Recomendação Automática de Exercícios de Programação Extraclasse em Ambientes de Correção Automática de Código

Coordenador/Proponente:	Prof. Dr. André Gustavo dos Santos
Orientadores:	Prof. Dr. Lucas Nascimento Ferreira
	Prof. Dr. Julio Cesar Soares dos Reis
Data:	24 de junho de 2024

Área de Tecnologia Prioritária: Tecnologias Habilitadoras / Inteligência Artificial
Justificativa: O projeto em questão visa explorar a aplicação de técnicas de mineração de dados e aprendizado de máquina com o intuito de investigar e propor abordagens automáticas para geração e recomendação de exercícios de programação extraclasse em ambientes de correção automática de código.

Departamento de Informática
Centro de Ciências Exatas e Tecnológicas
Universidade Federal de Viçosa

Resumo

As disciplinas de programação de computadores dos cursos de Ciência da Computação costumam ser grandes e compostas por alunos com alta variância em experiência de programação. Como essas disciplinas demandam uma grande quantidade de exercícios práticos, um dos principais desafios para professores é desenvolver listas de exercícios adequados para os diversos perfis de alunos. Esse projeto propõe o desenvolvimento de um sistema de recomendação para exercícios de programação extraclasse, visando possibilitar experiências de aprendizagem personalizadas para cada perfil de aluno. A hipótese central desse projeto é de que experiências de aprendizagem personalizadas por sistemas de recomendação são capazes de suavizar a trajetória acadêmica de alunos de disciplinas de programação, melhorando o desempenho dos alunos e formando melhores programadores. Para testar essa hipótese, serão exploradas diferentes técnicas de recomendação de exercícios, inicialmente considerando uma abordagem de filtragem baseada em conteúdo. Esses sistemas serão avaliados por estudantes do curso de Ciência da Computação da Universidade Federal de Viçosa, que serão recrutados para utilizar as diferentes versões do sistema em um contexto simulado de disciplina de programação. Espera-se que os alunos que usarem os sistemas de recomendação tenham uma experiência de aprendizagem mais adequada ao seu conhecimento de programação, do que os alunos que não usarem as recomendações.

Palavras-chave: recomendação, exercícios, programação.

1 Introdução

As disciplinas de programação de computadores são fundamentais para os cursos de Ciência da Computação, pois, além de serem fortes pré-requisitos para a maioria das outras disciplinas do curso, também são primordiais para a o mercado de Tecnologia da Informação. No entanto, elas costumam ser desafiadoras para os estudantes ingressantes, que geralmente possuem um conhecimento bastante heterogêneo de programação. Enquanto alguns não possuem experiência alguma, vários possuem conhecimentos básicos e até mesmo avançados, adquiridos por auto-aprendizado ou cursos de programação especializados. Isso exige dos professores uma atenção diferenciada para cada aluno, principalmente nas aulas práticas.

Segundo (MEDEIROS; RAMALHO; FALCÃO, 2018) e (BECKER; QUILLE, 2019), aprender programação exige muita prática, especialmente por meio de exercícios sobre os mais diversos tópicos da disciplina. Sendo assim, os professores de programação costumam usar Ambientes de Correção Automática de Código (ACACs), tanto para a criação de listas extraclasse, quanto para trabalhos práticos ou exames presenciais (LARANJEIRA, 2020). Nesses ambientes, os alunos recebem uma lista de problemas, onde são especificados os formatos de entrada e saída esperados para cada problema. Quando um aluno submete uma solução seguindo corretamente a especificação, o ACAC é capaz de fazer a correção automática dessa submissão. Esse alívio de carga com a correção permite que os professores criem mais exercícios para os alunos praticarem os diversos tópicos da disciplina.

