Informática Aplicada à Educação

Everson Mario Novak
(Organizador)

Editora Atena
Ano 2018

Everson Mario Novak
(Organizador)

Informática Aplicada à Educação

Atena Editora
2018

Editora Chefe: Profª Drª Antonella Carvalho de Oliveira

Edição de Arte e Capa: Geraldo Alves e Natalia Sandrini
Revisão: Os autores

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

I43 Informática aplicada à educação [recurso eletrônico] / Organizador Everson Mario Novak. – Ponta Grossa (PR): Atena Editora, 2018.
10.596 kbytes

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-85-85107-14-7
DOI 10.22533/at.ed.147181308

1. Educação. 2. Informática. 3. Tecnologia educacional. I. Novak, Everson Mario.

CDD 371.334

Elaborado por Maurício Amormino Júnior – CRB6/2422

www.atenaeditora.com.br
E-mail: contato@atenaeditora.com.br

APRESENTAÇÃO

Este livro foi divido em 3 eixos, fruto de pesquisa científica de ótima qualidade acadêmica sobretudo por equipes multidisciplinares e de diversas instituições. Os trabalhos realizados são para auxiliar na Educação a distância e presencial, utilizando recursos computacionais para o planejamento e desenvolvimento de aplicativos para apoiar o aprendizado de matemática e de atividades cotidianas para crianças autistas, desenvolvimento de jogos educacionais e ainda para avaliar os dados armazenados em LMS (Learning Management Software) da plataforma Moodle.

No primeiro eixo temos o desenvolvimento de softwares e aplicativos voltados para a EAD, iniciamos por uma aplicação m-learning Genius para o auxiliar no ensino de matemática na educação infantil, explorando formas geométricas, números e a adição e subtração através de figuras e sons. O ENEN foi tema de um aplicativo focado em preparar os alunos na disciplina de matemática. O relacionamento social, comunicação e alterações de comportamento do autista são o tema de estudo para o desenvolvimento de um aplicativo para auxiliar os autistas no aprendizado e no relacionamento social.

A Cloud Computing apoia a aprendizagem em ambientes U-learning para verificar os estilos de aprendizagem e aplicabilidade em ambientes educacionais. As métricas de software são utilizadas para fazer uma análise da aprendizagem em cursos de programação a distância. Uma base de conhecimento gerada das questões e códigos inseridos nas plataformas digitais de ensino, foi feita a classificação de códigos da linguagem C em medidas similares para fazer os agrupamentos para formação de uma base de questões com códigos e soluções associadas para correções de questões de forma automatizada.

O segundo eixo entra em jogos digitais e gamificação, auxiliam na aprendizagem de pessoas com deficiência visual, tenta garantir no processo pedagógico uma inclusão digital e social destas pessoas. O processo de aprendizado utilizou-se dos jogos construcionistas para propor quatro jogos educativos, simplificando a complexidade na sua criação. Problemas motivacionais dos alunos são tratados na gamificação para verificar o que ocorre em processos de aprendizagem em ambientes educacionais.

No terceiro e último eixo é abordada a aprendizagem de máquina (machin-learning), aplicada a educação e aprendizado. O conceito de Estilos de Aprendizagem (EA) da psicologia cognitiva e da pedagogia, são propostos em sistemas educacionais adaptativos, com algumas aplicações da Aprendizagem por Reforço, foi proposto uso de algoritmos relacionados a aprendizagem de máquina para obter os estilos de Aprendizagem. Aplicabilidade de modelos de Regressão Múltipla no contexto da EAD foi abordado para validar as variáveis de comportamento de autorregulação da aprendizagem na plataforma LMS – Moodle.

Ao escrever este prefácio contextualizei o alinhamento das análises e teorias desenvolvidas nos artigos contidos neste livro. Sugiro que o leitor faça este caminho para uma compreensão ampla destes trabalhos, agradeço a oportunidade de fazer parte de grupo e felicito a todos os integrantes.

Everson Mario Novak
Mestrando em Informática - PUCPR

SUMÁRIO

EIXO 1: SOFTWARES E APLICATIVOS VOLTADOS PARA A EAD

CAPÍTULO 1 1

GENIUS MATH: UMA APLICAÇÃO MOBILE PARA AUXILIAR A APRENDIZAGEM DA MATEMÁTICA NA PRÉ-ESCOLA

Stefane Vieira Menezes
Jiani Cardoso da Roza

CAPÍTULO 2 13

APLICATIVO MÓVEL PARA PREPARAÇÃO DE ESTUDANTES PARA O ENEM NO CONTEXTO DA DISCIPLINA DE MATEMÁTICA

Hannderson Faria Arantes
Rodrigo Duarte Seabra

CAPÍTULO 3 27

COTIDIANO: UM SOFTWARE PARA AUXILIAR CRIANÇAS AUTISTAS EM SUAS ATIVIDADES DIÁRIAS

Afranio Furtado de Oliveira Neto
Hugo Leonardo Pereira Rufino
Diovane de Godoi Beira
Rodolfo Bocado Palis
Paula Teixeira Nakamoto

