UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

RECOMENDAÇÃO DE EXERCÍCIOS PARA ALUNOS DE PROGRAMAÇÃO EM UM AMBIENTE DE CORREÇÃO AUTOMÁTICA DE CÓDIGOS

DION RIBEIRO LARANJEIRA

Área: Informática na Educação
Linha de Pesquisa: Inteligência Artificial
Orientadora: Elaine Harada Teixeira de Oliveira
Coorientador: David Braga Fernandes de Oliveira

Março de 2020
Manaus - AM

DION RIBEIRO LARANJEIRA

RECOMENDAÇÃO DE EXERCÍCIOS PARA ALUNOS DE PROGRAMAÇÃO EM UM AMBIENTE DE CORREÇÃO AUTOMÁTICA DE CÓDIGOS

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas (PPGI/IComp, UFAM) como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Orientadora: Elaine Harada Teixeira de Oliveira, D.Sc.

Coorientador: David Braga Fernandes de Oliveira, D.Sc.

Março de 2020
Manaus - AM

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Laranjeira, Dion Ribeiro
L318r Recomendação de exercícios para alunos de programação em um ambiente de correção automática de códigos / Dion Ribeiro Laranjeira . 2020
95 f.: il. color; 31 cm.

Orientadora: Elaine Harada Teixeira de Oliveira
Coorientador: David Braga Fernandes de Oliveira
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. programação introdutória. 2. juízes online. 3. sistemas de recomendação. 4. filtragem colaborativa. I. Oliveira, Elaine Harada Teixeira de. II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Recomendação de exercícios para alunos de programação em um ambiente de correção automática de códigos"

DION RIBEIRO LARANJEIRA

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Profa. Elaine Harada Teixeira de Oliveira - PRESIDENTE

Prof. Eduardo James Pereira Souto - MEMBRO INTERNO

Prof. Leandro Silva Galvão de Carvalho - MEMBRO EXTERNO

Manaus, 27 de Março de 2020

Resumo da Dissertação apresentada ao PPGI/IComp/UFAM como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática.

RECOMENDAÇÃO DE EXERCÍCIOS PARA ALUNOS DE PROGRAMAÇÃO EM UM AMBIENTE DE CORREÇÃO AUTOMÁTICA DE CÓDIGOS

DION RIBEIRO LARANJEIRA

Março/2020

Orientador(a): Elaine Harada Teixeira de Oliveira, D.Sc.

Coorientador(a): David Braga Fernandes de Oliveira, D.Sc.

Muitos alunos de disciplinas de Programação Introdutória (CS1) têm dificuldade em aprender a programar. Por isso, professores de programação têm utilizado juízes online para propor exercícios, realizar maratonas e campeonatos de programação, a fim de tentar melhorar a experiência de aprendizado dos alunos. No entanto, nos casos de juízes online que possuem em sua base de dados muitos exercícios cadastrados, o aluno poderá escolher um exercício que não seja adequado para seu nível de conhecimento. Nesse sentido, neste trabalho é proposto um sistema de recomendação de exercícios, que filtra exercícios por nível de dificuldade, em um juiz online chamado CodeBench. Esses exercícios são classificados manualmente por assuntos pelo professor, e o método proposto nesta pesquisa sugere exercícios ordenados pelo nível de dificuldade. Para isso, é utilizada a abordagem de recomendação de filtragem colaborativa para mapear as dificuldades experimentadas pelos alunos quando resolvem exercícios de programação no ambiente de desenvolvimento integrado do CodeBench. Após isso, é feita a predição de dificuldade dos exercícios que o aluno ainda não resolveu para, então, sugerir exercícios com graus de dificuldade crescentes. Este método de recomendação foi aplicado em uma base de dados com 645 alunos de graduação, distribuídos em 14 turmas de CS1, ministradas no ano de 2018, em uma universidade pública. A disciplina é dividida em 7 módulos, cada módulo possui duas listas de exercícios e uma prova. Em cada módulo foi feita a comparação da

ordem original de resolução de exercícios com a ordem de resolução de exercícios proposta pelo método. Os resultados mostram que em 6 módulos da disciplina o método aqui proposto sugere uma ordem de resolução adaptada para cada aluno, com nível crescente de dificuldade.

Palavras-chave: programação introdutória, juízes online, sistemas de recomendação e filtragem colaborativa.

Abstract of Dissertation presented to PPGI/IComp/UFAM as a partial fulfillment of the requirements for the degree of Master in Informatics.

CONTENT RECOMMENDATION FOR PROGRAMMING STUDENTS IN AUTOCORRECT ENVIRONMENTS

DION RIBEIRO LARANJEIRA

March/2020

Advisor: Elaine Harada Teixeira de Oliveira, D.Sc.

Co-advisor: David Braga Fernandes de Oliveira, D.Sc.

Many students in Introductory Programming (CS1) subjects have difficulty learning to program. For this reason, programming teachers have used online judges to propose exercises, run marathons and programming championships in order to try to improve students' learning experience. However, in the case of online judges who have many registered exercises in their database, the student may choose an exercise that is not suitable for their level of knowledge. In this sense, this work proposes an exercise recommendation system, which filters exercises by level of difficulty, in an online judge called CodeBench. These exercises are classified manually by subjects by the teacher, and the method proposed in this research suggests exercises ordered by the level of difficulty. For this, the collaborative filtering recommendation approach is used to map the difficulties experienced by students when solving programming exercises in the CodeBench integrated development environment. After that, the prediction of the difficulty of the exercises is made that the student has not yet solved to then suggest exercises with increasing degrees of difficulty. This recommendation method was applied to a database of 645 undergraduate students, distributed in 14 CS1 classes, taught in 2018, at a public university. The course is divided into 7 modules, each module has two lists of exercises and a test. In each module the comparison of the original order of resolution of exercises with the order of resolution of exercises proposed by the method was made. The results show that in 6 modules of

the discipline the method proposed here suggests an order of resolution adapted for each student, with level increasing difficulty.