Mesmo com o uso de ACACs, um dos principais desafios para os professores é criar listas de exercícios extraclasse adequadas para os diferentes perfis de alunos. Uma lista adequada desenvolve um determinado conjunto de tópicos em ordem crescente de dificuldade, de acordo com os conhecimentos de programação de cada aluno, não desmotivando o estudante menos experiente, nem entediando o estudante mais experiente. Ademais, selecionar exercícios manualmente para os variados perfis de alunos é uma solução inviável, uma vez que, além de diversas, as disciplinas de programação costumam ter uma grande quantidade de alunos (BEZ; FERREIRA; TONIN, 2013; PEREIRA et al., 2020).

Nesse contexto, a comunidade científica brasileira de Educação em Computação vêm propondo o uso de sistemas de recomendação ou classificação automática de exercícios de programação. Por exemplo, (LARANJEIRA, 2020) propõem um método de recomendação que filtra os exercícios por nível de dificuldade usando a técnica de recomendação de filtragem colaborativa. Os autores em (JÚNIOR et al., 2020) aplicam técnicas de processamento de linguagem natural para analisar o enunciado de um exercício anteriormente realizado pelo estudante e propor um novo problema com enunciado similar. Já em (LIMA et al., 2021), é proposto um modelo para classificação binária de exercícios de programação utilizando atributos do código fonte de uma dada solução, classificando-os

como “Fácil” e “Não Fácil”. Todos esses trabalhos foram desenvolvidos no contexto do ACAC CodeBench, criado pela Universidade Federal do Amazonas (UFAM).

Em suma, esses métodos mostraram evidências de que sistemas de recomendação podem aumentar a taxa de acerto e reduzir a taxa de erros e desistência dos alunos (JÚNIOR et al., 2020). No entanto, eles ainda possuem uma série de limitações, como a necessidade de um problema-alvo para se fazer uma recomendação, de um ACAC específico para coleta de dados de desempenho dos alunos, ou de um grande número de submissões prévias de um determinado aluno. Além disso, esses métodos foram avaliados considerando apenas uma única lista de exercícios, e não uma sequência de listas dentro de um contexto contínuo de aprendizagem (e.g., uma disciplina). Logo, este projeto tem como objetivo desenvolver sistemas de recomendação automática de exercícios extraclasse no contexto de disciplinas de programação, independente de ACAC, visando proporcionar uma experiência de aprendizagem personalizada para cada perfil de aluno. A hipótese central baseia-se na ideia de que experiências personalizadas de aprendizagem podem suavizar a trajetória acadêmica do aluno em disciplinas de programação, aumentando o desempenho individual dos alunos e formando programadores mais capacitados.

Para lidar com as limitações dos sistemas de recomendação atuais, propomos modelar o problema utilizando filtragem baseada em conteúdo. Nessa abordagem, os exercícios são representados como vetores de características e os perfis dos alunos são estimados a partir de métricas definidas implícita (e.g., número de submissões) ou explicitamente (e.g., uma nota de dificuldade) pelo aluno. O objetivo do sistema de recomendação é calcular a similaridade entre o perfil do aluno e os exercícios não resolvidos. Dessa forma, o problema de recomendação pode ser dividido em quatro sub-problemas: (1) definir e extrair características dos exercícios, (2) definir métricas de avaliação de exercícios, (3) estimar o perfil dos alunos e (4) definir uma métrica de similaridade entre exercícios e perfis de alunos. Esse sub-problemas fundamentam as questões de pesquisa (QP) desse projeto, que são descritas a seguir:

- **QP 1**: Como extrair características de um exercício que definam a sua dificuldade?
- **QP 2**: Qual métrica será utilizada pelos alunos para avaliar os exercícios?
- **QP 3**: Como estimar o perfil dos alunos?
- **QP 4**: Qual métrica de similaridade é mais adequada para recomendar exercícios de programação?

