Revista Brasileira de Informática na Educação – RBIE
Brazilian Journal of Computers in Education
(ISSN online: 2317-6121; print: 1414-5685)
http://br-ie.org/pub/index.php/rbie

Submission: 15/Feb/2017; 1st round notif.: 22/Apr/2017; New version: 22/Jul/2017; 2nd round notif.: 10/Sep/2017
Camera ready: 10/Dec/2017; Edition review: 21/Mar/2018; Available online: 28/Mar/2018; Published: 07/May/2018

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

Title: Automatic Recognition of Rubric Representations in Programming Exercises Clusters

Márcia Gonçalves de Oliveira
Centro de Referência em Formação e EaD
Instituto Federal do Espírito Santo
clickmarcia@gmail.com

Leonardo Leal Reblin
Departamento de Engenharia Elétrica
Universidade Federal do Espírito Santo
leoreblin@gmail.com

Mateus Batista de Souza
Departamento de Engenharia Elétrica
Universidade Federal do Espírito Santo
mateusbsouza@hotmail.com

Elias Silva de Oliveira
Programa de Pós-Graduação em Informática
Universidade Federal do Espírito Santo
elias_oliveira@acm.org

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 para reconhecer, a partir de soluções desenvolvidas por alunos, exemplos de soluções que representem, em um esquema de rubricas, os escores atribuídos por um professor. Os resultados dos experimentos em soluções reais de exercícios de programação indicam que o nosso método reconhece representações de rubricas demandando pouco esforço de avaliação de professores.
Palavras-chave: *Clustering; Análise de Componentes Principais; Rubricas; Programação*

Abstract

The evaluation of programming exercises is a complex process because, for each exercise that a teacher applies, there are common possibilities for solutions. As the teacher does not always know all the possible solutions of an exercise, it is always a challenge for him to justify all the criteria of his evaluation. In order to support the programming evaluation process, this work 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 rubric scheme, the scores assigned by a teacher. The results of the experiments in real programming exercises solutions indicate that our method recognizes representations of rubrics demanding little teacher evaluation effort.
Keywords: *Clustering; PCA; Rubrics; Programming*

Cite as: Oliveira, M.G., Reblin, L.L., Souza, M.B. & Oliveira, E. (2017). Automatic Recognition of Rubric Representations in Programming Exercises Clusters (Reconhecimento Automático de Representações de Rubricas em Agrupamentos de Soluções de Exercícios de Programação). Brazilian Journal of Computers in Education (Revista Brasileira de Informática na Educação – RBIE), 26(2), 60-79. DOI: 10.5753/RBIE.2018.26.02.60.

1 Introdução

A avaliação de exercícios de programação é um processo complexo que demanda muito esforço manual e cognitivo de professores na análise de uma grande quantidade de soluções de programação, em especial quando cada exercício tem várias possibilidades de resolução. Dessa forma, como nem sempre é possível o professor conhecer todas as possíveis soluções, ele quase sempre precisa analisar minuciosamente cada solução, compreender como o aluno a desenvolveu e atribuir à solução um escore informando sua corretude.

Uma vez que a avaliação de exercícios de programação baseia-se, de modo geral, em decisões subjetivas, ela se torna um desafio para o professor, especialmente quando ele precisa explicar seus critérios e responder a muitas reclamações dos alunos na entrega dos resultados das avaliações.

Para melhor justificar seus critérios de avaliação, o professor poderia selecionar exemplos representativos de cada classe de escore a partir de soluções já avaliadas. Mas essa tarefa torna-se onerosa quando há muitas soluções de vários alunos a serem avaliadas e essas soluções são complexas. Há, portanto, a necessidade de um tratamento automático desse problema.

Com o objetivo de auxiliar professores na avaliação de exercícios de programação através da composição de modelos de soluções que informem critérios de avaliação e justifiquem diferentes escores, propomos neste trabalho uma estratégia de reconhecimento automático de representações de rubricas a partir de soluções de programação desenvolvidas por estudantes.

Nessa estratégia, o reconhecimento de exemplos que representam gabaritos e a diversidade de soluções para um exercício de programação é realizado através da combinação das técnicas de *Análise de Componentes Principais (Principal Component Analysis - PCA)* e de *clustering*. Através de um algoritmo de *PCA*, reduzimos a dimensionalidade da matriz que reúne os vetores de cada solução, tornando possível uma melhor seleção da representação da diversidade de soluções.

Após os processos de *clustering*, identificamos, dentro dos *clusters* mais homogêneos, as soluções muito similares que podem receber um mesmo escore e ser reconhecidas como gabaritos por um professor após análise de alguns exemplos de um *cluster*. Já nos *clusters* mais cheios e mais heterogêneos, identificamos uma representação da diversidade de soluções de programação.

Para capturar essa diversidade de forma interativa, o professor participa de um processo de aprendizagem ativa atribuindo escores às soluções selecionadas pelo sistema de avaliação da estratégia tecnológica proposta até que tenham sido avaliados, em um esquema de rubricas, exemplos de soluções que melhor representem a diversidade de soluções dos alunos e os critérios de avaliação desse professor.

Para os estudantes de programação, a principal vantagem dessa estratégia é oferecer-lhes maior clareza do processo avaliativo fornecendo-lhes modelos de soluções representativas dos diferentes escores atribuídos por seus professores como também as diferentes possibilidades de soluções de um mesmo exercício.

Uma outra vantagem da estratégia proposta é a possibilidade de utilizar os modelos de soluções de programação reconhecidos em sistemas de avaliação automática como referências de gabaritos ou em sistemas de avaliação semiautomática, como amostras representativas de um conjunto de treino utilizado para aprendizagem de máquina.

Em sistemas de avaliação semiautomática, os modelos de soluções seriam as referências para descobrir e reunir automaticamente mais amostras semelhantes em uma base de soluções para formar um conjunto de treino mais reduzido e mais representativo. Dessa forma, esses sistemas aprenderiam por meio de alguns exemplos representativos como um professor dá uma nota e reproduziriam seus critérios em outros exemplos similares.

Em resumo, a contribuição deste trabalho que o diferencia de outros sistemas que geram representações de rubricas é que a estratégia proposta combina técnicas de *PCA* e de *clustering* para reconhecer modelos de soluções que melhor representem gabaritos e a variabilidade de soluções desenvolvidas para um exercício de programação. As representações reconhecidas poderão também contribuir para a aprendizagem de máquina, formando conjuntos de exemplos de treino mais representativos.

Com a finalidade de apresentar os estudos realizados e os resultados alcançados, este trabalho está organizado conforme a ordem a seguir. Na Seção 2, apresentamos uma revisão de literatura e destacamos os trabalhos relacionados que oferecem mecanismos de avaliação baseados em rubricas e estratégias voltadas para a programação. Na Seção 3, descrevemos a estratégia proposta de reconhecimento de representações de rubricas bem como as técnicas e algoritmos utilizados. Na Seção 4, apresentamos os experimentos realizados, os resultados alcançados e as extensões da estratégia proposta. Na Seção 5, concluimos com as considerações finais e trabalhos futuros.