Keywords: introductory programming, online judges, recommendation systems and collaborative filtering.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Disciplinas de CS1 são bases de cursos de computação. Os índices de reprovação ou desistência em tais disciplinas são considerados altos. De acordo com uma pesquisa realizada por [Oliveira et al. 2016], no ano de 2016 cerca de 42% dos alunos de ensino superior afirmaram que reprovaram em disciplinas de linguagem de programação, lógica de programação ou algoritmos, seguido por uma grande quantidade de reprovações em banco de dados, que é uma disciplina relacionada a essas e que também necessita um bom raciocínio lógico do aluno.

[Souza et al. 2016] fizeram uma revisão sistemática da literatura para identificar a causa do grande número de reprovações em disciplinas de programação. Seis causas foram identificadas, sendo elas: (i) alunos apresentam dificuldades em aprender conceitos de programação; (ii) alunos apresentam dificuldades em aplicar conceitos de programação; (iii) alunos apresentam dificuldades em compreender códigos de programas; (iv) alunos apresentam dificuldade em dividir problemas em módulos; (v) alunos apresentam falta de interesse ou desânimo para resolver problemas de programação; (vi) professores apresentam dificuldades em ensinar os conceitos de programação, desenvolver materiais e exercícios de apoio, bem como avaliar os trabalhos de programação.

A fim de minimizar os índices de reprovação ocasionados pelos seis fatores citados anteriormente, [Souza et al. 2016] propõem possíveis soluções. Dentre elas, encontram-se:

- Desenvolvimento de ambientes pedagógicos: utilização e desenvolvimento de ambientes que proveem funcionalidades para construção e execução de programas, mas visando ao ensino e a aprendizagem de programação.

- Resolução de problemas reais: atividades em que os alunos irão construir programas que podem ser utilizados para alguma finalidade (ao contrário dos *toy programs*, que após as atividades costumam ser descartados).

- *Scaffolding*: estratégias de ensino em que o professor vai adaptando a tarefa de acordo com o nível de habilidade dos alunos, realimentando-a por contínuo *feedback* durante a progressão da tarefa.

Um recurso muito utilizado por professores de computação, são os ambientes de correção automática de código (ACAC), também conhecidos como juízes online. Sua função principal é avaliar códigos-fonte em uma determinada linguagem de programação [Kurnia et al. 2001]. Os juízes online auxiliam o professor, pois oferecem recursos para gerenciar listas de exercícios, que podem ser replicadas para todos os alunos e gerar relatórios com dados de: questões resolvidas, quantidade de tentativas, tempo de execução do código, dentre outros.

Apesar dos benefícios de criar listas de exercícios em juízes online serem evidentes, cada exercício apresenta diferentes níveis de dificuldades para diferentes alunos. Ou seja, um determinado aluno pode achar um exercício fácil e outro aluno pode achá-lo difícil. Apresentar exercícios em uma mesma ordem para todos os alunos não satisfaz a estratégia de *Scaffolding*.

Adaptar tarefas de acordo com o nível do aluno não é simples, pois é desafiador identificar e recomendar materiais personalizados, baseando-se em necessidades individuais de cada aluno, além de interesses e competências a serem desenvolvidos [Cazella et al. 2009].

No entanto, grandes empresas como *Netflix*, *Google*, *Facebook* e *Amazo*n conseguem "tratar" um usuário dentre milhões de forma individual através do uso de sistemas de recomendação. Um sistema de recomendação é um agente de software que tenta recomendar ações a um usuário de forma inteligente com base nas ações de usuários anteriores [Zaíane 2002].

Nesse sentido, este trabalho propõe e valida um método de recomendação de exercícios em ambientes de correção automática de código. Para isso, o método leva em conta os índices de dificuldade de exercícios resolvidos por alunos.

Enquanto os alunos resolviam exercícios utilizando o ACAC foram determinados os índices de dificuldade dos exercícios. Para isso, foram coletados 10 atributos que representam a dificuldade de cada exercício. A quantidade de vezes que o aluno testa o código,

quantidade erros cometidos, tipos de erros cometidos e tamanho do *log* registrado pelo ACAC são alguns dos atributos coletados e utilizados para determinar o índice de dificuldade do exercício. Esses dados podem ser coletados em qualquer ACAC, e neste trabalho, foram armazenados em uma estruturada denominada perfil de programação.

Após isso, usando a abordagem de recomendação de filtragem colaborativa, foi feita a recomendação de exercícios por nível crescente de dificuldade. Após o aluno receber as recomendações, foram feitas análises para verificar a qualidade da recomendação.

1.1 Justificativa

Aprender programação de computadores não é uma tarefa simples para algumas pessoas. Diversas pesquisas têm sido feitas para melhorar o processo de aprendizagem de alunos de disciplinas introdutórias à programação de computadores. De acordo com [Gomes et al. 2008], o fato de alunos terem dificuldades em aprender a programar pode ser ocasionado por vários motivos, sendo os principais: falta de interesse por parte dos alunos, diversos conceitos abstratos necessários para aprender a programar, necessidade de treino intensivo em resolução de problemas, inadequação dos métodos pedagógicos aos estilos de aprendizagem dos alunos.

[Alves and Jaques 2014] afirmam que é fundamental a prática dos conteúdos vistos em aula para que o aprendizado de programação seja efetivo. Sendo assim, o professor deve elaborar trabalhos com desafios graduais de programação e disponibilizar para seus alunos. Quando um aluno soluciona um conjunto de questões ordenadas pelo nível de dificuldade, ele desenvolve suas habilidades de forma gradual e crescente. Cada questão ajuda o aluno a desenvolver as habilidades que serão úteis para as questões seguintes [Ng and Bereiter 1991].

O processo de personalização de trabalhos e *feedback* é custoso para o professor, pois deve ser feita a correção desses trabalhos e uma sinalização dos erros encontrados. Como em sua maioria, as turmas de programação são numerosas, o professor geralmente não consegue corrigir imediatamente todos os trabalhos. Por isso, os alunos não recebem uma resposta imediata do seu desempenho nos trabalhos realizados.

Muitos usuários utilizam ambientes de correção automática de códigos, um exemplo é o URI Online Judge[1], que contém mais de 177.000 usuários. Uma das principais vantagens desses sistemas é a existência de *feedback* imediato após a submissão da solução do exercício.

O uso de juízes online como ferramenta de apoio no ensino de programação pode ajudar a melhorar os índices de aprovação. A Figura 1.1 mostra os resultados obtidos no estudo feito por [Galvão et al. 2016].