Para responder essas perguntas e testar a hipótese definida, planeja-se realizar experimentos (teste com usuários) com alunos das disciplinas de programação do curso de Ciência da Computação da Universidade Federal de Viçosa (UFV). Para isso, dividiremos cada turma em dois grupos: A e B, onde o grupo A utilizará um sistema de recomendação de exercícios e o grupo B não (i.e, grupo controle). Ao final de cada atividade

avaliativa, aplicaremos um questionário para avaliar a experiência de aprendizagem de cada aluno. Além disso, mediremos o desempenho médio de cada grupo. Esperamos que os grupos que usarem o sistema de recomendação tenham uma experiência de aprendizado significativamente mais suave, com um desempenho superior aos grupos de controle.

2 Trabalhos Relacionados

Essa pesquisa está diretamente relacionada com ACACs e sistemas de recomendação e classificação de exercícios de programação no contexto de Educação em Computação. Essa seção apresenta uma breve revisão da literatura dessas áreas.

2.1 Ambientes de Correção Automática de Código

Os ACACs fazem correção automática de códigos submetidos para solução de questões previamente cadastradas. Eles podem ser classificados em dois tipos: juízes *online* de propósito geral e juízes dedicados. Os juízes *online* de propósito geral possuem milhares de questões cadastradas, de diferentes níveis e origens, que podem ser usadas pelos próprios alunos como ambiente de treinamento extraclasse. Exemplos desse tipo de ACAC incluem o Beecrowd[1] (ex URI online), o Online Judge[2] (ex UVa online judge), o CodeForces[3], o SPOJ[4] e o Neps Academy[5]. Os juízes dedicados podem ser instalados e gerenciados localmente pelos professores, com fácil cadastro de questões próprias e personalizadas para a disciplina ministrada. Exemplos desse tipo de ACAC incluem o BOCA (CAMPOS; FERREIRA, 2004) e o CodeBench[6] (GALVÃO; FERNANDES; GADELHA, 2016).

Os ACACs vêm sendo largamente utilizados nas disciplinas de programação favorecendo assim a escalabilidade do trabalho dos professores (BEZ; FERREIRA; TONIN, 2013; GALVÃO; FERNANDES; GADELHA, 2016; FRANCISCO et al., 2018; PEREIRA et al., 2020). No curso de Ciência da Computação da UFV, ACACs são utilizados na disciplina INF110 (Programação I) desde 2014, contribuindo bastante nas aulas práticas da disciplina, pois os alunos com conhecimento prévio ou que aprendem mais rapidamente, conseguem terminar os exercícios e saber que estão corretos sem (ou com pouca) necessidade de auxílio do professor, o que permite ao professor prestar um atendimento mais direcionado e por mais tempo aos alunos com mais dificuldade. Além dos exercícios propostos nas aulas práticas, os alunos têm uma série de exercícios cadastrados para treinamento extraclasse.

[1] https://www.beecrowd.com.br
[2] https://onlinejudge.org
[3] https://codeforces.com
[4] https://www.spoj.com
[5] https://neps.academy/
[6] https://codebench.icomp.ufam.edu.br

Vale ressaltar que os ACACs não sugerem exercícios de acordo com as necessidades de aprendizagem do estudante nos tópicos das disciplinas de programação de computadores. Eles apenas corrigem automaticamente os exercícios cadastrados pelos professores. Aos alunos é apresentada uma lista de exercícios, numa ordem pré-definida pelo professor, e cabe aos alunos (ou ao professor) escolher a ordem em que são feitos. Em se tratando de diferentes perfis de alunos, é essencial que seja incluído um sistema automático de recomendação de questões.

2.2 Modelos de Classificação de Exercícios de Programação

(LIMA et al., 2021) propõem um modelo automático para classificação de exercícios de programação de computadores que usa alguns atributos do código fonte da solução configurada para um exercício em um ACAC, como por exemplo, tamanho do código e complexidade ciclomática total do código, classificando-os como "Fácil"e "Não Fácil". A partir dessa classificação, o próprio estudante pode selecionar qual exercício deseja resolver de acordo com a sua classificação. Porém, um aluno pode ficar sempre escolhendo os exercícios "fáceis"para resolver, sem saber quais dos exercícios "não fáceis"já tem condições de resolver com o conhecimento adquirido dos exercícios já resolvidos.