CAPÍTULO 4 41

APRENDIZAGEM SIMULADA NA NUVEM

Rafaela R. Jardim
Roseclea Duarte Medina
Giliane Bernardi
Fabricio Herpich
Andressa Facalde
Eduardo Lemos

CAPÍTULO 5 55

ANÁLISE DA APRENDIZAGEM DE PROGRAMAÇÃO POR MAPEAMENTO DE PERFIS EM MÉTRICAS DE SOFTWARE

Márcia Gonçalves de Oliveira
Ádler Oliveira Silva Neves
Helen França Medeiros
Mônica Ferreira Silva Lopes
Leonardo Leal Reblin
Elias Silva de Oliveira

CAPÍTULO 6 68

CLASSIFICAÇÃO DE CÓDIGOS C USANDO MEDIDAS DE SIMILARIDADE PARA APOIO AO ENSINO DE PROGRAMAÇÃO

José Carlos Campana Filho
Elias Silva de Oliveira
Márcia Gonçalves de Oliveira

EIXO 2: JOGOS DIGITAIS E GAMIFICAÇÃO

CAPÍTULO 7 79
BEM EXPRESSÕES: JOGO DIGITAL VOLTADO PARA O ENSINO INCLUSIVO DA MATEMÁTICA

André Luis Bitencourt Fernandes
Claudia Pinto Pereira
Kayo Costa de Santana
Ana Jaize de Oliveira Silva Santos
Bruno Gonzaga de Mattos Vogel

CAPÍTULO 8 95
JINDIE: UMA LINHA DE PRODUTO DE SOFTWARE PARA JOGOS EDUCATIVOS COM FOCO NO CONSTRUCIONISMO

Carlos Alberto Correia Lessa Filho
Arturo Hernandez Dominguez

CAPÍTULO 9 107
METODOLOGIAS GAMIFICADAS PARA A EDUCAÇÃO: UMA REVISÃO SISTEMÁTICA DETECÇÃO AUTOMÁTICA DE ESTILOS DE APRENDIZAGEM: UMA ANÁLISE COMPARATIVA DE CLASSIFICADORES APLICADOS EM UM CENÁRIO REAL DE APRENDIZADO

André Luiz de Souza Brito
Charles Andryê Galvão Madeira

EIXO 3: APRENDIZAGEM DE MÁQUINA APLICADA A EDUCAÇÃO

CAPÍTULO 10 120
DETECÇÃO AUTOMÁTICA DE ESTILOS DE APRENDIZAGEM: UMA ANÁLISE COMPARATIVA DE CLASSIFICADORES APLICADOS EM UM CENÁRIO REAL DE APRENDIZADO

Lucas Daniel Ferreira
José Fernando Rodrigues Jr

CAPÍTULO 11 140
DETECÇÃO AUTOMÁTICA E DINÂMICA DE ESTILOS DE APRENDIZAGEM EM SISTEMAS ADAPTATIVOS E INTELIGENTES PARA A EDUCAÇÃO UTILIZANDO DYNAMIC SCRIPTING

Júlio César da Costa Silva
Cristiano Grijó Pitangui
Alessandro Vivas Andrade
Luciana Pereira de Assis
Cristiano Maciel da Silva

CAPÍTULO 12 156
UM PROCESSO DE VALIDAÇÃO DE VARIÁVEIS COMPORTAMENTAIS DE AUTORREGULAÇÃO DA APRENDIZAGEM EM PLATAFORMAS DE LMS

Rodrigo Lins Rodrigues
João Carlos Sedraz Silva
Jorge Luis Cavalcanti Ramos
Fernando da Fonseca de Souza
Alex Sandro Gomes

SOBRE O ORGANIZADOR 166

EIXO 1 – SOFTWARES E APLICATIVOS VOLTADOS PARA A EAD

APRESENTAÇÃO

No primeiro eixo temos o desenvolvimento de softwares e aplicativos voltado para EAD, iniciamos por uma aplicação m-learning Genius para o auxiliar no ensino de matemática na educação infantil, explorando formas geométricas, números e a adição e subtração através de figuras e sons. Com atividades lúdicas viabilizando práticas contemporâneas ao cotidiano infantil.

Agora abordando outro tema pertinente o ENEN, um aplicativo focado em preparar os alunos para o Exame Nacional do Ensino Médio na disciplina de matemática.