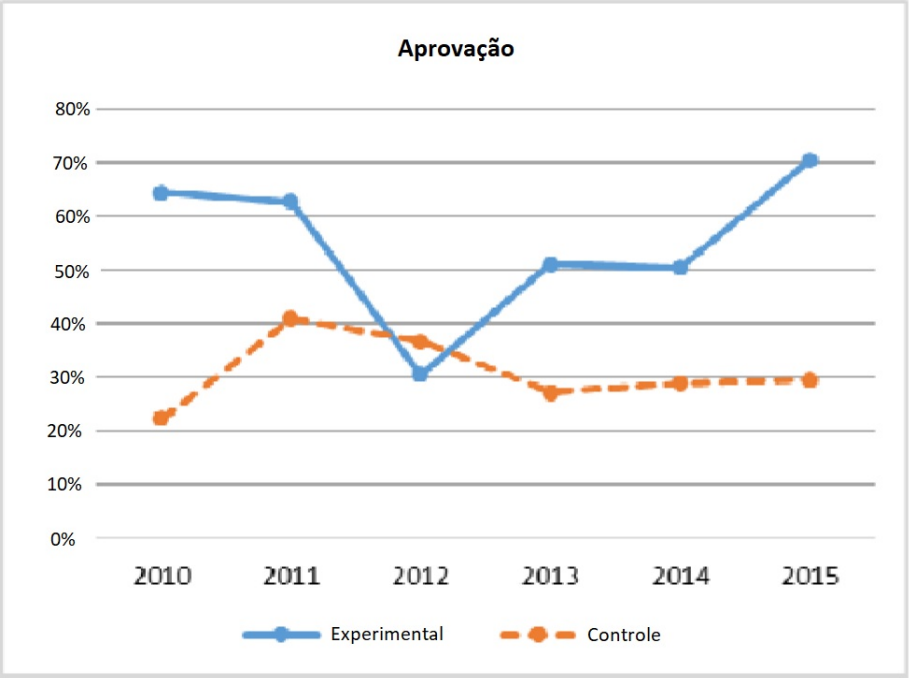


Figura 1.1 – Evolução das taxas de aprovação na disciplina de CS1
Fonte: [Galvão et al. 2016]

Nessa pesquisa, os autores utilizaram um juiz online em uma turma experimental. Os dados mostram, que a taxa de aprovação é maior na turma experimental do que na turma de controle. O uso do juiz online proporcionou um aumento nos índices de aprovação, que saltaram de 50% para 70% entre 2014 e 2015 nas turmas experimentais, ao passo que se manteve em torno de 30% nas turmas de controle [Galvão et al. 2016].

No entanto, [Yu and Chen 2016] relatam que nos últimos anos a quantidade de exercícios existentes nos juízes onlines aumentou, por isso, tornou-se cada vez mais difícil para os usuários buscarem os exercícios mais adequados para o seu nível de conhecimento sobre programação.

[1]https://www.urionlinejudge.com.br/judge/pt/rank?page=6842

[Isinkaye et al. 2015] afirmam que o crescimento explosivo da quantidade de informações digitais disponíveis criou um potencial desafio de sobrecarga de informações, o que dificulta o acesso oportuno a itens de interesse aos usuários. Isso ocasionou o aumento do uso de sistemas de recomendação.

Os sistemas de recomendação são sistemas de filtragem de informação que lidam com o problema da sobrecarga de informação, que é um dos pontos fracos existentes nos juízes online.

Nesse sentido, o presente estudo tem o intuito de prover uma estrutura adaptada a um juiz online, que recomendará para alunos exercícios de acordo com nível de dificuldade dos exercícios resolvidos anteriormente.

Para fazer a recomendação personalizada ao aluno, será necessário identificar suas características pessoais, para que seja possível "conhecê-lo". Esse processo é comum em todos os sistemas de recomendação, como mostram [Isinkaye et al. 2015] em sua RSL sobre o assunto. Neste trabalho, pretende-se fazer a coleta dos dados de forma implícita através da análise de códigos submetidos pelo aluno no juiz online.

[Souza et al. 2016] afirmam que um ambiente pedagógico que proporciona ao aluno a resolução de problemas reais, adaptados de acordo com seu nível de habilidade é uma das possíveis soluções para minimizar a quantidade de reprovações em disciplinas de programação. Por isso, acredita-se que este trabalho poderá melhorar a experiência de aprendizagem dos alunos, pois iremos propor listas de exercícios de programação personalizadas de acordo com o nível de dificuldade de cada aluno.

1.2 Objetivo Geral

O objetivo geral deste trabalho é desenvolver e validar um método que sugere ao aluno exercícios na ordem crescente de dificuldade em juízes online com base na análise de códigos de exercícios resolvidos.

1.3 Objetivos Específicos

Para se atingir o objetivo principal do trabalho proposto, faz-se necessário alcançar um conjunto de objetivos intermediários, sendo:

a. Desenvolver uma estratégia para mapear os índices de dificuldade dos exercícios resolvidos pelos alunos de disciplinas de CS1 utilizando os *logs* registrados pelo ACAC.

b. Utilizar os dados coletados sobre os índices de dificuldade dos exercícios resolvidos e aplicar a filtragem colaborativa para recomendar exercícios de programação com índices de dificuldade crescente.

c. Comparar os índices de dificuldade dos exercícios resolvidos com e sem o método proposto nesta pesquisa.

1.4 Questões de Pesquisa

Neste trabalho foi desenvolvido e validado um método de recomendação de exercícios em juízes online. As recomendações foram feitas com base nos dados sobre o índice de dificuldade de cada exercício resolvido pelo aluno enquanto utilizava um juiz online. Especificamente, este trabalho está focado em responder as questões abaixo:

1. É possível desenvolver um método que forneça aos alunos uma sequência de exercícios com níveis crescente de dificuldade?

2. É possível desenvolver um método de recomendação de exercícios que leve em consideração as individualidades dos alunos?

3. É possível desenvolver algum mecanismo automático de recomendação de exercícios que disponha as questões em uma ordem mais adequada aos alunos do que aquela recomendada pelos professores?