2 Revisão de Literatura

Uma rubrica é uma ferramenta de pontuação para avaliação qualitativa de várias dimensões de desempenhos que informa tanto o professor quanto o aluno sobre os critérios de uma avaliação, o que facilita o *feedback* do professor e a autoavaliação dos alunos (Panadero & Jonsson, 2007).

Atualmente, as rubricas têm sido uma tendência para agregar mais confiabilidade a sistemas de avaliação automática (Kwon & Jo, 2005). Um exemplo disso é a abordagem de composição de rubricas de sistemas de avaliação automática que aplicam a ideia de *Scoring Rubric*.

O *Scoring Rubric* consiste em informar critérios de avaliação de professores a partir de exemplos de escores atribuídos a soluções de exercícios (Perlman, 2003). De acordo com Perlman (2003), 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.

A abordagem de *Scoring Rubric*, no entanto, tem sido mais utilizada em sistemas de avaliação automática para formar referências de correção de professores para sistemas de aprendizagem gerando regras de atribuição de escores a serem aprendidas e reproduzidas por tais sistemas (Srikant & Aggarwal, 2014; Yamamoto, Umemura, & Kawano, 2018). Há, dessa forma, uma necessidade de se desenvolver sistemas que reconheçam representações de rubricas a partir de um conjunto de respostas de estudantes a atividades propostas para serem referências de escores tanto para sistemas de avaliação automática quanto para professores e alunos.

Em alguns 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, Markovitch, &

Rusakov, 2004), de aprendizagem ativa (Tuia, Pasolli, & Emery, 2011; Oliveira, Basoni, Saúde, & Ciarelli, 2014) e de composição de rubricas (Kwon & Jo, 2005).

No trabalho de Lindenbaum et al. (2004), a abordagem de amostragem seletiva é utilizada na seleção de amostras de um conjunto de exemplos não-rotulados para treinar um classificador automático. As amostras selecionadas são então rotuladas pelo especialista humano e utilizadas como treino do classificador, o que caracteriza um modelo de aprendizagem ativa.

Na estratégia de aprendizagem ativa de Oliveira et al. (2014), o sistema seleciona um número mínimo de exemplos de textos do *Twitter* para classificação por um especialista humano e, em seguida, classifica automaticamente um grande número de outros *tweets* baseando-se nos exemplos rotulados pelo especialista humano.

Na proposta de composição de rubricas de Kwon and Jo (2005), foi desenvolvida uma ferramenta para auxiliar professores a selecionarem e a desenvolverem critérios de avaliação de desempenhos considerando características individuais dos alunos e preferências dos professores. Para essa análise, foram utilizadas regras de classificação e associação através da mineração de dados, o que permitiu reduzir o tempo e os esforços para o desenvolvimento e seleção de rubricas.

A estratégia de reconhecimento automático de representações de rubricas a partir de soluções de programação não-rotuladas desenvolvidas por alunos é uma proposta nova em relação às propostas de *Scoring Rubric* apresentadas na literatura científica. No entanto, é uma estratégia que utiliza métodos como a amostragem seletiva e a aprendizagem ativa que já são aplicados em outros contextos. Além disso, nossa proposta combina tecnologias e metodologias utilizadas em diferentes domínios de conhecimento, conforme os trabalhos relacionados destacados a seguir.

2.1 Trabalhos relacionados

Os trabalhos mais relacionados à proposta deste trabalho de reconhecimento automático de rubricas a partir de programas desenvolvidos por estudantes são o método de seleção de modelos de soluções de Oliveira, Reblin, and Oliveira (2016), o modelo de seleção de características de Spalenza, Oliveira, Oliveira, and Nogueira (2016), a composição automática de rubricas baseada na técnica de redução de dimensionalidade *Latent Semantic Analysis (LSA)* de Olmos, Guillermo, Luzón, Martín-Cordero, and Leao (2016) e o método de reconhecimento automático de modelos de soluções de exercícios de programação de Oliveira, Souza, Reblin, and Oliveira (2016).

O método de seleção de modelos de soluções de exercícios de programação proposto por Oliveira, Reblin, and Oliveira (2016) utiliza o algoritmo de *clustering Bissecting K-means* com indicação estratégica do número de *clusters* para descobrir modelos de soluções representando gabaritos. Nesse trabalho mostra-se que, aumentando o número de *clusters*, formam-se *clusters* mais homogêneos. No entanto, o número de características que representam as dimensões de perfis de estudantes de programação é alto, o que interfere na qualidade de formação dos *clusters*.

Em nossos experimentos, conforme Oliveira, Reblin, and Oliveira (2016), também utilizamos o *Bissecting K-means* e realizamos ajustes de *clustering* aumentando o número de *clusters*. No entanto, utilizamos algoritmos de *PCA* para redução de dimensionalidade visando promover uma maior diferenciação dos *clusters* de forma a obter agrupamentos com as melhores representações da diversidade de soluções e das melhores soluções de exercícios.

Um exemplo de composição automática de rubricas utilizando redução de dimensionalidade é o trabalho de Olmos et al. (2016) que utiliza a técnica *Latent Semantic Analysis (LSA)* na avaliação de documentos textuais para compor rubricas a partir de conceitos visando reconhecer e avaliar eixos conceituais de um texto.

O trabalho de Oliveira, Monroy, Daher, and Oliveira (2015) também utiliza a técnica de redução de dimensionalidade para melhor representar as soluções de programação, mas com a finalidade de classificar perfis de estudantes por faixas de desempenhos em exercícios.

Em nossa proposta, utilizamos a estratégia de redução de dimensionalidade de Oliveira, Monroy, et al. (2015) com a finalidade de melhorar a seleção de modelos de soluções de programação por *clustering* proposta por Oliveira, Reblin, and Oliveira (2016). Mas, além de selecionar amostras para as representações de rubricas, também selecionamos características para descrever essas representações, baseando-se na ideia de mapa de características de Spalenza et al. (2016).

O modelo de seleção de características de Spalenza et al. (2016), combinando as técnicas de *clustering* e de algoritmo genético, cria um mapa de características selecionando termos relevantes nos textos dentre os grupos de notas da avaliação de um professor. De acordo com Spalenza et al. (2016), no *feedback* enviado aos alunos, os termos relevantes apareceriam em destaque, o que facilitaria a explicação das notas atribuídas por um professor. Em nossa proposta, mais do que apresentar os termos descritivos e discriminativos dos *clusters* representativos de gabaritos e da diversidade de soluções de programação, através dos fatores formados pela técnica de *PCA*, reconhecemos as características que eles representam bem como as relações entre elas.

O método de Oliveira, Souza, et al. (2016) dá um passo inicial na tentativa de selecionar a diversidade de soluções para composição de rubricas. Caminhando nessa direção, nossa estratégia aplica esse método utilizando a redução de dimensionalidade de Oliveira, Monroy, et al. (2015) e a seleção de características de Spalenza et al. (2016) com a finalidade de selecionar representações de gabaritos e a diversidade de soluções para composição de rubricas no esquema de *Scoring Rubric*. Neste trabalho, avançamos mais um passo utilizando algoritmos de *Graph Clustering* para seleção de amostras representativas de gabaritos e da diversidade de soluções de programação, o que reduz o problema da dependência do número de *clusters* para formar *clusters* mais homogêneos na proposta de Oliveira, Reblin, and Oliveira (2016).