As dificuldades apresentadas em relacionamento social, comunicação e alterações de comportamento por um autista são o tema de estudo para o desenvolvimento de um aplicativo para auxiliar os autistas no aprendizado e no relacionamento social.

A Cloud Computing está apoiando a aprendizagem em ambientes U-learning, criando um laboratório virtual U-Lab Cloud para verificar os estilos de aprendizagem para adotar a tecnologia em ambientes educacionais.

O software PCodigo II, utiliza métricas de software para fazer a análise da aprendizagem em cursos de programação a distância, para que sejam observadas dificuldades de aprendizagem, boas práticas de programação e perfis de aprendizagem de forma rápida, detalhada e holística.

Neste outro tema é gerado uma base de conhecimento de forma organizada das questões e códigos gerados nas plataformas digitais de ensino a distância. Abordando uma classificação de códigos da linguagem C baseada em medidas similares para fazer os agrupamentos para formação de uma base de questões com códigos e soluções associadas para correções de questões de forma automatizada.

Everson Mario Novak
Mestrando em Informática - PUCPR

CAPÍTULO 6

CLASSIFICAÇÃO DE CÓDIGOS C USANDO MEDIDAS DE SIMILARIDADE PARA APOIO AO ENSINO DE PROGRAMAÇÃO

José Carlos Campana Filho
Universidade Federal do Espírito Santo (Ufes)
Vitória - ES

Elias Silva de Oliveira
Universidade Federal do Espírito Santo (Ufes)
Vitória - ES

Márcia Gonçalves de Oliveira
Instituto Federal do Espírito Santo (Ifes)
Vitória – ES

RESUMO O aumento de cursos de programação de computadores em plataformas de ensino a distância tem gerado uma grande quantidade de questões e códigos. No entanto, essa base de conhecimento nem sempre está organizada de forma adequada para ser reaproveitada. Com o objetivo de auxiliar professores na geração dessa base, propomos uma melhoria em uma abordagem de classificação de códigos em Linguagem C baseada em medidas de similaridade para agrupar códigos de programação por tema ou por tipo de problema. A contribuição desse processo de classificação é a geração de uma base de questões com códigos de soluções associados, que pode ser utilizada como fonte de pesquisa ou para correção automática de questões de programação.
PALAVRAS-CHAVE classificação automática, medidas de similaridade, programação.

ABSTRACT The increase in computer programming courses in distance learning platforms has generated a lot of questions and codes. This knowledge base is not always organized in a suitable form to be reused. In order to assist teachers in the generation of the database by grouping programming codes by theme, or kind, of the problem, we propose an improvement in the classification of codes approach for the C language based on similarity measures. The contribution of this classification process is the generation of a base of questions associated to the code solutions that can be used as a source for research and for automatic assessment of program issues.
KEYWORDS automatic classification, similarity measures, programming.

1 | INTRODUÇÃO

A motivação deste artigo surgiu a partir de um problema que professores do Programa de Educação Tutorial (PET) da Universidade Federal do Espírito Santo (Ufes) enfrentaram (VALENTIM *et al.* 2014).

O PET/Ufes possui um histórico de atividades de programação cujas soluções foram desenvolvidas em programas escritos em Linguagem C e submetidos por seus alunos

na plataforma *Moodle*. Todavia, os enunciados das atividades foram desvinculados de seus códigos ao longo do tempo. Dessa forma, uma vez que esses códigos ficaram sem associações com seus enunciados, neste trabalho, temos como objetivo reconstruir essas associações.

Nossa proposta de solução é, portanto, a utilização de um processo de classificação desses códigos em que os enunciados passam a representar as classes almejadas. Essa estratégia visa apoiar essa tarefa de reconstrução de uma base de dados de cerca de 100 atividades e mais de 3000 códigos que foram submetidos ao longo do programa do PET/Ufes.

Essa base de dados gerada pode ser usada como gabarito de questões, servindo de referência para professores em uma correção manual, ou ser usada como conjunto de treino para tecnologias desenvolvidas que realizam correção automática de questões de programação (MOREIRA E FAVERO, 2009) e para sistema de apoio à prática assistida de programação por execução em massa e análise de programas (OLIVEIRA *et al*. 2015)

Para classificarmos os códigos, usamos como solução a abordagem de classificação proposta por Baby *et al*. (2014), que classifica códigos escritos em Linguagem C que tratam a mesma classe de atividade, isto é, agrupa códigos que tratam um mesmo tema (estruturas de árvore, fila, matriz e outros temas).

Assume-se, dessa forma, que os códigos podem ser classificados de acordo com a ocorrência de determinados termos, sejam eles identificadores, palavras-chave, operadores ou símbolos. Os códigos são decompostos em vetores de termos e a classificação é baseada no cálculo de similaridades dos vetores dado por um conjunto de 36 métricas de distância.