1.5 Metodologia

A Figura 1.2 ilustra o processo de construção do modelo de recomendação de exercícios desta pesquisa. Primeiramente, foi realizada a coleta e normalização de dados sobre os exercícios resolvidos pelos alunos enquanto utilizavam o juiz online. Para cada exercício resolvido, são coletados dez atributos que contêm indícios sobre a dificuldade do aluno, aqui intitulado de índice de dificuldade do exercício, determinado a partir de estatísticas extraídas do *log* gerado pelos alunos durante a resolução do exercício. A quantidade

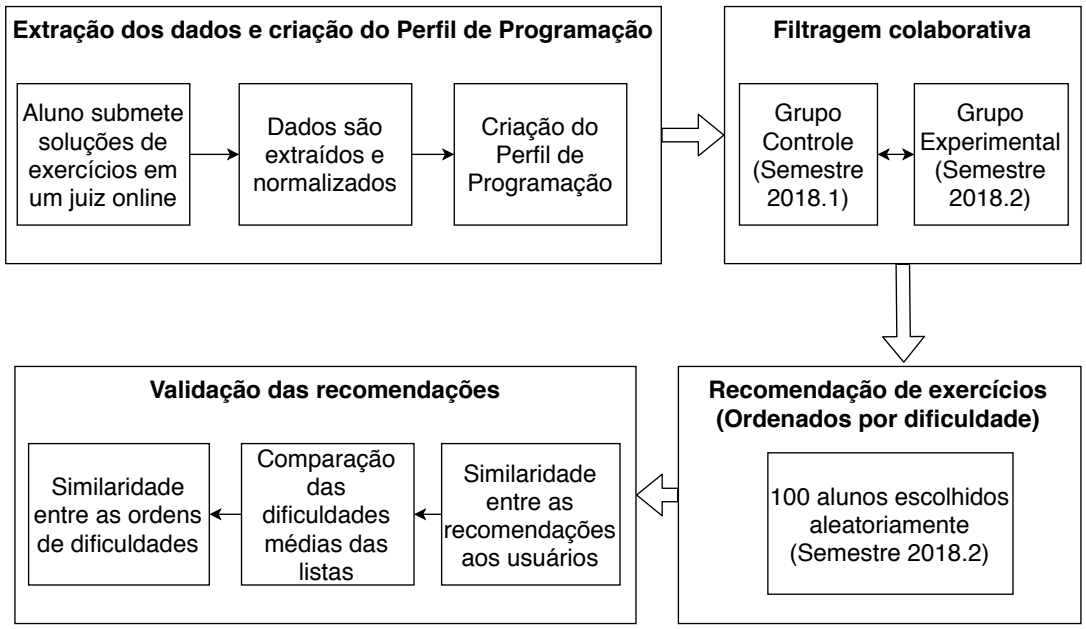


Figura 1.2 – Fluxograma de aplicação do método proposto
Fonte: Própria

de erros, quantidade de tentativa de submissão, tamanho do código submetido e tipo de erro cometido são exemplos dos dados coletados. Esses dados são armazenados em uma estrutura denominada Perfil de Programação (PP).

Após isso, os alunos foram divididos em dois grupos, sendo o grupo controle, composto pelos alunos do semestre 2018.1 e o grupo experimental, que é alvo da recomendação, pelos alunos do semestre 2018.2.

Logo após, para cada aluno pertencente ao grupo experimental, foi feita a filtragem colaborativa calculando a similaridade em relação aos alunos do grupo controle. A relação de similaridade calculada leva em consideração o quão parecido os alunos são, de acordo com os índices de dificuldade dos exercícios resolvidos, ou seja, o quão parecido é o PP de um aluno com o de outro.

Na próxima etapa, foi feita a recomendação de exercícios para 100 alunos escolhidos aleatoriamente pertencentes ao grupo experimental. A recomendação foi feita utilizando dois critérios, sendo:

- Similaridade entre os alunos do grupo experimental com os alunos do grupo controle: para fim de cálculos da similaridade, foram considerados apenas os exercícios avaliativos (exercícios que não são opcionais nem exames);

- Dificuldade que os alunos do grupo controle sentiram ao resolver exercícios opcionais: com base nisso, de acordo com a similaridade entre os alunos, foi prevista a dificuldade que os alunos do grupo experimental sentiriam ao resolver exercícios opcionais.

Finalmente, foi realizada a validação das recomendações feitas aos alunos do grupo experimental. As validações foram feitas em três etapas, sendo elas:

- Similaridade entre as recomendações aos alunos: nessa etapa foi verificada a similaridade entre as listas de exercícios opcionais recomendadas aos alunos. O cálculo da similaridade entre as listas de exercícios recomendados tem o objetivo de mostrar que há diferenças entre as recomendações para os alunos. Uma vez que os alunos apresentam dificuldades e níveis de aprendizagem diferentes, não faz sentido que as recomendações fossem iguais.

- Comparação das dificuldades médias das listas de exercícios: nessa etapa utilizando o PP dos alunos do grupo experimental, foram determinadas as dificuldades que os alunos sentiram ao responder a lista de exercícios opcionais na ordem proposta pelos professores e na ordem proposta pelo método de recomendação. Após isso, foi feita a comparação de dificuldades entre as listas de exercícios.

  O objetivo desse experimento é verificar se a ordem de exercícios proposta pelo método de recomendação possui menor dificuldade em relação á ordem de exercícios proposta pelos professores.

- Similaridade entre as ordens de dificuldades: nesta última etapa utilizando os dados do PP foram coletadas as dificuldades que os alunos do grupo experimental sentiram ao resolver a lista de exercícios. Após isso, para cada aluno, foi feita a ordenação dos exercícios por nível de dificuldade, sempre da menor dificuldade para maior dificuldade. Esses exercícios ordenados foram chamados de ordem ideal de exercícios.

  Então, foi feita a comparação dos exercícios propostos pelos professores e os exercícios propostos pelo método de recomendação com a ordem ideal de exercícios. Esse experimento busca verificar qual ordem de exercícios é mais semelhante à ordem ideal de exercícios.