Uma outra abordagem proposta por (SANTOS et al., 2019) para classificação de exercícios de programação explora, especificamente, os enunciados associados. Através do uso de técnicas de processamento de linguagem natural foram extraídos atributos de legibilidade do texto dos enunciados, os quais foram correlacionados com a dificuldade do exercício medida através das taxas de acerto de cada exercício coletadas no ACAC. Os resultados apresentam evidências de que exercícios com enunciados complexos e de difícil leitura normalmente possuem baixas taxas de acerto. Por outro lado, exercícios com enunciado simples e de fácil leitura nem sempre resultam em altas taxas de acerto. Segundo (SANTOS et al., 2019), isso é explicado pelo fato que alguns enunciados podem trazer informações que não contribuem para a resolução do exercício, apesar de serem de fácil legibilidade.

(NEVES et al., 2017) propõem uma ferramenta que mapeia a solução de exercícios de programação em perfis de aprendizagem, os quais são representados por 348 métricas de *software* caracterizando qualidade e esforço de programação. Essas métricas quantificam informações, como por exemplo, variabilidade de palavras reservadas, se o código compila e executa, esforço, dificuldade de programação, uso de funções, linhas de código e média de complexidade ciclomática de funções. A partir dos valores dessas métricas, é gerado um vetor multidimensional representando um perfil de aluno. A classificação do exercício é feita a partir dos perfis dos alunos que o resolveram. Porém, para o mapeamento do perfil de aprendizagem ocorrer satisfatoriamente, é necessário que as questões tenham sido resolvidas por um número razoável de alunos.

2.3 Sistemas de Recomendação de Exercícios

(JÚNIOR et al., 2020) propõem e avaliam métodos para recomendação automática de problemas em ACACs, em que as recomendações são realizadas a partir de um exercício anteriormente realizado pelo aluno. As recomendações sugeridas foram avaliadas por alunos e professores utilizando uma abordagem duplamente cega, sendo evidenciada a adequação das recomendações de acordo com o sucesso obtido (maior taxa de acerto e menor taxa de erro e desistência). Um ponto de atenção na abordagem proposta por esse trabalho é que toda a recomendação realizada é baseada na análise do enunciado (chamado de Método Baseado no Enunciado (MBE) usando técnicas de processamento de linguagem natural) e no comportamento (chamado de Método Baseado no Comportamento (MBC) do aluno no uso do ambiente integrado de desenvolvimento (IDE) para resolver o exercício) de um único exercício, chamado de problema-alvo. Essas limitações são destacadas pelos próprios autores no referido trabalho, que sugerem, como direções para trabalhos futuros, a proposição de um método híbrido no contexto supracitado (i.e., entre MBE e MBC).

Já o estudo apresentado em (LARANJEIRA, 2020) propõe um método de recomendação de exercícios de programação de computadores que filtra os exercícios por nível de dificuldade usando a técnica de recomendação de filtragem colaborativa, que mapeia as dificuldades enfrentadas pelos alunos quando resolvem os exercícios de programação no ACAC CodeBench. Posteriormente, é realizada a predição da dificuldade dos exercícios ainda não resolvidos pelo aluno, sendo assim possível sugerir exercícios com graus de dificuldade crescente. Duas limitações do trabalho apontadas pelo próprio autor são que, caso o aluno tenha resolvido poucos exercícios, a qualidade da recomendação do método proposto não será boa, o que é o caso dos alunos das disciplinas introdutórias de programação que estão iniciando o curso superior de computação. Além disso, o método não leva em consideração alguns fatores de contexto pedagógicos do aluno, como por exemplo, ambiente da sala de aula, o professor da disciplina, a base de conhecimento prévio do aluno, dentre outros. Para minimizar essas limitações é sugerido por (LARANJEIRA, 2020) como trabalhos futuros aplicar algoritmos de aprendizagem de máquina para determinar os atributos mais relevantes para cada perfil de aluno em tempo real, usando, por exemplo, retorno (*likes*, avaliação por estrelas, dentre outros) da recomendação recebida pelo aluno. Além disso, seria importante traçar perfis também do exercício, professor da disciplina e instituição de ensino, os quais podem influenciar na recomendação gerada para cada perfil de aluno. Uma outra sugestão de trabalho futuro seria a aplicação do método em uma quantidade maior de alunos, (LARANJEIRA, 2020) testou seu método usando 180 (cento e oitenta) alunos, e em uma turma em andamento.

3 Metodologia de Pesquisa

Esse projeto propõe inicialmente o uso de filtragem baseada em conteúdo para a recomendação de exercícios de programação extraclasse. A filtragem baseada em conteúdo recomenda itens semelhantes para um perfil de usuário com base nas características dos itens (ISINKAYE; FOLAJIMI; OJOKOH, 2015). Assim, os itens devem ser representados por um vetor de características e os perfis dos usuários devem ser estimados por meio das preferências do usuário, representando um usuário no mesmo espaço de características dos itens. Com essas representações vetoriais no mesmo espaço, o sistema pode usar uma métrica de similaridade para recomendar um novo item para um dado usuário. Para fins de ilustração, a Tabela 1 apresenta um contexto de recomendação de filmes para usuários de uma plataforma de *streaming*.

	θ_1	θ_2	θ_3	θ_4	θ_5		
	U_1	U_2	U_3	U_4	U_5	x^1 (romance)	x^2 (ação)
F_1	1	5	3	?	3	0.9	0
F_2	0	1	?	?	1	1.0	0.01
F_3	?	1	2	?	4	0.99	0
F_4	?	?	4	?	5	0.1	1.0
F_5	?	?	1	?	?	0	0.9

Tabela 1: Exemplo de contexto de recomendação de filmes com 5 usuários e 5 filmes.

Esse contexto possui 5 usuários (colunas) e 5 filmes (linhas). Cada filme é representado por um vetor de características x_i com duas dimensões, sendo que a primeira dimensão x_i^1 define o quanto aquele filme se enquadra no gênero de romance e a segunda x_i^2 o quanto ele se enquadra no gênero de ação. Cada entrada i, j na tabela representa uma nota que o usuário U_j atribuiu a um filme F_i. Uma entrada com ponto de interrogação significa que o usuário U_j ainda não assistiu o filme F_i. Assumindo que o usuário U_2 está utilizando o sistema de recomendação para decidir seu próximo filme, o sistema deve estimar o perfil desse usuário θ_2 e calcular a similaridade entre θ_2 e os filmes F_4 e F_5 utilizando as representações vetoriais dos mesmos, x_4 e x_5, respectivamente.

O contexto de recomendação desse projeto pode ser modelado da mesma forma, onde os itens são exercícios e os usuários são alunos de disciplinas de programação. Como o objetivo desse projeto é recomendar exercícios adequados à experiência dos alunos, a representação dos exercícios devem caracterizar o nível de dificuldade dos mesmos e o perfil do aluno deve representar o nível de exercícios que ele consegue resolver. Com essa abordagem, os principais desafios desse projeto são: (1) definir uma representação vetorial que capture a dificuldade de um exercício; (2) estimar e/ou inferir o perfil do aluno; (3) definir uma escala de notas para avaliação dos exercícios, e; (4) definir uma métrica de similaridade entre exercícios e perfis de alunos. Para investigar soluções para

essas problemas, esse projeto propõe as seguintes perguntas de pesquisa (QPs):

- **QP 1**: Como extrair características de um exercício que definam a sua dificuldade?

  Exercícios de programação são dados textuais que podem conter metadados produzidos pelo professor que os desenvolveu. Para representar a dificuldade de um exercício, serão exploradas diferentes formas de extração de características de texto, tanto de maneira manual quanto automática. Em representações manuais, os professores das disciplinas definirão, a partir do texto e dos metadados dos exercícios, características que eles julguem representar bem a dificuldade dos exercícios. Em representações automáticas, as características de dificuldade serão extraídas automaticamente por meio de técnicas de processamento de linguagem natural, como *bag of words* ou *word embeddings*.

- **QP 2**: Qual métrica será utilizada pelos alunos para avaliar os exercícios?

  As métricas de avaliação de um exercício são fundamentais para definir os perfis dos usuários e elas podem ser coletadas explícita ou implicitamente. Para responder essa pergunta de pesquisa, serão comparadas diferentes métricas explícitas e implícitas, visando obter um sistema que recomende exercícios de forma mais adequada para os alunos. No caso das métricas explícitas, os alunos irão atribuir notas para os exercícios logo após resolvê-los. As métricas implícitas serão extraídas por meio do comportamento dos alunos durante a solução dos exercícios (e.g., número de submissões).

- **QP 3**: Como estimar o perfil dos alunos?

  O perfil de um aluno é um vetor de características no mesmo espaço de características dos exercícios, que é estimado a partir dos exercícios que esse aluno já resolveu. Esse vetor pode ser calculado diretamente, por exemplo, pela média dos vetores dos exercícios resolvidos, ou aprendidas por meio de um algoritmo de regressão. Para responder essa questão de pesquisa, iremos investigar tanto as técnicas de estimativa direta quanto por meio de aprendizagem.

- **QP 4**: Qual métrica de similaridade é mais adequada para recomendar exercícios de programação?

  Uma vez que o sistema de recomendação é capaz de representar tanto os exercícios quando os perfis dos alunos usando vetores no mesmo espaço de características, ele deve usar uma métrica de similaridade para recomendar exercícios não resolvidos para um dado perfil de aluno. Essa métrica é utilizada para comparar a distância entre o perfil de aluno e todos os exercícios não resolvidos. Dessa forma, o sistema de recomendação pode selecionar o exercício não resolvido mais próximo do perfil, de acordo com a métrica de similaridade definida. Para responder essa questão de

pesquisa, serão exploradas diferentes métricas de similaridade entre vetores, como a similaridade por cosseno, distância euclidiana e k-vizinhos mais próximos.

4 Objetivos

O objetivo geral deste projeto é desenvolver um sistema de recomendação de exercícios extraclasse independente de ACAC, visando personalizar o processo de aprendizagem de alunos de disciplinas de programação. Considerando a metodologia de pesquisa proposta, esse projeto também tem os seguintes objetivos específicos:

- Desenvolver representações vetoriais para exercícios capazes de representar a dificuldade dos mesmos;
- Definir métricas de avaliação de exercícios que representem bem os perfis dos alunos, ou seja, o nível de dificuldade de exercícios que eles são capazes de resolver;
- Desenvolver métodos robustos para estimar o perfil dos alunos;
- Definir métricas de similaridade entre alunos e exercícios que capturem de maneira significativa a distância entre os perfis de alunos e os exercícios.

A hipótese central desse projeto é de que um sistema de recomendação de exercícios extraclasse é capaz de tornar a experiência de aprendizado de um aluno em disciplinas de programação mais suave, aumentando seu desempenho e tornando-o um programador melhor. Para testar essa hipótese, esse projeto propõe experimentos com alunos das disciplinas de programação do curso de Ciência da Computação da UFV. Esses experimentos serão testes com usuários, onde os alunos serão divididos em dois grupos A e B. O grupo A será submetido à uma sequência de atividades avaliativas (e.g., três provas práticas) intercaladas por listas de exercícios recomendadas pelo sistema de recomendação. O grupo B será submetido à mesma sequência de atividades, porém intercaladas por listas de exercícios fixas. Essa sequência de atividades e listas será organizada de tal forma a introduzir os tópicos de programação em ordem crescente de dificuldade.

Para medir a experiência de aprendizagem dos alunos dos dois grupos, serão aplicados questionários de avaliação das atividades logo após cada uma delas. Além disso, serão medidos os desempenhos de cada grupo nessas atividades. Esperamos que os alunos do grupo A avaliem as atividades como mais ajustadas a seus níveis de experiência do que os alunos do grupo B. Além disso, esperamos que o desempenho dos alunos do grupo A seja relativamente maior do que o dos alunos do grupo B.

5 Plano de Trabalho e Cronograma de Execução

Este projeto é espinha dorsal do trabalho de doutorado do aluno Carlos Eduardo Paulino Silva do Programa de Pós-Graduação em Ciência da Computação da UFV. As atividades previstas para a execução completa da pesquisa estão listadas a seguir e compreendem um período de 3 anos:

1. Levantamento do estado da arte de recomendação de exercícios de programação;
2. Implementação do estado da arte como *baseline*;
3. Investigação de métodos manuais e automáticos de extração de características dos exercícios de programação;
4. Avaliação de métricas diretas e indiretas de avaliação de exercícios;
5. Investigação de técnicas de estimativa para estimar o perfil dos alunos;
6. Exploração de diferentes métricas de similaridade entre vetores para comparar a distância entre exercícios e perfis de alunos;
7. Implementação de diferentes variações de sistema de recomendação, de acordo com as técnicas utilizadas nas últimas 4 atividades;
8. Execução de experimentos para verificar e validar o funcionamento dos sistemas desenvolvidos em um ambiente simulado de aprendizagem;
9. Elaboração de artigos científicos.

A Tabela 2 ilustra as atividades do candidato organizadas em um cronograma proposto. É importante destacar que alterações poderão ser realizas durante o período de execução do projeto para melhor adequação das atividades previstas.

		Atividades								
Período		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1º Ano	1º Semestre	X	X	X						
	2º Semestre		X	X	X	X				
2º Ano	1º Semestre			X	X	X	X	X		X
	2º Semestre						X	X	X	
3º Ano	1º Semestre							X	X	X
	2º Semestre								X	X

Tabela 2: Cronograma das atividades por semestre.

6 Considerações Finais

Este documento descreveu por meio de um escopo geral, diversas etapas de um projeto de pesquisa cujo objetivo é explorar a aplicação de técnicas de mineração de dados e aprendizado de máquina com o intuito de investigar e propor abordagens automáticas para recomendação de exercícios de programação extraclasse em ambientes de correção automática de código. Particularmente, acredita-se que ferramentas desta natureza podem ser extremamente úteis para apoiar o processo de aprendizagem dos alunos.

Além disso, um resultado esperado bastante relevante é a capacitação de alunos (i.e., bolsistas) em áreas importantes onde há uma crescente demanda por profissionais qualificados, como análise de dados e aprendizado de máquina.

Ademais, o tema abordado nesse projeto evoluirá no longo prazo, tornando-se mais complexo, desafiador, e ainda mais relevante dentro da Ciência da Computação, especificamente, Mineração de dados, Aprendizado de Máquina e Processamento de Linguagem Natural. Por fim, almeja-se publicar os resultados gerados em revistas e conferências internacionais e nacionais correlatas com este projeto.

6.1 Equipe e Colaborações

O aluno Carlos Eduardo Paulino Silva é orientado pelo Prof. André Gustavo dos Santos (coordenador/proponente) e pelos Profs. Julio Cesar Soares dos Reis e Lucas Nascimento Ferreira (coorientadores). Os professores coorientadores supracitados serão responsáveis pela orientação dos bolsistas alocados juntamente com o doutorando.

Finalmente, é importante mencionar que durante a execução deste projeto, vislumbram-se interações e colaborações com grupos de pesquisas da UFV que tangenciam áreas correlacionadas a este projeto de pesquisa. Dessa forma, convites poderão ser inicialmente enviados aos docentes do Departamento de Informática da UFV que tenham interesse em tais linhas de pesquisa. Futuramente, esses convites podem ser estendidos a pesquisadores de outras universidades no âmbito nacional e internacional com interesse no tema. Por exemplo, durante seu período de treinamento de pós-doutorado, o professor proponente deste projeto teve contato com professores do Departamento de Ciência de Computadores da Universidade do Porto, Portugal, que usam ACACs em disciplina de programação. Um deles já demonstrou interesse em ACAC customizado.

Referências

BECKER, B. A.; QUILLE, K. 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. [S.l.: s.n.], 2019. p. 338–344. Citado na página 1.

BEZ, J. L.; FERREIRA, C. E.; TONIN, N. Uri online judge academic: A tool for professors. In: ATLANTIS PRESS. *2013 International Conference on Advanced ICT and Education (ICAICTE-13)*. [S.l.], 2013. p. 744–747. Citado 2 vezes nas páginas 1 e 3.

CAMPOS, C.; FERREIRA, C. Boca: um sistema de apoio a competições de programação. In: *Workshop de Educação em Computação*. [S.l.]: Sociedade Brasileira de Computação, 2004. Citado na página 3.

FRANCISCO, R. E. et al. Juiz online no ensino de CS1-lições aprendidas e proposta de uma ferramenta. *Revista Brasileira de Informática na Educação*, v. 26, n. 03, p. 163, 2018. Citado na página 3.

GALVÃO, L.; FERNANDES, D.; GADELHA, B. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. In: SBC. *Anais do XXVII Simpósio Brasileiro de Informática na Educação*. [S.l.], 2016. v. 27, n. 1, p. 140. Citado na página 3.

ISINKAYE, F. O.; FOLAJIMI, Y. O.; OJOKOH, B. A. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, Elsevier, v. 16, n. 3, p. 261–273, 2015. Citado na página 6.

JÚNIOR, H. B. de F. et al. Recomendação automática de problemas em juízes online usando processamento de linguagem natural e análise dirigida aos dados. In: SBC. *Anais do XXXI Simpósio Brasileiro de Informática na Educação*. [S.l.], 2020. p. 1152–1161. Citado 3 vezes nas páginas 1, 2 e 5.

LARANJEIRA, D. R. *Recomendação de exercícios para alunos de programação em um ambiente de correção automática de códigos*. Dissertação (Mestrado em Informática) — Universidade Federal do Amazonas, Manaus, 2020. 110 f. Citado 2 vezes nas páginas 1 e 5.

LIMA, M. A. P. de L. et al. Uso de atributos de código para classificação da facilidade de questões de codificação. In: SBC. *Anais do Simpósio Brasileiro de Educação em Computação*. [S.l.], 2021. p. 113–122. Citado 2 vezes nas páginas 1 e 4.

MEDEIROS, R. P.; RAMALHO, G. L.; FALCÃO, T. P. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, IEEE, v. 62, n. 2, p. 77–90, 2018. Citado na página 1.

NEVES, A. et al. PCodigo II: O sistema de diagnóstico da aprendizagem de programação por métricas de software. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, v. 6, n. 1, p. 339, 2017. ISSN 2316-8889. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/wcbie/article/view/7398>. Citado na página 4.

PEREIRA, F. D. et al. Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. *British journal of educational technology*, Wiley Online Library, v. 51, n. 4, p. 955–972, 2020. Citado 2 vezes nas páginas 1 e 3.

SANTOS, P. dos et al. Classificação de dificuldade de questões de programação com base na inteligibilidade do enunciado. In: SBC. *Anais do XXX Simpósio Brasileiro de Informática na Educação*. [S.l.], 2019. v. 30, n. 1, p. 1886. Citado na página 4.