrada em um número menor de amostras no *Clusters 9*, que possui o mais baixo valor de *ISIM* e alto valor de *ISDev* indicando alta heterogeneidade, conforme sugere a cor vermelha nesse *cluster* (Figura 2 - (B)).

A Figura 5 apresenta os *clusters* da *Base-B*. Assim como aconteceu na *Base-A*, os *clusters* da *Base-B* melhor se diferenciaram com aplicação da técnica *PCA*. No entanto, o *cluster* representante da diversidade de soluções, que é o *Cluster 5* (com pico vermelho) (Figura 5 - (B)), aparece mais isolado e em maior evidência indicando a sua alta heterogeneidade e a homogeneidade dos demais *clusters*.

De acordo com a Figura 6, confirma-se essa diferenciação do *Cluster 5* pelo seu alto valor de *ISDev* e pelos altos valores de *ISIM* dos demais *clusters* da *Base-B* (com

Cluster	Indice	Escore	Classe
0	al49	4.0	B
0	al50	4.5	A
0	al46	5.0	A
0	al83	4.3	A
0	al22	5.0	A

Cluster	Indice	Escore	Classe
1	al85	2.7	D
1	al98	1.2	E
1	al58	0.75	E
1	al13	3.75	B
1	al87	3.2	C
1	al9	2.75	D
1	al54	2.0	E

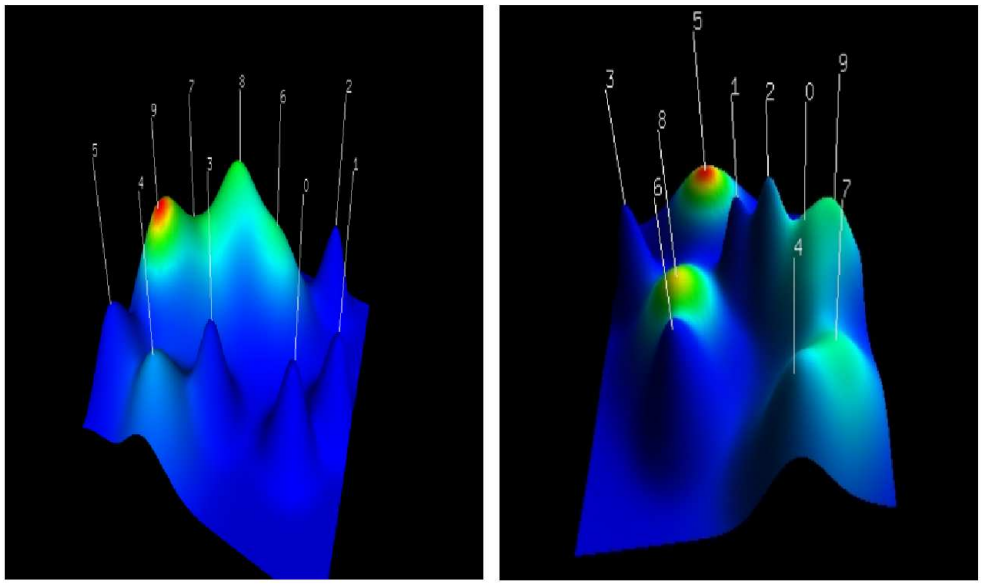
Cluster	Indice	Escore	Classe
2	al17	3.5	C
2	al36	3.0	D
2	al57	3.75	B
2	al88	3.2	C
2	al34	4.75	A
2	al23	3.0	D
2	al11	3.0	D
2	al42	3.25	C
2	al35	1.75	E
2	al45	3.5	C

Cluster	Indice	Escore	Classe
3	al95	3.7	B
3	al2	4.0	B
3	al70	2.5	E
3	al26	5.0	A
3	al96	5.0	A
3	al69	4.75	A
3	al93	1.5	E
3	al38	5.0	A
3	al74	5.0	A
3	al92	5.0	A
3	al43	5.0	A
3	al25	3.25	C
3	al39	4.5	A
3	al99	2.0	E
3	al62	2.5	E

Cluster	Indice	Escore	Classe
4	al8	4.0	B
4	al73	3.6	B
4	al12	3.75	B
4	al84	3.0	D
4	al53	4.75	A
4	al67	3.75	B
4	al91	3.2	C
4	al7	2.0	E
4	al63	1.5	E
4	al30	3.5	C
4	al3	3.25	C
4	al100	4.0	B
4	al27	4.0	B

Cluster	Indice	Escore	Classe
9	al76	3.5	C
9	al32	4.75	A
9	al40	3.5	C
9	al56	1.5	E
9	al78	0.5	E
9	al71	4.2	A
9	al6	3.0	D

Figura 4. Análise de clusters da Base-A



(A) Clusters da Base-B – Sem PCA ***(B) Clusters da Base-B – Com PCA***

Figura 5. Os clusters da Base-B

exceção do *Cluster 9*) em relação ao *Cluster 5*.

A Figura 7 apresenta a organização dos *clusters* da *Base-B* conforme os padrões de organização da *Base-A* (Figura 4). Analisando o *Cluster* 5 da *Base-B* na Figura 7, observamos a melhor representação da diversidade das soluções.