1.6 Contexto Educacional

Para desenvolver o método de recomendação de exercícios aos alunos, foram coletados dados de 14 turmas de CS1 da Universidade Federal do Amazonas, durante o ano letivo de 2018. No total, 645 alunos de disciplinas de CS1 resolveram 7 listas de exercícios avaliativos e 7 listas de exercícios opcionais usando a linguagem de programação *Python*, versão 3.

Os alunos tinham permissão de fazer um número ilimitado de tentativas de submissão, desde que estivessem dentro do prazo para entrega de cada lista. Isso gerou um total de 619.305 códigos desenvolvidos pelos alunos.

Cada uma das 7 listas de exercícios corresponde a um módulo da disciplina de CS1. A seguir, relacionam-se os assuntos estudados em cada módulo:

- Variáveis e estrutura de programação sequencial;
- Estrutura condicional simples e composta;
- Estrutura condicional encadeada;
- Estrutura de repetição por condição;
- Vetores e strings;
- Estrutura de repetição por contagem;
- Matrizes.

Cada um dos 7 módulos possui exercícios avaliativos, exercícios opcionais e exames. Os exercícios avaliativos são utilizados para aprendizagem de conceitos. Exercícios opcionais são um pouco mais complexos do que os exercícios avaliativos.

Os exercícios avaliativos e exercícios opcionais são escolhidos manualmente pelos professores que ministram a disciplina de CS1. A mesma ordem de exercícios é proposta para todos os alunos, sem considerar a individualidade de cada um.

O método proposto nesta pesquisa recomenda os mesmos exercícios opcionais propostos pelos professores da disciplina. Porém, em uma ordem individualizada para os alunos. Para isso, é levada em consideração a dificuldade dos exercícios e similaridade com outros alunos que resolveram os exercícios anteriormente.

1.7 Organização do documento

Com relação à organização, este documento encontra-se dividido em seis capítulos.

No Capítulo 2 é apresentada a fundamentação teórica necessária para o entendimento do presente estudo, onde são apresentados os principais conceitos relativos a juízes online e sistemas de recomendação de conteúdo.

No Capítulo 3 são apresentados os trabalhos que possuem objetivos e métodos de pesquisa semelhantes ao deste e que foram destacados como o corrente estado da arte. Adiante é feita uma descrição das publicações, categorizando-as por local em que são feitas as recomendações, tipo de usuários para quais as recomendações são feitas, técnicas de filtragem que são utilizadas para recomendação de conteúdo e métricas utilizadas para validação de conteúdo recomendado.

No Capítulo 4 é apresentado o método proposto nesta pesquisa, a arquitetura do trabalho, a estrutura de dados utilizada para mapear os índices de dificuldade dos exercícios resolvidos pelos alunos e a técnica utilizada para recomendar os exercícios para os alunos.

No Capítulo 5 são apresentados os experimentos e resultados obtidos nesta pesquisa e as análises dos resultados. Finalmente, no Capítulo 6, são feitas as considerações finais, são apresentadas as limitações desta pesquisa e são feitas propostas de trabalhos futuros.

Capítulo 2

Referencial Teórico

Neste capítulo discutimos os conceitos básicos de juízes online e sistemas de recomendação. Apresentamos uma breve conceituação dos principais tópicos necessários para o bom entendimento do método proposto que será apresentado nos próximos capítulos.

2.1 Juízes online

Os juízes online são sistemas projetados para a avaliação confiável do código-fonte do algoritmo enviado pelos usuários, que é compilado e testado em um ambiente homogêneo [Wasik et al. 2018].

Caso a solução proposta pelo algoritmo do usuário seja correta, uma mensagem irá aparecer confirmando ao usuário que sua solução é válida para o problema em questão. Nos casos em que a solução do usuário não está correta, é mostrada uma mensagem de erro. Os erros geralmente são classificados nas seguintes categorias [Toledo 2014]:

- Resposta errada: acontece quando a solução proposta pelo usuário não passa em algum caso de teste executado pelo juiz online;
- Erro de compilação: acontece quando o compilador da linguagem não consegue compilar o código submetido. Esse tipo de erro acontece somente em linguagens de programação compiladas, como é o caso das linguagens de programação C e C++.
- Erro de tempo de execução: o programa falha durante a execução devido a uma falha de segmentação, exceção de ponto flutuante ou problema semelhante. Esse

tipo de erro acontece somente em linguagens de programação interpretadas, como é o caso das linguagens de programação C# e Python.

- Erro de apresentação: ocorre quando o código submetido está correto, porém não está no formato requerido pelo juiz online. Exemplo: o código submetido gera "soma=10"e o juiz online requer "SOMA = 10";

- Limite de tempo excedido: a solução proposta pelo usuário demora mais tempo para ser executado do que o tempo estabelecido pelo juiz online.

A Figura 2.1 mostra a tela de feedback de erros do Juiz Online URI Online Judge[1], com a mensagem de erro de compilação.