Como contribuição de melhoria no processo proposto por Baby et al. (2014), foi realizada uma análise sintática e semântica (CHA, 2007) do conjunto das 36 medidas de similaridade, visando reduzir esse conjunto e melhorar o desempenho do processamento.

Este trabalho está organizado conforme a ordem a seguir. Na Seção 2, apresentamos os trabalhos relacionados a nossa proposta. Na Seção 3, explicamos a metodologia utilizada para classificar os códigos. Na Seção 4, apresentamos a análise feita nas métricas de similaridade em busca de melhoria de desempenho. Na Seção 5, relatamos os experimentos. Na Seção 6, concluímos com as considerações finais e trabalhos futuros.

2 | TRABALHOS RELACIONADOS

Muitos trabalhos já foram realizados em relação à similaridade de códigos de programação, seja para detecção de plágio ou mesmo para classificação. Muitas

ferramentas estão disponíveis para realizar a análise de similaridade entre códigos, entre as quais destacamos o *JPlag* (PRECHELT *et al.,* 2002), *MOSS* (AIKEN, 2014) e *Sherlock* (PIKE, 2012). A proposta desenvolvida neste artigo se difere desses trabalhos, principalmente no tocante ao cálculo de distância entre os arquivos para determinar a similaridade. Enquanto nos trabalhos relacionados o grau de similaridade é dado pelo cálculo de distância de uma medida (*cosseno* ou euclidiana, normalmente), neste artigo ele é dado pelo cálculo de distâncias de um conjunto de medidas.

O *JPlag* é uma ferramenta que reconhece semelhanças entre conjuntos de arquivos de códigos-fontes. O *JPlag* considera a sintaxe da linguagem de programação e a estrutura de programas ao compará-los, o que é bem adequado para detectar tentativas de disfarces de semelhanças entre os arquivos plagiados. O *JPlag* suporta códigos em Java, C#, C, C++, *Scheme* e texto em linguagem natural. O *JPlag* já desempenhou importante papel em casos de propriedade intelectual, onde ele tem sido usado com sucesso por peritos.

O Measure of Software Similarity (MOSS) é um sistema automático para a análise de similaridade de programas. A principal aplicação do MOSS foi na detecção de plágios realizados em aulas de programação e, desde o seu desenvolvimento em 1994, o MOSS tem sido muito eficaz nesse papel. O MOSS é capaz de analisar similaridades entre códigos escritos em várias linguagens de programação, dentre as quais destacamos C, C++, Java, C#, Python, Visual Basic, Javascript, Fortran, Haskell, Lisp, Pascal, Ada e Perl.

O *Sherlock, por sua vez,* é um programa que verifica similaridades entre documentos de textos. Ele gera uma assinatura digital (*hash*) para uma palavra ou sequência de palavras e compara para detectar semelhanças.

A assinatura digital é um número que representa um conjunto de palavras convertidas em uma série de bits. O *Sherlock* é implementado em C e possui código aberto, podendo ser aprimorado pela comunidade. Como essa ferramenta compara códigos como documentos de texto, é interessante que se faça uma preparação do arquivo do código antes de submetê-lo ao *Sherlock*.

Um trabalho muito interessante é o de Baby *et al*. (2014), que trata a análise de similaridade um pouco diferente ao determinar a similaridade entre arquivos por um conjunto de medidas de distância. Por ser um tópico no estudo de similaridade ainda não muito explorado, o nosso trabalho baseou-se nessa proposta, que será apresentada em mais detalhes na próxima seção.

Considerando a proposta de Baby *et al*. (2014), visando uma performance melhor, usamos os conceitos de Cha (2007) para reduzir o conjunto de medidas de similaridade utilizadas. Na proposta de Cha (2007) é realizada uma análise sintática e semântica de várias medidas de similaridade e elas são agrupadas conforme algumas características. A melhoria nesse processo irá, consequentemente, contribuir para a geração de uma base de enunciados e códigos de programação.

3 I METODOLOGIA

Nesta seção descrevemos o método usado neste artigo, que é baseado no método proposto por Baby *et al*. (2014). Na Figura 1, é ilustrado o processo desse método, onde os dados são primeiramente divididos em treino e teste e o modelo é então avaliado comparando as predições com o conjunto de teste.

De acordo com a Figura 1, uma lista de *tokens* é criada contendo as funções nativas e operadores usados na Linguagem C, sendo que cada *token* é associado a um código inteiro único.

Na conversão, o código em Linguagem C é transformado em um vetor de termos, onde cada dimensão é a frequência de ocorrência de um termo, que pode ser uma palavra-chave, um operador, um literal, uma constante ou um outro símbolo. Assim, cada termo é convertido no valor inteiro equivalente e representa uma posição no vetor de frequência. O valor na posição referente ao termo é, portanto, a sua frequência de ocorrência em um código-fonte.