3 Reconhecimento automático de representações de rubricas

A estratégia tecnológica deste trabalho combina as estratégias de *clustering* e *PCA* para selecionar representações de rubricas a partir 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, Nogueira, & Oliveira, 2015). 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 *tokens*, símbolos, operadores, funções e pelos valores lógicos (0=Falso; 1=Verdadeiro) dos indicadores de funcionamento (*compila* e *executa*) da Linguagem C (Oliveira, Nogueira, & Oliveira, 2015).

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

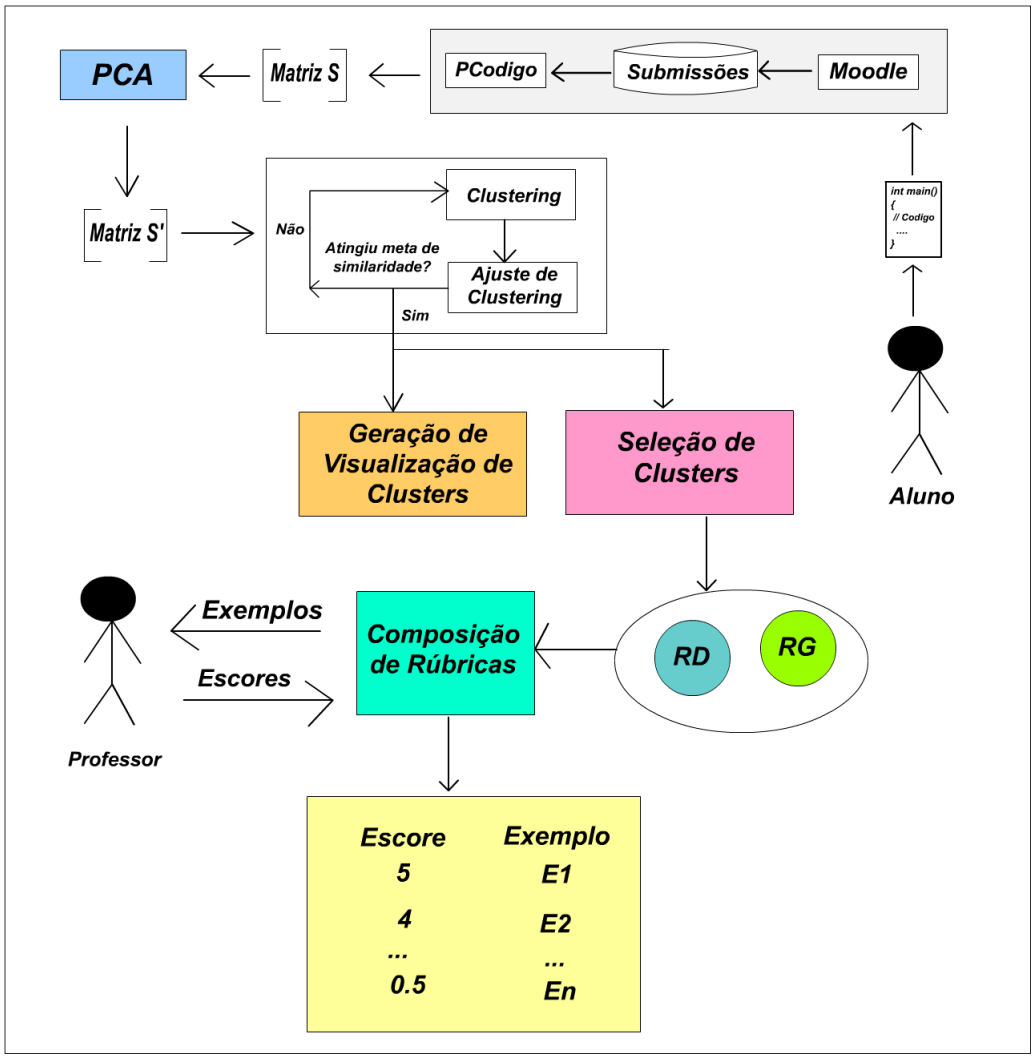


Figura 1: Estratégia de reconhecimento automático de representações de rubricas.

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.

Conforme Oliveira, Reblin, and Oliveira (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 Rubricas*.

Durante o processo de *Composição de Rubricas* da Figura 1, são selecionados exemplos do *cluster RG* e do *cluster RD* e esses exemplos são apresentados ao professor para que ele lhes

atribua escores. Essa interação ocorre até que se obtenha um conjunto mais representativo da diversidade de soluções 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, 2002) e a visualização gráfica *3D Mountain Visualization (Visualização 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. Escolhemos um número de *clusters* maior porque assim formamos *clusters* com maiores índices de similaridade interna (Oliveira, Reblin, & Oliveira, 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 de desvio interno. As Figuras 3 e 7 são exemplos desses relatórios. O gráfico *3D* 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* (Karypis, 2002). 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 6 são exemplos de gráficos *Mountain Visualization* gerados pelo *GCluto*.

3.2 Análise de Componentes Principais

A fatoração de uma matriz por estatística multivariada tem o intuito de reduzir a dimensionalidade e facilitar a análise e a representação dos dados obtidos. A motivação para isso foi devido à possibilidade de se sair de uma matriz de sessenta colunas para matrizes de oito a quinze colunas, de forma que as informações relevantes dos dados fossem preservadas em novas variáveis formadas a partir de uma combinação linear das variáveis antigas.

O *PCA* é um método de estatística multivariada que, fazendo uso da decomposição *SVD (Singular Value Decomposition – Decomposição em Valores Singulares)*, gera novas variáveis a partir de combinações lineares das variáveis antigas (Wall, Rechtsteiner, & Rocha, 2003).

Ao aplicar o método *PCA*, é obtido como saída do processo uma matriz de covariância, uma matriz de componentes principais e uma matriz de componentes latentes. Observa-se que a matriz de componentes principais possui as novas variáveis 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.

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

Sabe-se que a componente latente é diretamente proporcional à variância acumulada. Podemos então concluir que a componente principal com a maior componente latente é a variável com a maior quantidade de informação. Explorando esse fato, obtemos um critério subjetivo para a seleção do melhor número de variáveis para a nova *Matriz S*.

Nos experimentos, concluiu-se que algumas componentes principais possuíam 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 automático de representações de rubricas, 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, Reblin, and Oliveira (2016) e de Oliveira, Souza, et al. (2016). Esse exercício de programação, que chamaremos de E_P, envolve comandos de entrada e de saída, expressões lógicas, estruturas de controle condicional e de repetição, e possui o seguinte enunciado:

> Obtenha o número de pontos P de três times em um campeonato de futebol, de acordo com a expressão matemática a seguir:
>
> $$P = 5G_P - G_N + 3V_F + 2V_C + E$$
>
> Nessa formula, G_P é o número de gols positivos, G_N é o número de gols tomados, V_F é o número de vitorias fora de casa, V_C é o número de vitorias em casa e E é o número de empates. No final, o programa deve mostrar, de acordo com o número de pontos obtidos por um time, o campeão e o vice-campeão do campeonato.