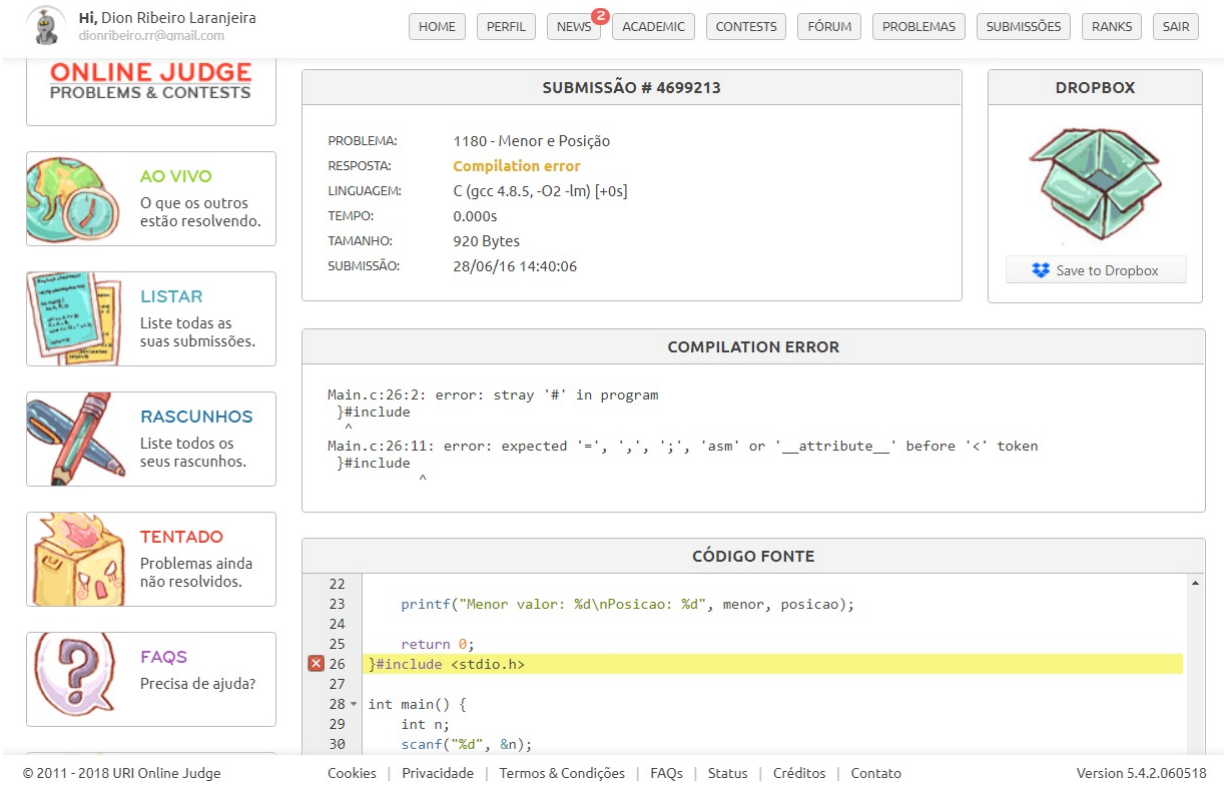


Figura 2.1 – Tela de feedback de erros que um usuário cometeu no Juiz Online URI.
Fonte: https://www.urionlinejudge.com.br

Outro recurso didático interessante presente em alguns juízes online é a possibilidade de um usuário cadastrado como professor criar listas de exercícios que podem ser replicadas para diversas turmas. Nessas listas o professor pode acompanhar: atividades resolvidas corretamente ou erroneamente pelo aluno, quantidade de tentativas realizadas, quais erros cometidos e qual linguagem de programação foi utilizada.

[1] https://urionlinejudge.com.br

A Figura 2.2 mostra um relatório gerado pelo juiz online URI Online Judge, versão Academic (versão para professores). Esse relatório apresenta várias informações importantes, tais como: exercícios que os alunos de determinada turma fizeram corretamente (marcados com a cor verde), exercícios com erros (marcados com a cor vermelha) e exercícios que não tiveram nenhuma submissão (sem nenhuma cor de preenchimento).

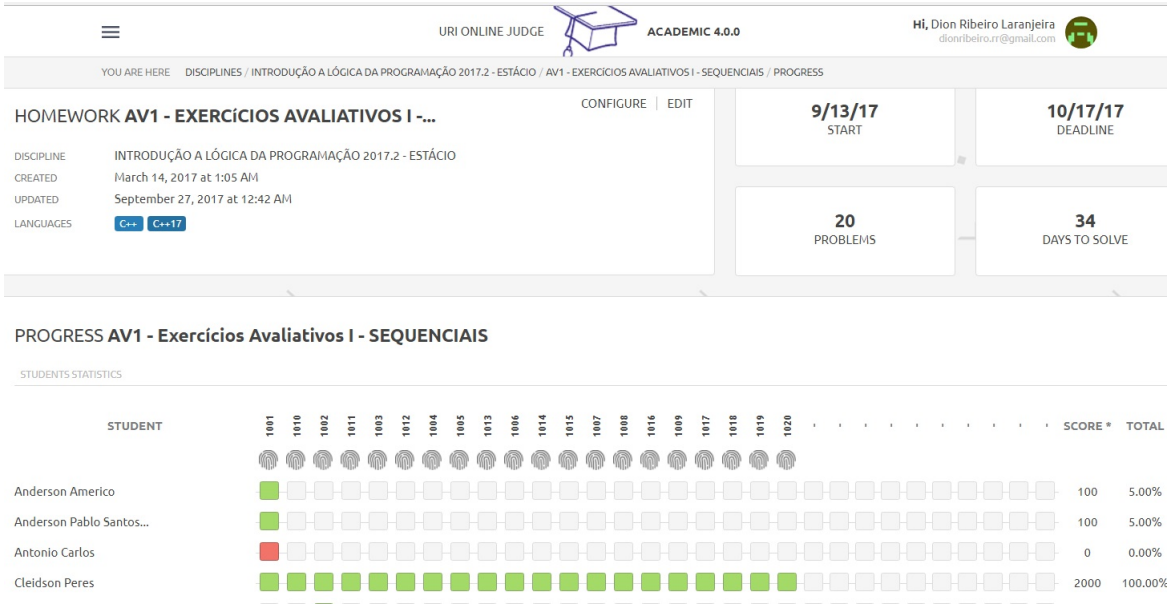


Figura 2.2 – Tela de relatórios da versão para professores do Uri Online Judge.
Fonte: https://www.urionlinejudge.com.br/academic

Como é apresentado no relatório, o professor também tem a possibilidade de criar uma data de início, data de fim e escolher os exercícios que achar mais adequado. Porém, na grande maioria dos juízes online existentes, a criação da lista de exercícios deve ser feita manualmente pelo professor. No ano de 2019, foi implementado um método de recomendação de exercícios no *URI online Judge*, porém, poucos juízes online apresentam esse recurso.

Nesta pesquisa o juiz online utilizado foi o CodeBench[1]. O CodeBench é um sistema online de correção automática de código-fonte, desenvolvido pelo Instituto de Computação da UFAM, para dar suporte a professores e alunos de disciplinas de programação. Através do CodeBench, os professores podem disponibilizar exercícios de programação para seus alunos, que por sua vez podem desenvolver soluções para tais exercícios e submetê-las através da interface do sistema.

[1] http://codebench.icomp.ufam.edu.br/

A Figura 2.3 mostra o Ambiente de Desenvolvimento Integrado ou *Integrated Development Environment* (IDE) do CodeBench, que é usado pelos alunos para desenvolver as soluções dos exercícios propostos. Esse ambiente suporta as principais funcionalidades de um IDE típico, tais como: *autocompletion*, *autosave*, *syntax highlighting*, busca e substituição de *strings*, etc.