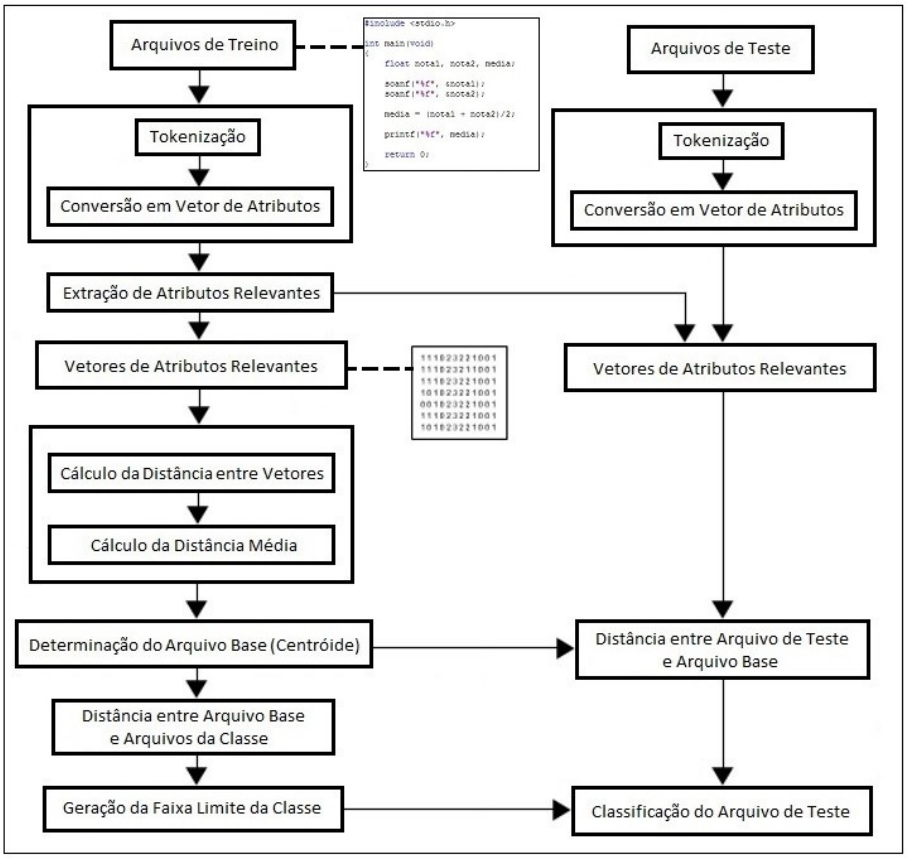

Figura 1. O método proposto

A Tabela 1 mostra um exemplo simples de conversão de código C em um vetor *V* de frequência dos termos. Nesse exemplo, os valores do vetor *V* correspondem à

frequência dos termos *int*, *main*, "{", *literal (x e y)*, "=", *constante (10 e 0)*, ";", *return* e "} ", respectivamente. A similaridade entre dois códigos é obtida através do cálculo da distância entre dois vetores, onde P_i é o *i-ésimo* termo do primeiro vetor e Q_i é o *i-ésimo* termo do segundo vetor.

Nem todo termo é necessário para representar as características de uma classe. Somente os termos presentes em pelo menos 40% dos arquivos da classe são considerados relevantes para representar a classe, os outros são descartados. Em uma classe que contém 26 programas, por exemplo, termos presentes em mais de 40% dos 26 arquivos (ou seja, presentes em mais de dez arquivos) são selecionados como atributos relevantes.

A distância entre os vetores que representam os arquivos de uma classe é calculada para encontrar a similaridade entre esses arquivos. Dessa forma, quanto menor a distância, mais similares são dois arquivos.

Para verificar a similaridade entre os arquivos, utilizamos 36 medidas de distância, que são apresentadas na Tabela 2.

Para cada medida, é gerada uma matriz de distâncias *n x n,* onde *n* é o número de programas reunidos em uma classe. Em seguida, é gerada outra matriz *n x 36* com as médias de distância para cada arquivo e medida.

A partir dessa matriz, é selecionado um arquivo-base, que é o centróide da classe, para cada medida. O centróide é o arquivo com a menor distância em determinada medida. O centróide pode ser ou não o mesmo arquivo para cada uma das 36 medidas de distância.

Para gerar a faixa limite da classe para cada medida de distância, é calculada a distância entre os arquivos-base de cada medida de distância e os arquivos similares. Se o número de arquivos que possuem valor de distância menor que a distância média é maior que o número de arquivos com distância maior que a distância média, então a faixa limite é a menor distância até a média, caso contrário, a faixa é a distância média até a maior distância.

Código	*Tokens*	Posição do token no vetor	Vetor de frequências

int main {
int x = 10;
int y = x;
return 0;
}
 	
	int	1	*v* = {3,1,1,3,2,2,3,1,1}

	main	2	
	{	3	
	literal (x e y)	4	
	=	5	
	constante (10 e 0)	6	
	;	7	
	return	8	
	}	9	

Tabela 1. Exemplo de conversão de código em vetor de frequências de termos

Para classificar arquivos de teste, é avaliada a distância entre o centróide e o

teste, e se o valor está dentro da faixa limite em mais de 50% das medidas, o arquivo é dito como pertencente à classe.

Additive Symmetric	*Average*	*Bhattacharrya*	*Canberra*
Chebyshev	*City Block*	*Clark*	*Czekanowski*
Dice	*Divergence*	*Euclidean*	*Gower*
Harmonic Mean	*Hellinger*	*Intersection*	*Jeffreys*
Jensen Difference	*Jensen Shannon*	*Kdivergence*	*Kulczynski*
Kullbackliebler	*Kumar Johnson*	*Lorentzian*	*Matusita*
Neyman	*Probabilistic Symmetric*	*Ruzicka*	*Soergel*
Sorensen	*Squared*	*Squared Euclidean*	*Squared Chord*
Taneja	*Tanimoto*	*Topsoe*	*Wave Hedges*

Tabela 2. Conjunto das 36 medidas de similaridade da abordagem original

4 I ANÁLISE DO CONJUNTO DE MÉTRICAS

Para tentar melhorar o desempenho na classificação, focamos em alterar o conjunto de medidas propostas por Baby et al. (2014). Para as fórmulas apresentadas nesta seção, considere d o número de termos do vetor, P_i *o i-ésimo* termo do primeiro vetor e Q_i *o i-ésimo* termo do segundo vetor.

Observando as 36 métricas utilizadas, e considerando as análises semânticas e sintáticas de Cha (2007), pode-se perceber que algumas métricas são equivalentes, ou seja, produzem resultados idênticos, como Soergel e Tanimoto, nas equações (1) e (2).

$$d_{Kdivergence} = \frac{\sum_{i=1}^{d} \left| P_i - Q_i \right|}{\sum_{i=1}^{d} \max\left(P_i, Q_i\right)} \quad (1)$$

$$d_{Topsoe} = \frac{\sum_{i=1}^{d} \max\left(P_i, Q_i\right) - \sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} \max(P_i, Q_i)} \quad (2)$$

Outras métricas são proporcionais, como *City Block* e *Gower*, em (3) e (4).

$$d_{CityBlock} = \sum_{i=1}^{d} \left| P_i - Q_i \right| \quad (3)$$

$$d_{Gower} = \frac{1}{d}\sum_{i=1}^{d}\left|P_i - Q_i\right|,\ d_{Gower} = \frac{1}{d}d_{CityBlock} \quad (4)$$

Ainda temos métricas que são assimétricas, ou seja, a distância entre *P* e *Q*, *d*(*P*,*Q*) é diferente de *d*(*Q*,*P*), como a métrica *Kdivergence*, na equação (5). A métrica *Topsoe* é uma versão simétrica da *Kdivergence*, na equação (6).

$$d_{Kdivergence} = \sum_{i=1}^{d} P_i \ln \frac{2P_i}{P_i + Q_i} \quad (5)$$

$$d_{Topsoe} = \sum_{i=1}^{d}\left[P_i \ln\left(\frac{2P_i}{P_i + Q_i}\right) + Q_i \ln\left(\frac{2Q_i}{P_i + Q_i}\right)\right] \quad (6)$$

As métricas idênticas ou proporcionais não trazem novas informações sobre a similaridade de arquivos. Portanto, foram escolhidas apenas uma de cada métrica idêntica ou proporcional. As métricas assimétricas foram descartadas, mantendo apenas as suas versões simétricas.

Com isso o número de métricas passou de 36 para 26. Foi adicionada a este novo conjunto a métrica c*osseno*, pois é bastante mencionada na literatura (BAEZA e RIBEIRO, 1999) e não fazia parte do conjunto inicial. O novo conjunto passou a ter então 27 métricas, conforme a Tabela 3.

Additive Symmetric	*Average*	*Bhattacharrya*	*Canberra*
Chebyshev	*City Block*	*Clark*	-
Dice	*Divergence*	*Euclidean*	-
Harmonic Mean	-	*Intersection*	*Jeffreys*
Jensen Difference	*Jensen Shannon*	-	*Kulczynski*
-	*Kumar Johnson*	*Lorentzian*	*Matusita*
-	-	*Ruzicka*	-
-	*Squared*	*Squared Euclidean*	*Squared Chord*
Taneja	*Tanimoto*	-	*Wave Hedges*
Cosseno			

Tabela 3. Novo conjunto com 27 medidas de similaridade

5 I EXPERIMENTOS E RESULTADOS

Para o experimento deste trabalho, foram utilizadas duas bases de dados: a base utilizada no trabalho de Baby *et al*. (2014) e uma base de uma turma de programação formada por alunos do PET/UFES. O desempenho de classificação foi analisado com a métrica *Acurácia*.

Para a validação do modelo foi usada a técnica de validação cruzada com o método o *leave-one-out*. O método *leave-one-out* consiste em retirar um elemento da base para teste e utilizar o restante da base como conjunto de treino. Esse processo é repetido para todos os elementos. No final das iterações, calcula-se a *Acurácia* sobre as predições.

a)Base do artigo de referência

A base utilizada por Baby *et al*. (2014) está dividida em quatro classes: *Btree, Crque, Dqueue* e *Matrix*, que agrupam códigos referentes aos temas á*rvore B*, *fila circular*, *fila duplamente encadeada* e *matriz*, respectivamente. Cada uma dessas classes reúne 26 arquivos.

O processo de classificação foi realizado utilizando o conjunto inicial de medidas e, depois, o novo conjunto. A Tabela 4 mostra os resultados obtidos. As predições obtidas para as classes *Btree* e *Matrix* foram muito boas. Já as classes *Crque* e *Dqueue* não obtiveram a mesma qualidade de predição. Para todas as classes, alguns arquivos passaram a ser classificados na classe correta somente quando executado com as 27 medidas, o que sugere uma melhora na classificação.

Classe	Total de arquivos	Qtde acertos 36 medidas	Qtde acertos 27 medidas	% de acerto 36 medidas	% de acerto 27 medidas
Btree	26	22	23	84.61	88.46
Crque	26	11	11	42.30	42.30
Dqueue	26	11	14	42.30	53.84
Matrix	26	21	22	80.77	84.61
Total	**104**		**Média**	**62.50**	**67.30**

Tabela 4. Acurácia obtida com conjuntos de 36 e 27 medidas de similaridade para a base do artigo de referência

b) Base de Exercícios do PET/UFES

A base de exercícios do PET/UFES utilizada foi obtida de uma turma de alunos de Introdução à Programação em Linguagem C do projeto *Introcomp* (MENESES *et al*., 2015). Essa base possui treze classes, que são atividades, cada uma com quantidade de arquivos variados em um total de 591 arquivos.

A Tabela 5 mostra a quantidade de arquivos classificados corretamente para os dois conjuntos de medidas da base do PET/UFES.

Ao executar o processo de classificação, verificamos que alguns arquivos passaram a ser classificados na classe correta somente quando executado com as 27 medidas, como havia ocorrido na outra base.

Alguns arquivos foram classificados em mais de uma classe. Isso era esperado e se deve ao fato de algumas atividades abordarem conceitos parecidos de programação.

Nesse caso, alunos submeteram códigos de programas com conjunto de *tokens* parecidos. Situações como essa devem ser tratadas com a supervisão do professor para a indicar a correta classificação.

Classe	Total de arquivos	Qtde acertos 36 medidas	Qtde acertos 27 medidas	% de acerto 36 medidas	% de acerto 27 medidas
Questão 1	22	18	18	81.82	81.82
Questão 2	23	20	19	86.96	82.61
Questão 3	26	21	22	80.77	84.62
Questão 4	31	14	18	45.16	58.06
Questão 5	50	42	44	84.00	88.00
Questão 6	50	38	39	76.00	78.00
Questão 7	52	34	36	65.38	69.23
Questão 8	53	36	35	67.92	66.04
Questão 9	53	40	44	75.47	83.02
Questão 10	56	20	20	35.71	35.71
Questão 11	58	47	47	81.03	81.03
Questão 12	58	39	45	67.24	77.59
Questão 13	59	51	51	86.44	86.44
Total	**591**		**Média**	**71.84**	**74.78**

Tabela 5. Acurácia obtida com conjuntos de 36 e 27 medidas de similaridade para a base do PET/UFES

6 I CONSIDERAÇÕES FINAIS

Considerando o desempenho de classificação dos códigos C dos dois conjuntos de métricas, observamos que o novo conjunto obteve uma acurácia média de arquivos classificados corretamente melhor do que o primeiro, para as duas bases utilizadas, isto é, 67.30% contra 62.50% na primeira base, e 74.78% contra 71.84% na base do PET. Com isso, conclui-se que, com a retirada das medidas idênticas, proporcionais e assimétricas, e a inclusão da medida de similaridade *cosseno*, evidenciamos melhor a similaridade entre os códigos, e melhoramos o desempenho do processo.

Com esses resultados, o processo proposto se mostra capaz de solucionar o problema tratado nesse artigo, que é gerar uma base com uma grande quantidade de códigos associados a enunciados de questões e disponibilizar para professores do PET, ou para outros professores que tenham o mesmo problema. Essa base pode ser usada, por exemplo, como um repositório de pesquisa para alunos e professores ou como base de treino para a realização de correções automáticas.

Como trabalho futuro a partir deste, temos a análise de outras medidas e outros

pontos que podem ser alterados na abordagem inicial, como a *tokenização*, extração de atributos relevantes e a faixa de corte das classes. Também como trabalho futuro, pode-se aplicar essa abordagem de similaridade para detecção de possíveis plágios de códigos de programação.

REFERÊNCIAS

BABY, Julie et al. **Distance indices for the detection of similarity in C programs**. In: Computation of Power, Energy, Information and Communication (ICCPEIC), 2014 International Conference **on**. IEEE, 2014. p. 462-467.

BAEZA-YATES, Ricardo et al. **Modern information retrieval**. New York: ACM press, 1999.

CHA, Sung-Hyuk. **Comprehensive survey on distance/similarity measures between probability density functions**. City, v. 1, n. 2, p. 1, 2007.

MENESES, Leonardo F. et al**. IntroComp: Atraindo alunos do ensino médio para uma instigante experiência com a programação**. In: Anais do XXIII Workshop sobre Educação em Computação (WEI 2015), Recife, PE, 2015.

MOREIRA, Mireille Pinheiro; FAVERO, Eloi Luiz. **Um ambiente para ensino de programação com feedback automático de exercícios**. In: Workshop sobre Educação em Computação (WEI 2009). 2009.

AIKEN, Alex. **Moss (measure of software similarity) plagiarism detection system**. Disponível em: http://www. cs. berkeley. edu/moss**/**, 2000. Acesso em 16/04/2018.

OLIVEIRA, M.; NOGUEIRA, M.; OLIVEIRA, Elias. **Sistema de apoio à prática assistida de programação por execução em massa e análise de programas**. In: XIV Workshop de Educação em Computação (WEI-SBC), 2015.

PIKE, R. **The sherlock plagiarism detector**. **Disponıvel em:** http://sydney.edu.au/engineering/it/~scilect/sherlock/**.** Acesso em 16/04/2018, v. 21, 2012.

PRECHELT, Lutz; MALPOHL, Guido; PHILIPPSEN, Michael. **Finding plagiarisms among a set of programs with JPlag**. J. UCS, v. 8, n. 11, p. 1016, 2002.

VALENTIM, R. et al. **Em busca de uma metodologia para a disseminaçao em massa do ensino de programação**. Seminário Nacional de Inclusao Digital (SENID), Passo Fundo, RS, SBC, 2014.

EIXO 2 – JOGOS DIGITAIS E GAMIFICAÇÃO

APRESENTAÇÃO

O segundo eixo entra em jogos digitais e gamificação, no primeiro trabalho focado na aprendizagem de pessoas com deficiência visual, possuem uma diferente percepção devido às suas particularidades tenta garantir no processo pedagógico uma inclusão digital e social destas pessoas. No próximo artigo o tema de jogos construcionistas foi inserido no processo de aprendizagem e utilizado na construção de quatro jogos educativos, simplificando a complexidade e direcionando o processo de criação de jogos.

A abordagem gamificação ajuda a resolver problemas motivacionais dos alunos neste último artigo deste eixo, promovendo um engajamento no processo de aprendizagem, fazendo uma revisão sistemática da literatura para verificar o que ocorre em ambientes educacionais gamificados.

Everson Mario Novak

Mestrando em Informática - PUCPR

SOBRE O ORGANIZADOR

Everson Mario Novak Possui graduação em Tecnologia em Sistemas para Internet, Especialização em Desenvolvimento Web e MBA em Gestão de TI pela Faculdade Educacional de Ponta Grossa (Faculdade UNIÃO). Atualmente está cursando Mestrado em Informática na PUCPR - Pontifícia Universidade Católica do Paraná é professor do curso de Sistemas de Informação na Faculdades Integradas de Itararé – FAFIT. Ainda como Professor pela PUCPR na TECPUC na unidade de Ponta Grossa. É Analista de Sistemas, programador e tem experiência na área de Ciência da Computação, com ênfase em Arquitetura de Sistemas de Computação, Agentes de Software e Inteligência artificial.

Agência Brasileira do ISBN
ISBN 978-85-85107-14-7
9 788585 107147