A solução do exercício E_P é considerada difícil porque considera que aluno saiba operar logicamente usando um número mínimo de comparações. Dessa forma, o que caracteriza uma boa solução para E_P, segundo a avaliação do professor que aplicou esse exercício, está principalmente no uso de expressões lógicas e de estruturas de controle condicional. Espera-se, portanto, que na solução de E_P aplique-se o menor número possível de expressões lógicas e estruturas de controle condicionais. No gabarito de E_P, além de uma expressão aritmética e uma estrutura de repetição, aparecem apenas três expressões lógicas de comparação em estruturas condicionais: duas em uma estrutura *if-ladder* (estrutura *if aninhada*) e uma dentro de uma estrutura *if*.

Muitas soluções de E_P presentes na *Base-A* e na *Base-B* estavam corretas, porém, utilizaram muitas expressões lógicas para realizar comparações. Essas soluções foram consideradas como parcialmente corretas. As soluções consideradas incorretas, por sua vez, apresentavam falhas na compreensão das expressões lógicas e no uso das estruturas de controle condicional.

A primeira base, que chamamos de *Base-A*, reúne 100 amostras de soluções do exercício

E_P coletadas em diferentes turmas de programação. Já a segunda base, que chamamos de *Base-B*, é formada por 39 amostras de soluções de E_p. A *Base-B* é considerada mais difícil uma vez que foi obtida em condições controladas de aplicação de prova. Desse modo, essa base apresenta uma maior diversidade de soluções e menos possibilidades de conter amostras de soluções 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 *tokens*, funções e indicadores de funcionamento (*compila, executa*) da Linguagem C (Oliveira, Nogueira, & Oliveira, 2015; Oliveira, Monroy, et al., 2015).

Os escores da *Base-A* e da *Base-B* atribuídos por professores variam de 0 a 5 e são categorizados nas classes de *A* a *E*, conforme a Tabela 1.

Tabela 1: Classes de escores.

Classes	**Faixa de escores**
A	4.1 a 5.0
B	3.6 a 4.0
C	3.1 a 3.5
D	2.6 a 3.0
E	0 a 2.5

Após a aplicação da técnica *PCA*, a *Matriz S* da *Base-A* teve sua dimensionalidade reduzida a doze componentes principais (Tabela 2) e a da *Base-B*, a cinco componentes (Tabela 3).

Tabela 2: Componentes principais da Base-A.

Componentes	**Representação**	**Tokens/Símbolos/Identificadores da Linguagem C**
F1	Expressão aritmética	+, *, -
F2	Estrutura de Seleção	break, case, switch
F3	Laço de repetição	–, for, ++
F4	Expressão lógica	|| (OR), ! (NOT), char
F5	Condição se verdadeira	if
F6	Comparação maior ou igual	>=
F7	Execução (Compila/Executa)	
F8	Comparação menor ou igual	int, <=
F9	Função principal	main
F10	Entrada de dados	scanf
F11	Condição se falsa	else
F12	Comparação de Igualdade	==

Na primeira e na segunda colunas da Tabela 2, indicamos os nomes de *Representação* e os *Tokens/Símbolos/Identificadores da Linguagem C* relacionados linearmente em cada componente principal da *Base-A*. Essas mesmas indicações da Tabela 2 aparecem na Tabela 3 para a *Base-B*.

Após a aplicação do método *PCA* nas matrizes *S* da *Base-A* e da *Base-B*, as matrizes reduzidas S' obtidas foram submetidas aos algoritmos de *clustering*.

Para realizar o processo de *Clustering*, escolhemos a medida de similaridade *cosseno* e o algoritmo *Bissecting K-means*. Após os *Ajustes de Clustering* para identificar o melhor número de *clusters*, fixamos o número de *clusters* em dez com o propósito de formar *clusters* mais distintos entre si e com padrões internos mais semelhantes entre si.

Tabela 3: Componentes principais da Base-B.

Componentes	**Representação**	**Tokens/Símbolos/Identificadores da Linguagem C**
F1	Expressão aritmética	+, *, -
F2	Estrutura de Seleção, Comparação maior	case, break, else, >
F3	Comparação menor, Atribuição	<, =
F4	Entrada de dados	scanf
F5	Comparação >=, Laço de repetição	>=, do

4.1 Resultados dos experimentos na Base-A

Os resultados alcançados demonstram que é possível selecionar exemplos representativos 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*. O *Gráfico (B)*, com aplicação do *PCA*, sugere maior diferenciação entre os *clusters*, o que permite uma melhor separação das soluções de exercícios por classes de soluções.

No *Gráfico (B)* da Figura 2, nas ”montanhas” mais à esquerda, observamos uma maior proximidade entre os *clusters* 0 e 3. O *Cluster* 0 é o mais homogêneo e o *Cluster 3* é o segundo mais heterogêneo. Esses resultados podem ser confirmados na Figura 3 pelos valores de similaridade interna (*ISIM*), que é maior no *Cluster* 0, e de desvio interno (*ISDev*), que é maior no *Cluster* 3.

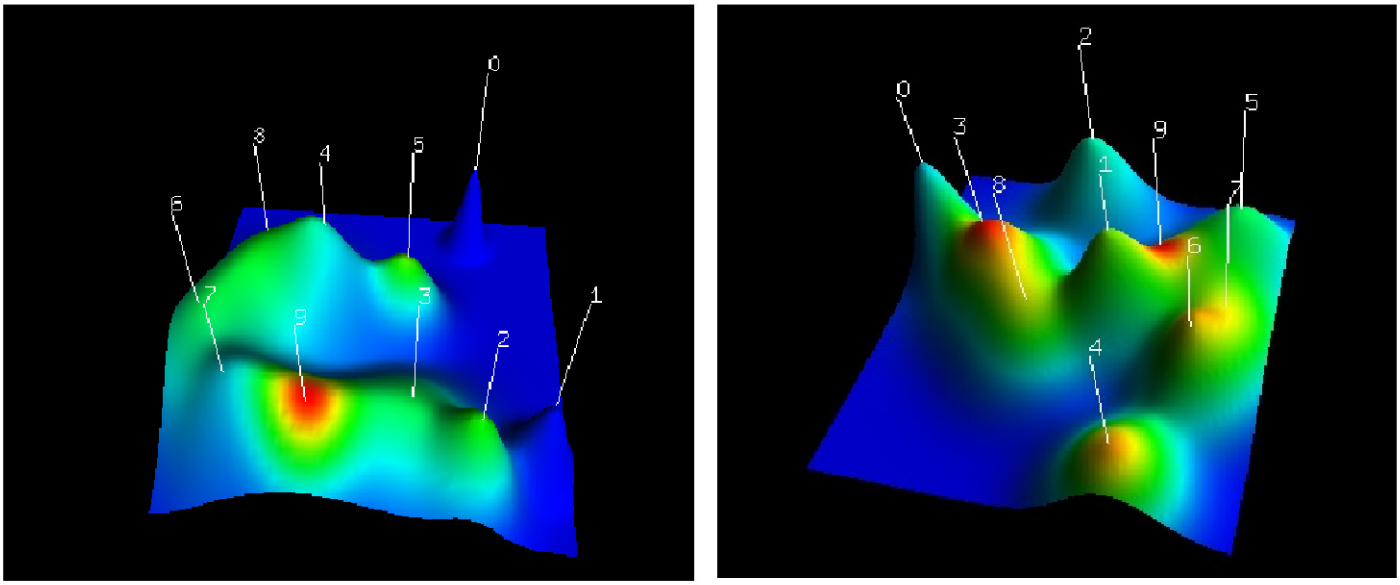
A Figura 4 apresenta como as amostras de soluções de exercícios da *Base-A* foram organizadas em alguns *clusters*. Nessas tabelas, *Cluster* informa o *id* dos *clusters*, *Índice* é um rótulo de amostra, *Escore* é a nota atribuída por um professor a uma amostra de solução de exercício, a *Classe* indica a faixa de escores de cada classe e essa faixa é representada por uma cor (A= Azul; B = Verde; C = Amarelo; D = Laranja; E = Vermelho).

Analisando os *clusters* 0 e 3 da *Base-A* (Figura 2-(B)), observamos que esses *clusters* se aproximam e são formados por soluções com notas mais altas (*Classe A*). Além disso, observa-se que os *clusters* formados com exemplos de notas mais baixas (*Classe E*) se diferenciam mais, considerando que essas amostras contêm mais informações dentro dos códigos-fontes, conforme as tabelas à esquerda na Figura 4. Os *clusters* 0 e 3 podem conter, portanto, a melhor representação de notas altas, o que pode ser confirmado quando o professor pontuar alguns exemplos de amostras desses *clusters*.

De acordo com as Figuras 2 e 4, a melhor representação da diversidade de soluções pode ser encontrada com um número menor de amostras no *Cluster 9* (última tabela à direita, na Figura 4), que aparece com um pico vermelho na Figura 2-(B). O *Cluster 9*, conforme a Figura 3, possui o mais baixo valor de *ISIM* e um alto valor de *ISdev*, o que indica uma alta heterogeneidade das amostras reunidas nesse *cluster*.

A Figura 5 apresenta as duas componentes principais descritivas e discriminantes de cada *cluster* da *Base-A* com *PCA* (Figura 2–(B)).

De acordo com a Figura 5, os *clusters* 0 e 3, onde predominam as soluções de gabarito, são descritos pelas componentes *F5 (Condição se verdadeira)* e *F1 (Expressão Aritmética)*, respectivamente.



(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 com PCA.

Isso significa que as questões corretas, em geral, conterão a expressão aritmética e a estrutura *if* da Linguagem *C*, conforme critérios informados pelo professor.

O *Cluster 9*, que é o *cluster* mais heterogêneo conforme Figura 5, por sua vez, tem como principal componente descritiva e discriminante, a componente *F8 (Comparação Menor ou Igual)*. Isso significa que as soluções desses *clusters*, em geral, caracterizam-se pela presença de comparações utilizando o operador <= *(menor ou igual)*. Isso nos leva a concluir que, de fato, o critério de avaliação de E_P utilizado pelo professor envolvendo expressões lógicas e estruturas de controle condicional aparece na formação dos *clusters* da *Base-A*.

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 com PCA.

Descriptive & Descriminating Features				
Cluster 0 **Size:** 5 **ISim:** 0.872 **ESim:** -0.024				
Descriptive:	F5	38.9%	F1	30.2%
Descriminating:	F10	33.5%	F1	18.2%
Cluster 1 **Size:** 7 **ISim:** 0.809 **ESim:** -0.017				
Descriptive:	F7	98.5%	F3	0.5%
Descriminating:	F7	67.4%	F5	15.9%
Cluster 2 **Size:** 10 **ISim:** 0.739 **ESim:** -0.086				
Descriptive:	F11	76.1%	F1	9.1%
Descriminating:	F11	62.0%	F5	10.9%
Cluster 3 **Size:** 15 **ISim:** 0.717 **ESim:** -0.077				
Descriptive:	F5	95.8%	F1	2.5%
Descriminating:	F5	78.0%	F1	7.0%
Cluster 4 **Size:** 13 **ISim:** 0.665 **ESim:** -0.102				
Descriptive:	F11	96.7%	F5	1.0%
Descriminating:	F11	86.1%	F5	9.3%
Cluster 5 **Size:** 13 **ISim:** 0.633 **ESim:** -0.128				
Descriptive:	F1	64.1%	F11	10.3%
Descriminating:	F1	53.0%	F5	15.4%
Cluster 6 **Size:** 7 **ISim:** 0.669 **ESim:** -0.017				
Descriptive:	F10	89.8%	F12	7.9%
Descriminating:	F10	54.3%	F5	22.3%
Cluster 7 **Size:** 13 **ISim:** 0.465 **ESim:** -0.096				
Descriptive:	F5	52.4%	F12	35.7%
Descriminating:	F5	64.6%	F12	29.6%
Cluster 8 **Size:** 10 **ISim:** 0.450 **ESim:** -0.035				
Descriptive:	F1	61.1%	F9	25.9%
Descriminating:	F1	47.1%	F9	31.0%
Cluster 9 **Size:** 7 **ISim:** 0.415 **ESim:** -0.020				
Descriptive:	F8	84.5%	F4	8.6%
Descriminating:	F8	56.1%	F5	14.7%

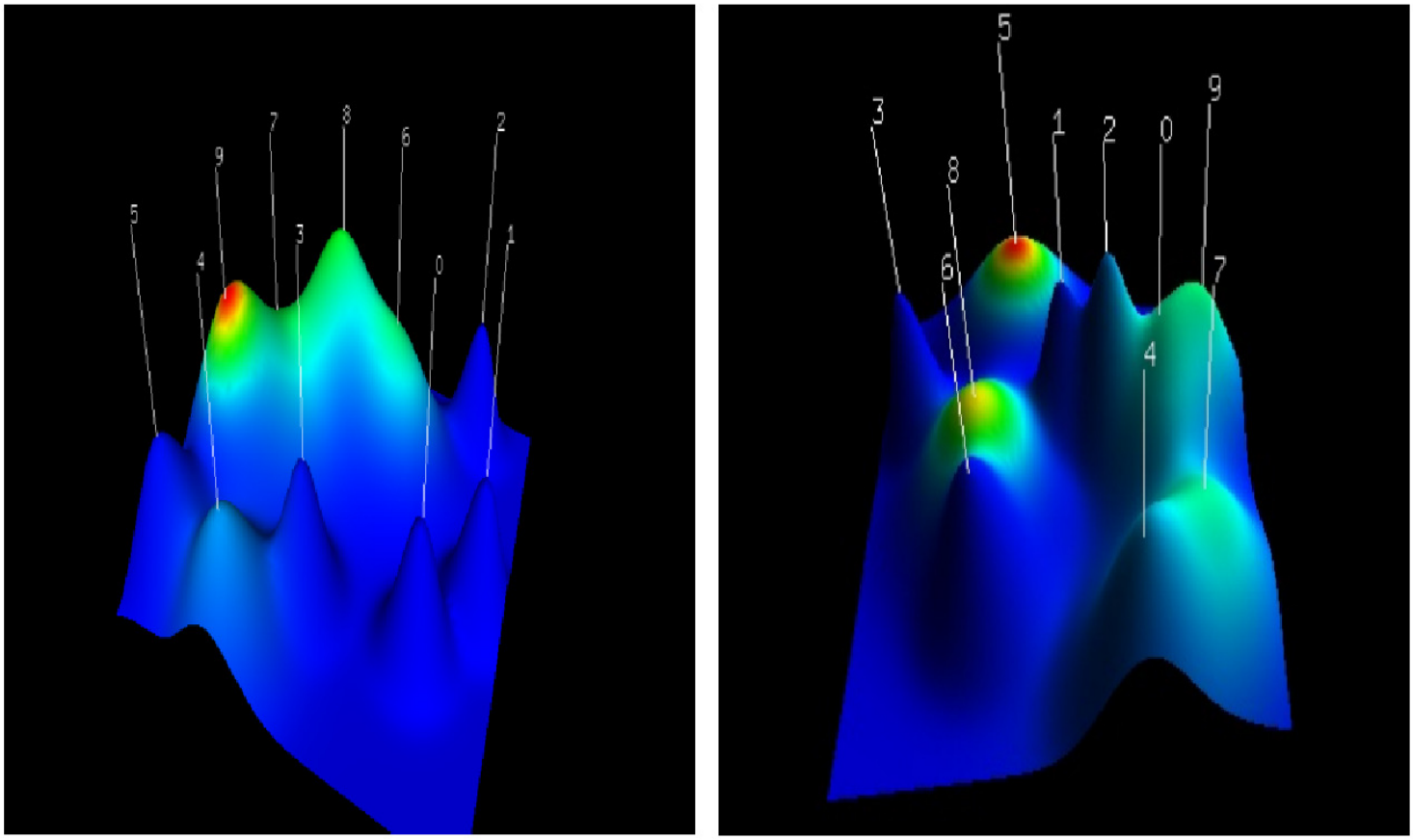
Figura 5: Componentes principais descritivas e discriminantes da Base-A com PCA.

4.2 Resultados dos experimentos na Base-B

A Figura 6 apresenta os *clusters* da *Base-B*. Assim como aconteceu com a *Base-A*, os *clusters* da *Base-B* melhor se diferenciaram com a aplicação da técnica *PCA*. Essa diferenciação é visualizada principalmente no *Cluster* 5, que é o *cluster* com pico mais vermelho na Figura 6-(B). Esse *cluster* aparece mais isolado e em maior evidência sinalizando a sua alta heterogeneidade em relação aos demais *clusters*.

Essa diferenciação do *Cluster* 5 da Figura 6–(B) pode ser confirmada na Figura 7 pelo maior valor de *ISdev* desse *cluster* e pelos altos valores de *ISIM* dos demais *clusters* da *Base-B*, com exceção do *Cluster 9*, que tem *ISIM* ligeiramente menor do que do *Cluster 5*.

A Figura 8 apresenta a organização dos *clusters* da *Base-B* seguindo os mesmos padrões de organização da *Base-A* (Figura 4). Analisando o *Cluster 5* da *Base-B* na primeira tabela à direita na Figura 8, observamos a melhor representação da diversidade das soluções, uma vez que esse *cluster* reúne amostras avaliadas por um professor com escores altos, médios e baixos.



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

Figura 6: Os clusters da Base-B.

Uma outra observação nos *clusters* da Figura 6-(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, conforme a Figura 8, ambos se aproximam pelas soluções que foram avaliadas com escores médios da *Classe D*.

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 7: Resultados de clustering da Base-B com PCA.

Os *clusters* 0, 4 e 6 estão entre os *clusters* mais homogêneos da *Base-B*, conforme a Figura 6-(B) e a Figura 9. De acordo com a Figura 8, visualiza-se que esses *clusters* assemelham-se pelas soluções que foram avaliadas com escores mais baixos.

A Figura 9 apresenta as componentes principais que mais descrevem e discriminam cada *cluster* da *Base-B*. Os *clusters* 0, 4, 6 e 7 são os *clusters* mais homogêneos e que reúnem em si predominantemente amostras que foram avaliadas com escores mais baixos.

De acordo com a Figura 9, o *Cluster* 0 tem como componente descritiva e discriminante *F4 (Entrada de Dados)*. Observa-se, dessa forma, que as soluções do *Cluster* 0 não representam gabaritos, uma vez que os gabaritos do exercício E_P são caracterizados principalmente por expressões lógicas e estruturas condicionais.

Ao contrário da *Base-A*, que reúne nos *clusters* mais homogêneos as soluções de gabarito e as principais componentes descritivas e discriminantes deles coincidem, na *Base-B*, as componentes principais dos *clusters* mais homogêneos (*clusters* 0 e 4) não coincidem. Até mesmo o *Cluster 9*, que, na Figura 6-(b), é o que mais se aproxima do *Cluster* 0, também não coincide com este nas componentes principais.

De acordo com a Figura 9, apenas os *clusters* 6 e 7 coincidem nas componentes principais descritivas e discriminantes, no entanto, há uma grande diferença no percentual de descrição e de discriminação de ambos.

Não se pode afirmar, portanto, que os *clusters* mais próximos e mais homogêneos da *Base-B* reúnem gabaritos, uma vez que estes não convergem para um padrão de acordo com suas componentes principais. No entanto, com exceção do *Cluster* 0 e do *Cluster* 2 (que só contém uma amostra), descritos, respectivamente, por *F4 (Entrada de dados)* e *F1 (Expressão aritmética)*, em todos os demais *clusters*, aparecem operações lógicas de comparação ou estruturas condicionais.

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 8: Análise de clusters da Base-B com PCA.

Logo, é muito provável que tais *clusters* não reúnam amostras de gabaritos já que neles não aparecem os critérios de avaliação que envolvem estruturas condicionais e expressões lógicas.

O *Cluster* 5 da *Base-B* é o maior *cluster* e mais heterogêneo (Figura 9). Esse *cluster* é descrito e discriminado pela componente principal *F3*. Logo, *F3* é a componente representante da diversidade de soluções. Isso significa que as comparações e as atribuições presentes no código caracterizam a diversidade de notas aplicadas pelo professor.

4.3 Conclusão dos resultados

Os resultados de análise dos *clusters* da *Base-A* e da *Base-B* indicam, portanto, que podemos reconhecer nos *clusters* mais homogêneos a mais possível representação das soluções de gabarito, pois as soluções de altos escores tendem a ser mais semelhantes (Naudé, Greyling, & Vogts, 2010). No entanto, para verificar se de fato os gabaritos estão presentes em um *cluster* mais homogêneo, é necessário observar se as componentes que descrevem esse *cluster* são as que o professor considera como chaves de gabarito e verificar os escores atribuídos pelo professor para alguns exemplos desse *cluster*.

Já as soluções de baixos escores, em geral, diferenciam-se entre si, principalmente quando há excesso de código em um programa. No entanto, podem ser semelhantes quando há pouca informação escrita nos códigos-fontes como, por exemplo, apenas os cabeçalhos dos programas. Para confirmar se o *cluster* mais homogêneo de fato reúne amostras com escores baixos, assim

Descriptive & Descriminating Features

Cluster 0	**Size:** 5	**ISim:** 0.947	**ESim:** -0.268		
Descriptive:			F4	94.3%	F1	2.2%
Descriminating:			F4	96.7%	F5	1.3%
Cluster 1	**Size:** 1	**ISim:** 1.000	**ESim:** -0.146		
Descriptive:			F2	93.3%	F3	4.8%
Descriminating:			F2	71.7%	F4	14.0%
Cluster 2	**Size:** 1	**ISim:** 1.000	**ESim:** -0.140		
Descriptive:			F1	98.9%	F4	0.6%
Descriminating:			F1	76.3%	F4	16.6%
Cluster 3	**Size:** 1	**ISim:** 1.000	**ESim:** -0.020		
Descriptive:			F5	72.6%	F3	17.4%
Descriminating:			F5	68.8%	F2	10.7%
Cluster 4	**Size:** 4	**ISim:** 0.978	**ESim:** -0.040		
Descriptive:			F3	86.3%	F2	8.9%
Descriminating:			F3	83.5%	F4	7.0%
Cluster 5	**Size:** 8	**ISim:** 0.795	**ESim:** -0.177		
Descriptive:			F3	97.6%	F1	1.2%
Descriminating:			F3	84.9%	F2	6.8%
Cluster 6	**Size:** 2	**ISim:** 0.855	**ESim:** -0.046		
Descriptive:			F5	65.2%	F3	18.7%
Descriminating:			F5	64.1%	F4	21.4%
Cluster 7	**Size:** 7	**ISim:** 0.869	**ESim:** -0.015		
Descriptive:			F5	33.1%	F3	32.6%
Descriminating:			F3	68.6%	F5	30.1%
Cluster 8	**Size:** 7	**ISim:** 0.835	**ESim:** -0.011		
Descriptive:			F4	91.0%	F5	3.4%
Descriminating:			F4	47.3%	F5	27.8%
Cluster 9	**Size:** 3	**ISim:** 0.775	**ESim:** 0.014		
Descriptive:			F5	88.7%	F3	5.8%
Descriminating:			F5	49.7%	F4	24.6%

Go to Top

Figura 9: Componentes principais descritivas e discriminantes da Base-B com PCA.

como no *cluster* de gabaritos, é necessário verificar os escores atribuídos pelo professor a alguns exemplos de soluções desse *cluster*.

Nos *clusters* mais heterogêneos, que se caracterizam pelo alto valor de desvio interno (*ISdev*), podemos obter a melhor representação dos critérios de avaliação de um professor na diversidade. Isso porque os exemplos de soluções presentes nesses *clusters* poderão ter mais possibilidades de serem rotuladas com diferentes classes de escores, sejam eles de valores altos, médios ou baixos.

4.4 Extensões do método

Com a finalidade de estender o método proposto por Oliveira, Souza, et al. (2016), em novos experimentos, aplicamos um algoritmo de *Graph Clustering* sem *PCA* para reconhecimento de modelos de soluções de exercícios de programação.

Os grafos são estruturas formadas por um conjunto de vértices e de arestas. Em um grafo, os vértices são nós ou representações de objetos e as arestas são conexões entre pares de vértices. O algoritmo *Graph Clustering* realiza a tarefa de agrupar os vértices de um grafo em *clusters* de forma que haja muitas arestas dentro dos *clusters* e poucas entre eles (Schaeffer, 2007).

A vantagem de se utilizar o *Graph Clustering* é que o número de *clusters* fornecido como entrada pode ser ampliado pelo próprio algoritmo, caso um conjunto de amostras não se adeque ao

índice de similaridade mínimo definido para os *clusters* formados. Dessa forma, é recomendável que o valor do número de *clusters* fornecido como entrada do algoritmo seja 1 (um) para que, caso seja necessário, o próprio algoritmo forme novos *clusters*, conforme o índice de similaridade limite fornecido como entrada para o algoritmo para a formação dos *clusters*.

Neste trabalho, utilizamos o algoritmo de *Graph Clustering* do software *Cluto* (Karypis, 2002). A medida de similaridade escolhida foi o *cosseno*. Já os parâmetros de corte escolhidos após vários testes foram o *edgeprune=0.5* e o *vtxprune=0.6*. O primeiro parâmetro separa certas arestas do grafo dos vizinhos mais próximos que tendem a conectar vértices pertencentes a diferentes *clusters*. Já o segundo parâmetro separa certos vértices do grafo de vizinhos mais próximos que não se agrupam em qualquer um dos *clusters* formados (Oliveira, Monroy, et al., 2015).

Nos testes iniciais com a *Base-A*, o algoritmo com os parâmetros de corte definidos gerou dois *clusters* (0 e 1) com o índice de similaridade interna mínimo fixado em 60% (isto é, com *vtxprune=0.6*) e um outro *cluster* (-1) contendo todas as amostras que não foram inseridas nos outros dois *clusters*. Em resumo, o *cluster* -1 reúne aquelas amostras que não possuem pelo menos 60% de similaridade com as amostras dos dois *clusters* ou não contêm um peso de conexão forte de pelo menos 50% entre os vértices de diferentes *clusters* (*edprune*=0.5).

Chamamos de *RG* o *Cluster* 0 por conter a melhor representação de gabaritos e de *RD*, o *cluster* -1, por reunir as amostras com vértices presentes em diferentes *clusters* ou em nenhum deles. Já o *Cluster* 1, chamamos de *RN*, por não conter representações de gabaritos.

A Tabela 4 apresenta os resultados dos testes iniciais de aplicação do algoritmo de *Graph Clustering* para seleção de gabaritos e da diversidade de soluções de exercícios de programação.

Tabela 4: Reconhecimento automático de rubricas por Graph Clustering.

Análise de Clusters							
		Classes (em %)					
Cluster	No.de Alunos	A	B	C	D	E	ISim
(-1) RD	57	22.8	19.3	22.8	15.8	19.3	–
(0) RG	10	70.0	10	10	0	10	0.9070
(1) RN	33	0	18.2	42.4	27.3	12.1	0.8370

De acordo com a Tabela 4, observa-se que o *Cluster RD* reúne exemplos de todas as classes de soluções de exercícios de programação em proporções bem próximas. Além de favorecer a seleção de rubricas de soluções com diferentes escores, essa distribuição tem uma melhor representação de classes para formar o conjunto de treino de um sistema de avaliação semiautomática.

O *Cluster RG*, por sua vez, conforme a Tabela 4, sendo bem homogêneo (*ISIM=0.90*), reúne de forma bem distinta 70% das soluções de *Classe A*, isto é, da classe de gabaritos. Já o *Cluster RN* não contém amostras da *Classe A*.

Os testes iniciais com o algoritmo de *Graph Clustering* sem aplicação da técnica *PCA* já apontam para resultados promissores. Os próximos passos de experimentação desse método são realizar testes em novas bases e desenvolver um critério para combinação de parâmetros de forma a identificar os melhores valores de *edgeprune* e *vtxprune* para selecionar de forma mais precisa, em diferentes tipos de bases de exercícios de programação, a representação de gabaritos e a representação da diversidade de soluções.

5 Conclusão

Este trabalho apresentou uma estratégia de reconhecimento automático de representações de rubricas a partir de agrupamentos de soluções de exercícios de programação. Os resultados alcançados indicam que a estratégia proposta reconhece exemplos representativos da diversidade de soluções de programação para composição de rubricas.

A técnica de *Análise de Componentes Principais* melhorou o reconhecimento de soluções representativas dos critérios de avaliação dos professores a partir dos *clusters* formados pelo algoritmo *Bissecting K-means*. Uma limitação dessa estratégia, porém, é a escolha do melhor número de *clusters* de forma a capturar nos *clusters* formados exemplos representativos de gabaritos e da diversidade de soluções.

Como extensão do trabalho de Oliveira, Souza, et al. (2016), utilizamos o algoritmo *Graph Clustering* sem aplicação da técnica *PCA* para reconhecimento de representações de rubricas e de conjunto de treino para sistemas de avaliação semiautomática de exercícios de programação. Os testes iniciais apresentaram resultados promissores.

Como trabalhos futuros a partir deste, propomos realizar mais testes de reconhecimento de rubricas em exercícios de programação considerando também a dificuldade dos exercícios. Além disso, destacamos a necessidade de identificar um critério para escolher a melhor combinação de parâmetros do algoritmo de *Graph Clustering* para seleção de amostras e características que melhor representem gabaritos e a diversidade de soluções de programação.

Concluindo, as principais contribuições deste trabalho para a aprendizagem de programação são auxiliar o trabalho de avaliação de professores através da composição de representações de rubricas e possibilitar *feedbacks* mais claros e mais objetivos para estudantes de programação.

Agradecimentos

Agradecemos à Fundação de Apoio à Pesquisa e Inovação do Espírito Santo (FAPES) pelo apoio dado ao projeto *Tecnologias de Avaliação Semi-automática da Aprendizagem de Programação (do EDITAL 006/2014 – Universal Projeto Individual de Pesquisa)* do qual resultou esta publicação.

References

Karypis, G. (2002). *Cluto-a clustering toolkit* (Tech. Rep.). Retrieved from http://www.dtic.mil/get-tr-doc/pdf?AD=ADA439508

Kwon, H., & 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. Retrieved from http://www.ndsl.kr/ndsl/search/detail/article/articleSearchResultDetail.do?cn=JAKO200532056743373 [GS Search]

Lindenbaum, M., Markovitch, S., & Rusakov, D. (2004, Feb 01). Selective

sampling for nearest neighbor classifiers. *Machine Learning*, *54*(2), 125–152. doi: 10.1023/B:MACH.0000011805.60520.fe [GS Search]

Naudé, K. A., Greyling, J. H., & Vogts, D. (2010). Marking student programs using graph similarity. *Computers & Education*, *54*(2), 545 - 561. doi: 10.1016/j.compedu.2009.09.005 [GS Search]

Oliveira, M. G., Basoni, H., Saúde, M., & 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 - Volume 1: KDIR, (IC3K 2014)* (p. 465-472). SciTePress. doi: 10.5220/0005159304650472 [GS Search]

Oliveira, M. G., Monroy, N., Daher, P., & Oliveira, E. (2015). Representação da diversidade de componentes latentes em exercícios de programação para classificação de perfis. In *IV Congresso Brasileiro de Informática na Educação (CBIE 2015)* (pp. 1177–1186). Maceió: Anais do SBIE 2015. doi: 10.5753/cbie.sbie.2015.1177 [GS Search]

Oliveira, M. G., Nogueira, M. A., & Oliveira, E. (2015). 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. Retrieved from http://www.lbd.dcc.ufmg.br/colecoes/wei/2015/010.pdf [GS Search]

Oliveira, M. G., Reblin, L., & Oliveira, E. (2016). Sistema de apoio a 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. Retrieved from http://ebooks.pucrs.br/edipucrs/anais/csbc/assets/2016/wei/40.pdf [GS Search]

Oliveira, M. G., Souza, M., Reblin, L. L., & Oliveira, E. (2016). Reconhecimento automático de representações de rubricas em agrupamentos de soluções de exercícios de programação. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016)* (pp. 1106–1115). doi: 10.5753/cbie.sbie.2016.1106 [GS Search]

Olmos, R., Guillermo, J. B., Luzón, J. M., Martín-Cordero, J. I., & Leao, J. A. (2016). Transforming LSA space dimensions into a rubric for an automatic assessment and feedback system. *Information Processing & Management*, *52*(3), 359 – 373. doi: 10.1016/j.ipm.2015.12.002 [GS Search]

Panadero, E., & Jonsson, A. (2007). The use of scoring rubrics: Reliability, validity and educational consequences. *Educational Research Review*, *2*(2), 130 - 144. doi: 10.1016/j.edurev.2013.01.002 [GS Search]

Perlman, C. (2003). Performance assessment: Designing appropriate performance tasks and scoring rubrics. Retrieved from https://eric.ed.gov/?id=ED480070 [GS Search]

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, *1*(1), 27 - 64. doi: 10.1016/j.cosrev.2007.05.001 [GS Search]

Spalenza, M., Oliveira, E., Oliveira, M., & Nogueira, M. (2016). Uso de mapa de características na avaliação de textos curtos nos ambientes virtuais de aprendizagem. In *Anais do SBIE 2016* (pp. 1165–1174). doi: 10.5753/cbie.sbie.2016.1165 [GS Search]

Srikant, S., & Aggarwal, V. (2014). A system to grade computer programming skills using machine learning. In *Proceedings of the 20th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1887–1896). New York, NY, USA: ACM. doi: 10.1145/2623330.2623377 [GS Search]

Tuia, D., Pasolli, E., & Emery, W. (2011). Using active learning to adapt remote

sensing image classifiers. *Remote Sensing of Environment*, *115*(9), 2232 - 2242. doi: 10.1016/j.rse.2011.04.022 [GS Search]

Wall, M. E., Rechtsteiner, A., & Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In D. P. Berrar, W. Dubitzky, & M. Granzow (Eds.), *A practical approach to microarray data analysis* (pp. 91–109). Boston, MA: Springer US. doi: 10.1007/0-306-47815-3_5 [GS Search]

Yamamoto, M., Umemura, N., & Kawano, H. (2018). Automated essay scoring system based on rubric. In R. Lee (Ed.), *Applied computing & information technology* (pp. 177–190). Cham: Springer International Publishing. doi: 10.1007/978-3-319-64051-8_11 [GS Search]