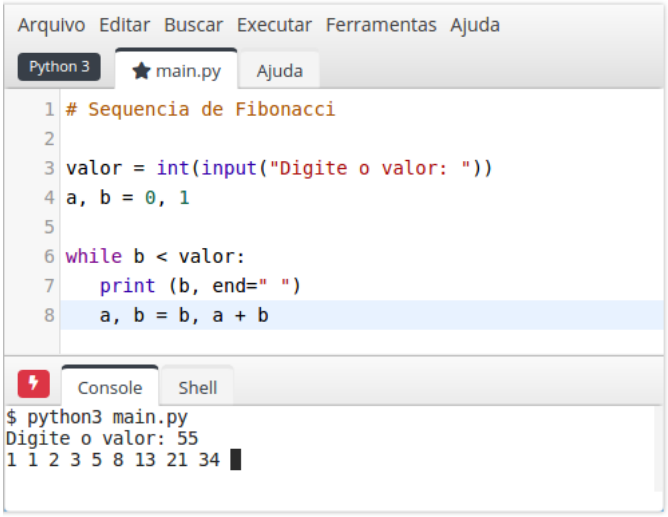


Figura 2.3 – IDE do juiz online CodeBench.
Fonte: http://cbtestes.icomp.ufam.edu.br/

Atualmente, o CodeBench suporta as seguintes linguagens de programação: C, C++, Java, Python, Haskell, Lua, Prolog, Script Bash e Assembly (ARM). Além dessas linguagens, o ambiente também suporta a linguagem SQL, para exercícios envolvendo consultas a bancos de dados.

Além disso, o CodeBench disponibiliza recursos de:

- Gamificação: além de praticar programação, os alunos ajudam a libertar o reino Midgard das garras do poderoso monstro Quimera.
- Banco de exercícios: o CodeBench conta com mais de 3000 exercícios já cadastrados, que podem ser usados pelos professores para compor os trabalhos de suas turmas.
- Materiais Didáticos: os materiais didáticos são recursos pedagógicos compartilhados pelos professores para auxiliar os alunos em seu processo de aprendizagem.
- Troca de mensagens: cada usuário do CodeBench possui uma caixa de e-mails que pode ser usada para troca de mensagens entre os demais usuários do sistema.

- Detecção de plágio: o CodeBench possui um detector de plágio que identifica códigos semelhantes submetidos pelos alunos para um mesmo exercício.
- Tecnologia *Docker*: os códigos dos alunos são executados dentro de ambientes seguros e virtualizados chamados de *docker containers*.

A Figura 2.4 mostra um recurso disponível para o professor, que permite visualizar os códigos desenvolvidos pelos alunos para os exercícios de um dado trabalho.

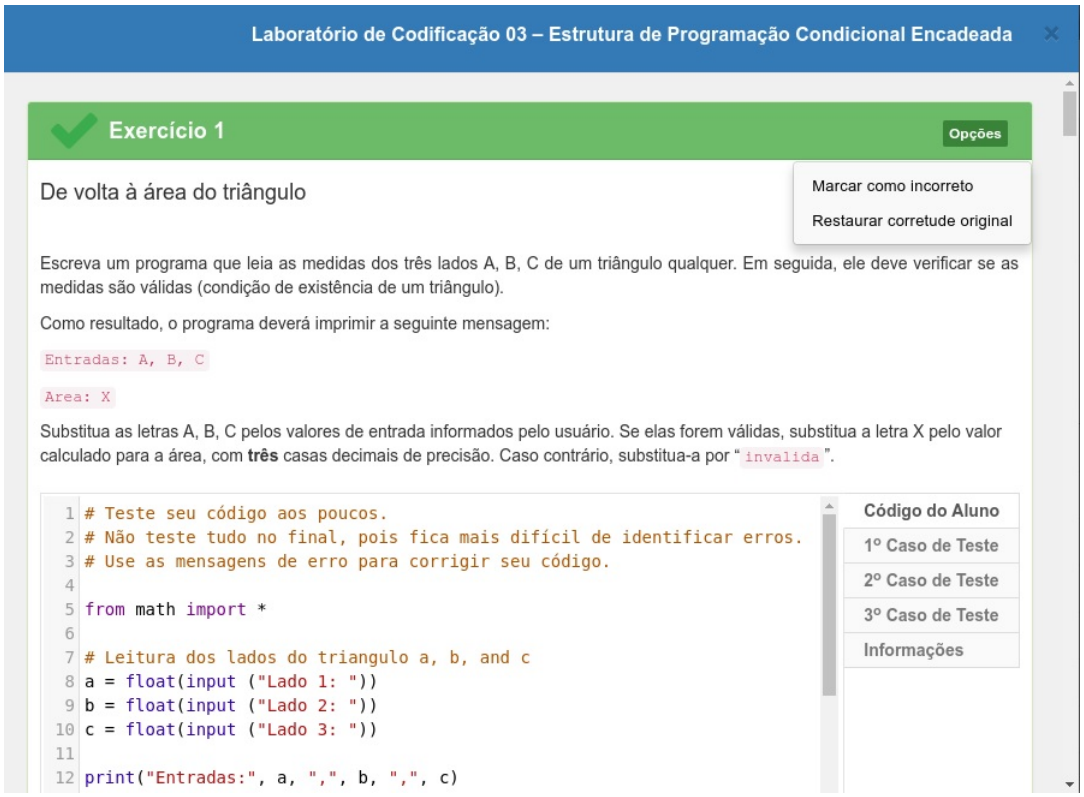


Figura 2.4 – Código de um exercício resolvido por um aluno, disponível para visualização do professor no CodeBench.
Fonte: http://cbtestes.icomp.ufam.edu.br/

Na Figura 2.5 é possível ver o recurso de criação de um novo trabalho. Com esse recurso é possível importar exercícios de outros trabalhos, criados anteriormente em outras disciplinas.