Uma outra observação nos *clusters* da Figura 5 - (B) é a proximidade entre os *clusters* 0 e 9, sendo que ambos, segundo os processos de *clustering*, representam, respectivamente, o *cluster* mais homogêneo e o *cluster* mais heterogêneo. No entanto, ambos

10-way clustering: [39 of 39]

Cluster	Size	ISim	ISdev
0	5	0.947	0.021
1	1	1.000	0.000
2	1	1.000	0.000
3	1	1.000	0.000
4	4	0.978	0.007
5	8	0.795	0.072
6	2	0.855	0.000
7	7	0.869	0.021
8	7	0.835	0.055
9	3	0.775	0.022

Go to Top

Figura 6. Resultados de Clustering da Base-B

Cluster	Indice	Escore	Classe
0	r27	2.0	E
0	r37	0.5	E
0	r19	2.0	E
0	r13	2.5	E
0	r30	2.75	D

Cluster	Indice	Escore	Classe
4	r8	0.0	E
4	r34	0.0	E
4	r40	0.0	E
4	r22	0.5	E

Cluster	Indice	Escore	Classe
5	r31	0.25	E
5	r4	3.0	D
5	r9	4.75	A
5	r26	2.75	D
5	r12	2.0	E
5	r36	5.0	A
5	r1	4.0	B
5	r15	5.0	A

Cluster	Indice	Escore	Classe
6	r32	0.0	E
6	r20	0.0	E

Cluster	Indice	Escore	Classe
9	r5	2.75	D
9	r24	0.0	E
9	r10	2.75	D

Figura 7. Análise de clusters da Base-B

se aproximam pelas soluções de escores médios.

Os *clusters* mais homogêneos da *Base-B*, como pode ser visualizado nas Figuras 5 e 7, assemelham-se pelas soluções de escores mais baixos.

Os resultados de análise dos *clusters* da *Base-A* e da *Base-B* indicam, portanto, que podemos reconhecer nos *clusters* mais homogêneos a melhor representação das soluções com altos ou baixos escores. Já, nos *clusters* mais heterogêneos com alto desvio interno (*ISdev*), podemos obter a melhor representação dos critérios de avaliação de um professor através da diversidade de exemplos de soluções com escores baixos, médios e altos.

5. Considerações Finais

Este trabalho apresentou uma estratégia tecnológica de reconhecimento automático de representações de rúbricas a partir de agrupamentos de soluções de exercícios de programação. Os resultados alcançados indicam que a estratégia proposta reconhece automaticamente exemplos de escores altos, médios e baixos para composiçao de rúbricas.

O método de Análise de Componentes Principais melhorou o processo de reconhecimento de soluções representativas dos critérios de avaliação dos professores a partir dos agrupamentos de soluções formados em um processo de *clustering*. No entanto, uma limitação dessa estratégia proposta é escolher o melhor número de *clusters* que forme os melhores agrupamentos para capturar exemplos da diversidade de soluções.

Como trabalhos futuros a partir deste, propomos reconhecer, além das amostras, as características e suas relações que melhor expliquem os exemplos de rúbricas. Além

disso, propomos uma nova representação vetorial de perfis de alunos formada por outras métricas para caracterizar habilidades e dificuldades de programação.

Concluindo, as principais contribuições da estratégia proposta para a aprendizagem de programação são assistir o trabalho do professor no processo avaliativo e possibilitar *feedbacks* mais claros e mais objetivos para estudantes de programação.

Referências

Jonsson, A. and Svingby, G. (2007). The use of scoring rubrics: Reliability, validity and educational consequences. *Educational research review*, 2(2):130–144.

Karypis, G. (2003). CLUTO - A Clustering Toolkit. Dept. of Computer Science, University of Minnesota.

Kwon, H. and Jo, M. (2005). Design and implementation of the automatic rubric generation system for the neis based performance assessment using data mining technology. *Journal Of the Korean Association of information Education*, 9(1):113–126.

Lindenbaum, M., Markovitch, S., and Rusakov, D. (2004). Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54(2):125–152.

Oliveira, E., Basoni, H., Saúde, M. R., and Ciarelli, P. (2014). 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 (IC3K 2014)*, pages 465–472.

Oliveira, M., Leal, L., and Oliveira, E. (2016). Sistema de Apoio à Avaliação de Atividades de Programação por Reconhecimento Automático de Modelos de Soluções. In *XXIV Workshop sobre Educação em Computação (WEI) - CSBC 2016*, Porto Alegre, RS. SBC.

Oliveira, M., Monroy, N., Daher, P., and Oliveira, E. (2015a). Representação da diversidade de componentes latentes em exercícios de programação para classificação de perfis. In *IV Congresso Brasileiro de Informática na Educação (CBIE 2015)*, Maceió. Anais do SBIE 2015.

Oliveira, M., Nogueira, M. A., and Oliveira, E. (2015b). Sistema de Apoio à Prática Assistida de Programação por Execução em Massa e Análise de Programas. In *XXIII Workshop sobre Educação em Computação (WEI) - CSBC 2015*, Recife, PE. SBC.

Olmos, R., Jorge-Botana, G., Luzón, J. M., Martín-Cordero, J., and León, J. (2016). Transforming LSA space dimensions into a rubric for an automatic assessment and feedback system . *Information Processing & Management*, 52(3):359–373.

Perlman, C. (2003). Performance assessment: Designing appropriate performance tasks and scoring rubrics.

Tuia, D., Pasolli, E., and Emery, W. (2011). Using active learning to adapt remote sensing image classifiers. *Remote Sensing of Environment*, 115(9):2232–2242.

Wall, M. E., Rechtsteiner, A., and Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer.