Sistema de Apoio à Avaliação de Atividades de Programação por Reconhecimento Automático de Modelos de Soluções

Márcia G. de Oliveira [1], Leonardo Leal Reblin [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
Vitória – ES – Brasil

clickmarcia@gmail.com, elias@acm.org

Abstract. *Developing a computer program is a process of solving problems that results in several possibilities of solutions. Thus, the assessment of programming exercises demands a lot of efforts both in manual evaluation when several potential solutions are analyzed, as the automatic assessment, when many solutions should be provided as inputs. In order to assist teachers to identify solution models from programs developed by students, this paper proposes a clustering-based system to recognize solution models and to map them in scores assigned by teachers. The first application of this system experiments on two real bases of programs developed by programming students showed promising results.*

Keywords*: Solution Models, Programming, Clustering, Rubrics.*

Resumo. *O desenvolvimento de um programa de computador é um processo de resolução de problema que resulta em várias possibilidades de soluções. Dessa forma, a avaliação de exercícios de programação demanda muito esforço do professor tanto na avaliação manual, quando analisam-se várias possibilidades de soluções, quanto na avaliação automática, quando vários modelos de soluções devem ser fornecidos como entradas. Com o objetivo de auxiliar professores na identificação de modelos de soluções a partir de programas desenvolvidos por alunos, este trabalho propõe um sistema baseado em clustering para reconhecimento de modelos de soluções e para mapeamento dessas soluções em escores atribuídos por professores. Os primeiros experimentos de aplicação desse sistema em duas bases de programas desenvolvidos por estudantes de programação apresentaram resultados promissores.*

Palavras-chave*: Modelos de soluções, Programação, Clustering, Rúbricas.*

1. Introdução

O desenvolvimento de programas de computador como parte do processo de aprendizagem de programação é uma prática que resulta em várias possibilidades de soluções para

cada problema a ser resolvido. Dessa forma, o processo de avaliação de exercícios de programação demanda de professores um grande esforço de análise de diversas soluções para descobrir modelos de soluções, no caso de avaliações manuais; e de composição de modelos de soluções, no caso de avaliações automáticas que requerem entradas de gabaritos. Nesse último caso, o problema se amplia, uma vez que nem sempre o professor conhece antecipadamente todas as possíveis soluções para um exercício de programação.

Problemas similares ao problema de reconhecer modelos de soluções ou representações da diversidade a partir de um conjunto de soluções de exercícios são tratados em muitos trabalhos científicos como um processo de seleção de amostras representativas de um conjunto de padrões como, por exemplo, textos, imagens e pessoas.

Para resolver essa classe de problemas, têm-se desenvolvido, várias estratégias tecnológicas de amostragem seletiva ([Lindenbaum et al. 2004], de aprendizagem ativa [Tuia et al. 2011, Oliveira et al. 2014] e de rúbricas [Kwon and Jo 2005] para apoiar docentes na avaliação manual e automática de exercícios. Essas tecnologias, em geral, são desenvolvidas para sistemas de aprendizagem supervisionada, isto é, sistemas que aprendem através de exemplos para tomarem decisões de forma autônoma [Kotsiantis 2007].

Para o domínio da programação, uma sugestão para reconhecer a variedade de soluções é a aplicação de algoritmos de *clustering* [Naude et al. 2010]. Seguindo essa ideia, com o objetivo de auxiliar professores na avaliação de exercícios de programação, propomos um sistema de reconhecimento automático de possíveis soluções e de mapeamento dessas soluções em escores atribuídos por professores.

O reconhecimento de modelos de soluções e da diversidade destas é realizado a partir de uma coleção de programas em Linguagem C desenvolvidos por estudantes para um exercício de programação. Já o mapeamento em escores consiste em solicitar ao professor a atribuição de escores para os modelos de soluções reconhecidos. Esses modelos de soluções são, desse modo, utilizados como gabaritos de entrada de um avaliador automático ou como treino de avaliadores de aprendizagem supervisionada.

Este trabalho foca-se na seleção de modelos de soluções para composição de gabaritos de exercícios de programação. No entanto, esses modelos podem também ser utilizados por professores para gerar exemplos representativos de padrões de desempenhos informando critérios de avaliações no esquema de rúbricas [Mertler 2001].

As contribuições do sistema proposto são agilizar o processo de avaliação identificando automaticamente os potenciais gabaritos para cada exercício de programação e oferecer aos estudantes maior clareza do processo avaliativo fornecendo-lhes uma ampla variedade de soluções explicadas a partir dos escores atribuídos por professores.

Este trabalho está organizado conforme a ordem a seguir. Na Seção 2, apresentamos os trabalhos relacionados que oferecem mecanismos de seleção automática de modelos de soluções de exercícios. Na Seção 3, descrevemos a nossa proposta metodológica para reconhecimento de soluções e o algoritmo de *clustering* utilizado. Na Seção 4, apresentamos o primeiro experimento realizado em duas bases de soluções de exercícios desenvolvidas por estudantes de programação e os resultados alcançados. Na Seção 5, concluimos com as considerações finais e trabalhos futuros.

2. Trabalhos Relacionados

A estratégia tecnológica deste trabalho foi desenvolvida como uma extensão do sistema *PCodigo* [Oliveira et al. 2015b], visando reduzir esforço do especialista humano [Oliveira et al. 2014] através da seleção de amostras de treino para sistemas de aprendizagem supervisionada [Lindenbaum et al. 2004, Tuia et al. 2011, Oliveira et al. 2015a] e da apresentação de critérios de avaliação no esquema de rúbricas [Kwon and Jo 2005].

O *PCodigo* é um sistema de apoio à prática assistida de programação que, integrado ao *Moodle*, recebe soluções de atividades de programação submetidas por alunos, executa-as em massa, realiza análises e emite relatórios de avaliação para professores como, por exemplo, relatórios de plágio [Oliveira et al. 2015b].

A redução de esforço do especialista humano no fornecimento de exemplos para treino de sistemas de aprendizagem supervisionada tem sido uma temática abordada em muitos trabalhos de pesquisa. O trabalho de [Oliveira et al. 2014], por exemplo, propõe uma estratégia de aprendizagem ativa que visa selecionar uma quantidade mínima de exemplos de textos do *Twitter* para um especialista classificá-los e, a partir desses exemplos, classificar automaticamente uma grande quantidade de outros *tweets*.

Uma estratégia mais recente, que é aplicada para selecionar amostras representativas de exercícios de programação, é o método de seleção não-aleatória de amostras de treino para classificação de perfis proposto por [Oliveira et al. 2015a].

Nesse último método, visa-se reduzir o esforço humano na atribuição de escores aos exemplos selecionados e, da máquina, na classificação automática, uma vez que realiza-se redução de dimensionalidade da matriz de sessenta dimensões que representa soluções de exercícios em Linguagem C. Para a redução de dimensionalidade, utiliza-se a análise fatorial e, para a seleção de amostras que representem a diversidade de uma coleção de exercícios de programação, utiliza-se o *Clustering em Grafo*, [Oliveira et al. 2015a].

A estratégia de seleção de amostras representativas para treino de sistemas de aprendizagem supervisionada também pode ser uma opção para definir critérios de avaliação de um professor no esquema de rúbricas. Seguindo essa ideia, o trabalho de [Kwon and Jo 2005] aplica uma ferramenta que possibilita a análise das preferências dos professores e as características dos alunos para cada rúbrica, explorando a classificação e as regras de associação utilizando técnicas de *data mining*.

O diferencial da nossa estratégia em relação às estratégias apresentadas é que utilizamos processos de *clustering* ajustados pelo número de *clusters* e pelos índices de similaridade interna e externa desses *clusters* para selecionar modelos de soluções e representações da diversidade de uma coleção de exercícios.

3. Sistema de Reconhecimento de Soluções de Exercícios

O sistema de reconhecimento de soluções proposto neste trabalho é uma extensão do *PCodigo*, que é um sistema de apoio à prática assistida de programação através da execução em massa e análise de exercícios de programação em Linguagem C [Oliveira et al. 2015b].

O processamento do *PCodigo* se inicia após o professor disponibilizar via *Moodle*, na *Visão do Professor*, *Exercícios de Programação* e após os alunos realizarem

Submissões de soluções para esses exercícios [Oliveira et al. 2015b].

Em resumo, as etapas de processamento do *PCodigo* consistem em receber soluções de atividades de programação submetidas por alunos, executá-as, analisá-las para avaliação de indícios de plágios e emitir relatórios de avaliação para professores.

A Figura 1 apresenta a arquitetura do sistema de análise de soluções bem como a sua integração ao *Pcodigo*.

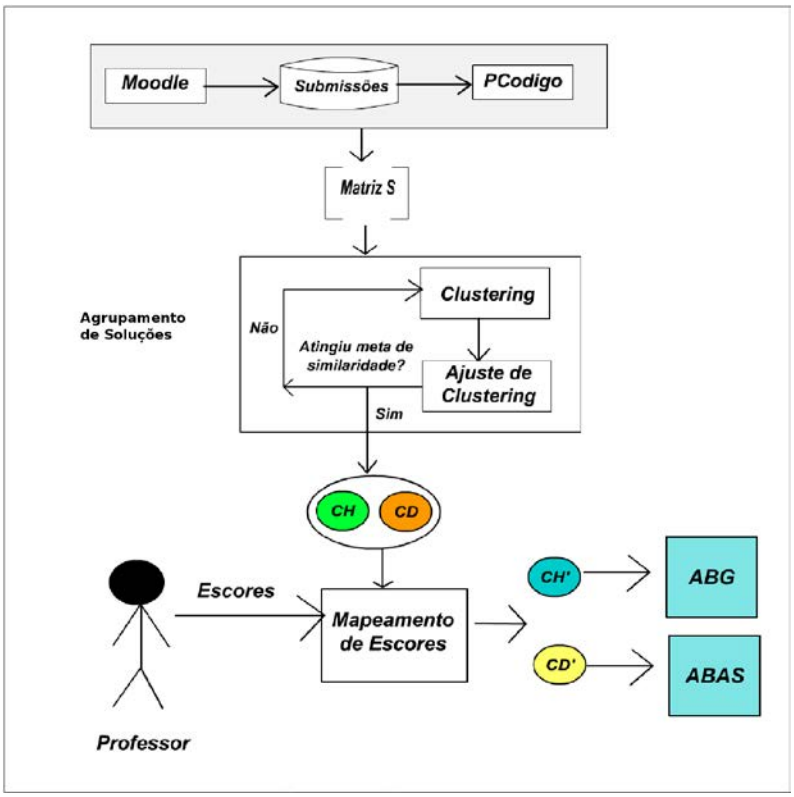

Figura 1. Sistema de análise de soluções

Conforme a Figura 1, o *Pcodigo* é integrado ao *Moodle* para receber as *Submissões* de exercícios e fornece como entrada ao sistema de reconhecimento de soluções uma *Matriz S* contendo as submissões vetorizadas em 60 dimensões. Nessa matriz, cada dimensão de um vetor representa a frequência de ocorrência de uma palavras-chave ou função da Linguagem C em código-fonte ou um valor lógico (1=Verdadeiro; 0= Falso) informando, por exemplo, se um programa compila e executa [Oliveira et al. 2015b].

No processamento de *Agrupamento de Soluções* da Figura 1, a *Matriz S* é submetida ao algoritmo de *Clustering Bissecting K-means* do software *Cluto* [Karypis 2003].

Sabendo que há mais possibilidades das melhores soluções de um exercício se assemelharem por aproximarem-se de um padrão de resolução ensinado por um professor [Naude et al. 2010], para obter um agrupamento com os melhores modelos de soluções, realizamos o *Ajuste de Clustering*. Para isso, repetimos os processos de *Formação de Agrupamentos* até que, pelo número de *clusters* escolhido, sejam obtidos *clusters* com maiores índices de similaridade interna (ISIM) e de similaridade externa (ESIM).

Após o processamento de *Agrupamento de Soluções*, são selecionados dois *clus-*

ters: os *clusters CH* e *CD*. O *cluster CH* representa o agrupamento com as melhores soluções e o *cluster CD*, o agrupamento com maior representação da diversidade.

No módulo de *Mapeamento de Escores*, os *clusters CH* e *CD* são apresentados a um professor para que este atribua escores para cada um dos padrões desses *clusters*. Em seguida, os *clusters CH'* e *CD'* contendo padrões avaliados pelo professor são enviados como entradas para os algoritmos de avaliação automática de exercícios *ABG (Avaliação Baseada em Gabaritos)* e *ABAS (Avaliação Baseada em Aprendizagem Supervisionada)*.

Os padrões de *CH'* enviados para o algoritmo de *ABG* podem ser utilizados como modelos de soluções para atribuição de escores a outras soluções muito semelhantes a estas. Já os padrões do *CD'* enviados para o algoritmo de *ABAS*, em um passo mais avançado, representando a diversidade de um conjunto de soluções composto de padrões com altos, baixos e médios escores, podem ser utilizados como treino do avaliador automático para que este aprenda o padrão de correção de um exercício e realize predições de escores para outros exemplos de soluções desse mesmo exercício.

Uma outra aplicação da metodologia deste trabalho é utilizar os padrões avaliados dos *clusters CH'* e *CD'* como referências de avaliação para diferentes níveis de desempenhos em um esquema de rúbricas. Nesse caso, os professores apresentariam de forma mais clara seus critérios de avaliação de cada exercício de programação aplicado, dando possibilidades ao aluno de refletir sobre sua própria aprendizagem.

4. Experimentos e resultados

Para a experimentação do sistema de reconhecimento de soluções, 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].

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 de 40 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]. Essas bases compostas de amostras em formato vetorial são geradas pelo *PCodigo* nos processos de normalização e indexação de submissões [Oliveira et al. 2015b].

No sistema de reconhecimento de soluções, essas bases são submetidas no formato da Matriz *S* (Figura 1) ao algoritmo de *clustering Bissecting K-means* [Karypis 2003]. O *Ajuste de Clustering* é realizado até que se alcance o melhor número de *clusters* com maiores valores de similaridade interna e externa (*ISIM* e *ESIM*, respectivamente).

Após escolher o número de *clusters* mais adequado, selecionamos o *cluster* mais provável de ter as melhores soluções (*CH*) e o *cluster* com a maior representação da diversidade de soluções (*CD*).

Para análise dos *clusters* formados, um *script* gera um relatório com a soma das dimensões de cada vetor, o que nos fornece um indicativo de esforço de programação do estudante. Em geral, as melhores soluções apresentam menos instruções e menos linhas de código, o que pode ser observado nos relatórios das figuras 2 e 3.

4.1. Resultados

Ao aumentarmos o número de *clusters* no processamento de *Ajuste de Clustering*, alcançamos valores maiores de ISIM e de ESIM para os *clusters* formados. Isso nos possibilita obter *clusters* mais homogêneos e identificar soluções atípicas, isto é, soluções que são únicas em um *cluster*.

A Tabela 1, relativa à *Base-A* de 100 amostras, apresenta como os valores de *ISIM* máximo e mínimo aumentam e o *ESIM* mínimo diminui, indicando que, aumentando o número de *clusters*, formam-se *clusters* mais homogêneos e distintos de outros *clusters*.

Análise de Similaridade ISIM-ESIM

No. de clusters	ISim Máximo (%)	ISim Mínimo (%)	ESim Máximo(%)	ESim Mínimo (%)
1	69.8	69.8	100	100
2	80.8	76	59.5	59.5
3	87.6	76.1	67.9	59.5
4	87.6	72.2	67.9	49.2
5	92.6	72.2	70.2	49.2
6	92.6	72.2	70.5	49.2
7	92.6	72.2	70.5	49.2
8	92.6	83.8	70.5	42.0
9	92.6	79.1	72.6	42.0
10	**100**	**85.0**	**72.6**	**30.5**

Tabela 1. Análise de similaridade por número de clusters da Base-A

Para termos *clusters* mais homogêneos, conforme a Tabela 1, escolhemos dividir a *Base-A* em dez *clusters*. Na Tabela 2, apresentamos como esses dez *clusters* se organizaram conforme os seus valores de ISIM e ESIM.

Análise de similaridade dos clusters

Cluster	Tamanho	ISim (%)	ESim (%)
0	1	100	30.5
1	2	85.4	42.0
2	4	86.8	53.1
3	5	92.2	63.4
4	21	85.2	61.8
5	22	92.6	70.3
6	8	89.9	67.7
7	10	89.1	70.2
8	14	91.3	72.6
9	13	85.0	70.5

Tabela 2. Análise de similaridade dos clusters da Base-A

Uma vez que os *clusters* são organizados de forma crescente de 0 a N - 1 (N = Número de *Clusters*), os maiores valores de ISIM-ESIM estão nos *clusters* identificados com valores maiores. Da mesma forma, os *clusters* mais apertados e mais distantes dos outros, isto é, com menor *ESIM*, são identificados com valores menores [Karypis 2003].

A Figura 2 apresenta os *clusters* 0, 1, 2, 3, 4 e 9 da *Base-A*. Nessas tabelas, *cluster* é o identificador do *cluster*, *índice* é o rótulo de cada amostra dentro dos *clusters*, *nota* é o escore atribuído pelo professor a uma solução e *somalinha* é a soma dos valores dos atributos, isto é, das dimensões de um vetor representando uma solução desenvolvida por um

aluno. Os escores altos são marcados de verde, os médios, de amarelo e os escores baixos, de lilás. Observa-se que o *Cluster 4* apresenta uma maior diversidade de escores médios e altos. Por outro lado, o *Cluster 9* apresenta-se mais homogêneo com predominância das soluções de altos escores.

Cluster	Indice	Nota	Somalinha
0	al21	3.0	7
1	al89	4.1	10
1	al60	3.75	6
2	al79	2.5	128
2	al61	4.8	67
2	al48	3.0	151
2	al64	2.0	84

Cluster	Indice	Nota	Somalinha
4	al77	3.7	39
4	al96	5.0	51
4	al52	3.25	52
4	al78	0.5	99
4	al43	5.0	57
4	al38	5.0	57
4	al53	4.75	88
4	al100	4.0	78
4	al49	4.0	51
4	al46	5.0	52
4	al25	3.25	70
4	al28	3.75	99
4	al74	5.0	56
4	al47	3.5	33
4	al72	4.0	80
4	al1	4.25	75
4	al22	5.0	51

Cluster	Indice	Nota	Somalinha
3	al12	3.75	66
3	al37	3.25	45
3	al8	4.0	37
3	al73	3.6	63
3	al19	3.5	79

Cluster	Indice	Nota	Somalinha
9	al39	4.5	46
9	al26	5.0	50
9	al34	4.75	56
9	al2	4.0	84
9	al83	4.3	44
9	al50	4.5	39
9	al62	2.5	37
9	al32	4.75	79
9	al56	1.5	113
9	al95	3.7	99
9	al69	4.75	75
9	al98	1.2	41
9	al15	3.75	104

Figura 2. Análise de clusters da Base-A

Para a *Base-A*, concluimos, portanto, que o *Cluster 9* é o mais indicado para ser o *cluster CH*, isto é, o *cluster* homogêneo com mais possibilidades de representar as soluções de altos escores, isto é, os modelos de soluções.

Já o *Cluster 4* é o mais indicado para ser o *cluster CD*, uma vez que apresenta maior variedade de soluções com escores altos e médios, que são os escores predominantes na *Base-A*.

Vale destacar que, assumindo que desconhecemos os escores das amostras reunidas nos *clusters*, nós identificamos o *cluster CH* pelo maior valor de identificação do *cluster*, isto é, pelo maior valor de ISIM-ESIM. Já, para identificar o *cluster CD*, selecionamos o *cluster* mais cheio com menor ISIM-ESIM.

A Tabela 3 apresenta como os valores de ISIM e ESIM dos *clusters* da *Base-B* variam à medida que o número de *clusters* aumenta. Para uma divisão de 10 *clusters* temos, portanto, maior ISIM máximo, maior ISIM mínimo, maior ESIM máximo e menor ESIM mínimo. Isso significa que a divisão em 10 *clusters* da *Base-A* oferece maior possibilidade de formar *clusters* mais homogêneos e bem distintos dos demais *clusters*.

Análise de Similaridade ISIM-ESIM

No. de clusters	ISim Máximo (%)	ISim Mínimo (%)	ESim Máximo(%)	ESim Mínimo (%)
1	82.8	82.8	—-	—-
2	90.0	78.9	73.0	73.0
3	94.7	78.9	79.4	73.0
4	100	85.1	79.4	51.7
5	100	87.3	79.4	51.7
6	100	87.3	79.7	51.7
7	100	91.6	79.7	51.7
8	100	91.6	81.8	51.7
9	100	92.9	81.8	51.7
10	**100**	**95.0**	**82.8**	**51.7**

Tabela 3. Análise de similaridade por número de clusters da Base-B

Da mesma forma que analisamos os *clusters* da *Base-A*, analisamos a *Base-B* identificando os *clusters CH* e *CD*. Mas vale destacar, conforme a Figura 3, que os *clusters* da *Base-B* apresentaram-se mais homogêneos e com as soluções inéditas isoladas nos primeiros *clusters*.

Cluster	Indice	Nota	Somalinha
0	al28	3.75	99
1	al3	3.25	101
2	al79	2.5	128

Cluster	Indice	Nota	Somalinha
3	al22	5.0	51

Cluster	Indice	Nota	Somalinha
4	al36	3.0	50
4	al84	3.0	82
4	al13	3.75	45

Cluster	Indice	Nota	Somalinha
5	al87	3.2	80
5	al91	3.2	95
5	al80	3.7	98
5	al35	1.75	89
5	al63	1.5	108

Cluster	Indice	Nota	Somalinha
6	al44	3.75	83
6	al82	2.7	99

Cluster	Indice	Nota	Somalinha
7	al32	4.75	79
7	al33	4.5	55
7	al23	3.0	98

Cluster	Indice	Nota	Somalinha
8	al2	4.0	84
8	al69	4.75	75
8	al40	3.5	133
8	al50	4.5	39
8	al20	3.5	88
8	al97	2.7	91
8	al26	5.0	50
8	al7	2.0	82
8	al67	3.75	78
8	al76	3.5	109
8	al94	2.7	88
8	al81	3.2	94
8	al14	3.25	112
8	al71	4.2	90
8	al9	2.75	85
8	al39	4.5	46
8	al60	3.75	6
8	al19	3.5	79
8	al29	3.5	93

Cluster	Indice	Nota	Somalinha
9	al30	3.5	86
9	al92	5.0	46
9	al31	3.25	105

Figura 3. Análise de clusters da Base-B

Para a *Base-B*, selecionamos, conforme a Figura 3, o *Cluster 9* como o *cluster CH* e o *Cluster 8*, como o *cluster CD*. Para essa seleção, foram utilizados os mesmos critérios de seleção dos *clusters CH* e *CD* da *Base-A*.

Concluindo, os resultados da *Base-A* e da *Base-B* indicam que é possível selecionar modelos de soluções e amostras representando a diversidade de um conjunto de exercícios através de processos de *clustering*. No entanto, para gerar os conjuntos *CH'* e *CD'* com as amostras de *CH* e *CD*, respectivamente, pontuadas por um professor, demanda-se ainda esforço de correção de todas as amostras dos *clusters CH* e *CD*.

Para reduzir o esforço do professor de corrigir todas as amostras desses *clusters*, propomos, como próximos trabalhos, desenvolver estratégias para reduzir os conjuntos

CH' e *CD'* através da seleção das amostras mais representativas dos modelos de soluções e da diversidade de exercícios presentes nos *clusters CH* e *CD*, respectivamente.

5. Considerações Finais

Este trabalho de pesquisa apresentou uma proposta inicial de sistema de seleção de modelos de soluções e da diversidade de soluções de exercícios de programação. Uma vantagem desse sistema é que modelos de soluções reconhecidos podem ser utilizados como entradas de sistemas baseados em gabaritos e as soluções de representação da diversidade como treino de sistemas de avaliação automática de aprendizagem supervisionada.

Uma outra vantagem do sistema proposto é auxiliar professores na identificação de modelos de soluções de um exercício para apresentação de critérios de avaliação no esquema de rúbricas.

A principal limitação do sistema proposto é determinar o número mínimo de *clusters* que possibilita a melhor formação de agrupamentos dos modelos de soluções e *clusters* mais reduzidos.

Como trabalhos futuros a partir deste, propomos então desenvolver pesquisas para determinar esse menor número máximo de *clusters* e realizar transformações nas bases de exercícios com técnicas de redução de dimensionalidade como a análise fatorial [Oliveira et al. 2015a] de forma a melhorar o processo de *clustering*.

Com essas propostas de trabalhos futuros, teríamos boas possibilidades de formar *clusters* menores reunindo os modelos de soluções e as amostras de representação da diversidade, o que implicaria em menor esforço de correção dos professores e mais precisão na escolha dos modelos de soluções.

Concluindo, as principais contribuições do sistema proposto para a aprendizagem de programação é agilizar o processo de envio de *feedbacks* de professores, principalmente se estes utilizarem sistemas de avaliação automática, e possibilitar esclarecer critérios de avaliações a partir de exemplos corrigidos dinamicamente.

Referências

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

Kotsiantis, S. (2007). Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24. IOS Press.

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.

Mertler, C. A. (2001). Designing scoring rubrics for your classroom. *Practical Assessment, Research & Evaluation*, 7(25):1–10.

Naude, K. A., Greyling, J. H., and Vogts, D. (2010). Marking student programs using graph similarity. *Computers & Education*, 54(2):545 – 561.

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., 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.

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.