Além de todos esses recursos, o CodeBench implementa um mecanismo de coleta de dados no momento em que os alunos codificam os exercícios utilizando a sua IDE. Esses dados podem ser utilizados em pesquisas que tenham o objetivo de fazer intervenções pedagógicas, mineração de dados, ciência de dados, dentre outras. Nesta pesquisa foram utilizados dados coletados pelo CodeBench de turmas de CS1 do ano letivo de 2018.

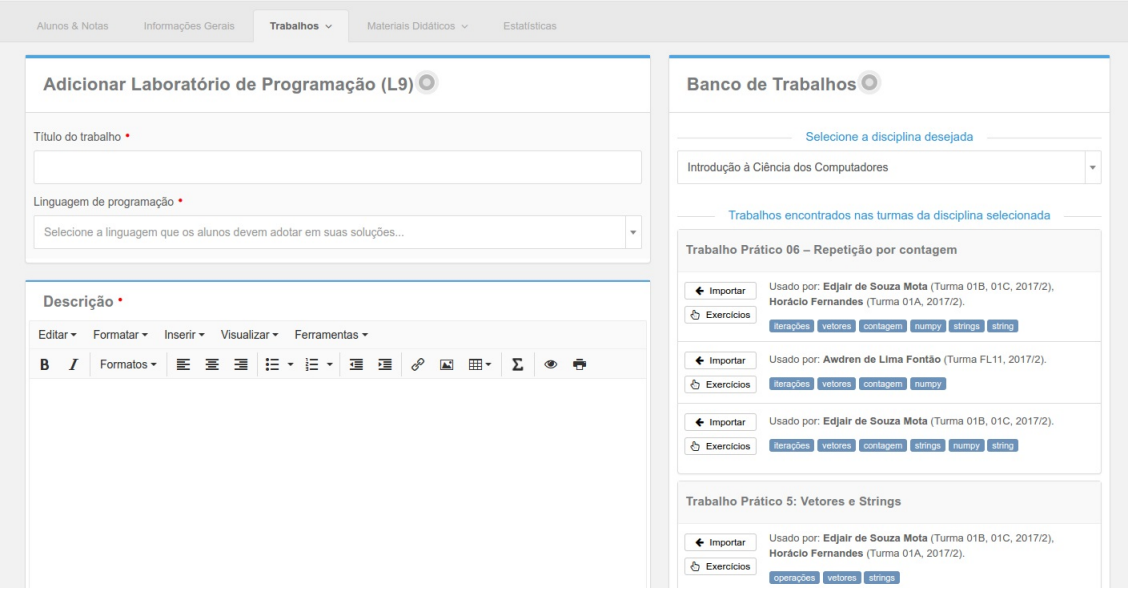


Figura 2.5 – Recurso de criação de novos trabalhos no ambiente do professor
Fonte: http://cbtestes.icomp.ufam.edu.br/

2.2 Sistemas de recomendação

Um problema atual para usuários de ambientes com grande quantidade de informação é a escolha de qual melhor opção para consumo. É necessário filtrar, priorizar e fornecer informações relevantes, de forma eficiente para aliviar o problema de sobrecarga de informações.

Os sistemas de recomendação solucionam esse transtorno, pesquisando por um grande volume de informações geradas dinamicamente, fornecendo aos usuários conteúdos e serviços personalizados [Isinkaye et al. 2015].

Os sistemas de recomendação melhoram a experiência do usuário no processo de seleção e compra de produtos no mercado eletrônico [Hu and Pu 2009]. Consequentemente, aumentam as receitas das empresas que utilizam essa nova forma de oferecer produtos aos usuários [Pu et al. 2011].

Grandes empresas utilizam sistemas de recomendação para fornecer conteúdo personalizado para seus usuários [Ziegler et al. 2005]. *Google*, *Facebook*, *Netflix* e *Amazon* são apenas alguns dos principais exemplos de grandes empresas já consolidadas no mercado que implementam recomendação em suas aplicações.

No entanto, os sistemas de recomendação não são utilizados somente no contexto empresarial e comercial. Atualmente, várias pesquisas estão sendo feitas utilizando sistemas de recomendação no contexto educacional. [Drachsler et al. 2015] realizaram um estudo

em que foram identificados 82 sistemas de recomendação de 35 países diferentes, todos esses sistemas são voltados para auxiliar o aluno no processo de aprendizagem.

[Laisa et al. 2018] fizeram um mapeamento sistemático da literatura sobre o uso de sistemas de recomendação educacionais, a fim de responder as seguintes perguntas:

- Quais tipos de recomendações estão sendo feitas na área educacional?

  Nesse caso, os Objetos de Aprendizagem são os recursos mais recomendados, seguidos por recomendação de livros, recomendação de cursos, recomendação de documentos, e por último, recomendação de vídeos educacionais.

- Quais as principais limitações e desafios encontrados na área de recomendação adotada pelos estudos?

  Aqui, as principais dificuldades listadas foram: recomendar conteúdos dinâmicos abertos de aprendizagem para milhares de alunos; fornecer aos alunos um caminho de aprendizagem adaptado, que lhes permita atingir as competências necessárias na aprendizagem; trabalhar questões afetivas em sistemas de recomendações educacionais; modo de avaliação e validação de questões das ferramentas educacionais; compartilhar conhecimentos práticos sobre problemas e projetos de engenharia.

- Quais as técnicas de recomendação estão sendo adotadas para implementar tais sistemas?

  Dentre as principais técnicas identificadas, encontram-se a filtragem colaborativa, filtragem híbrida e filtragem baseada em conhecimento e a filtragem baseada em conteúdo.

- Quais variáveis estão sendo consideradas por essas técnicas para se determinar cada tipo de recomendação?

  Muitas variáveis foram identificadas, sendo as principais: perfil do aluno; histórico do desempenho acadêmico; conhecimento prévio do aluno; histórico do navegador de internet e redes sociais; estado afetivo e emocional; registro de leitura.

- Quais são os níveis de ensino que estão sendo contemplados pelos sistemas de recomendação?

  O nível superior de ensino foi o mais utilizado, seguido pelo ensino médio e fundamental.

2.3 Etapas de funcionamento dos sistemas de recomendação

A seguir, listamos as principais etapas que um sistema de recomendação percorre para sugerir conteúdo aos usuários. A Figura 2.6 mostra as etapas de funcionamento de um sistema de recomendação.

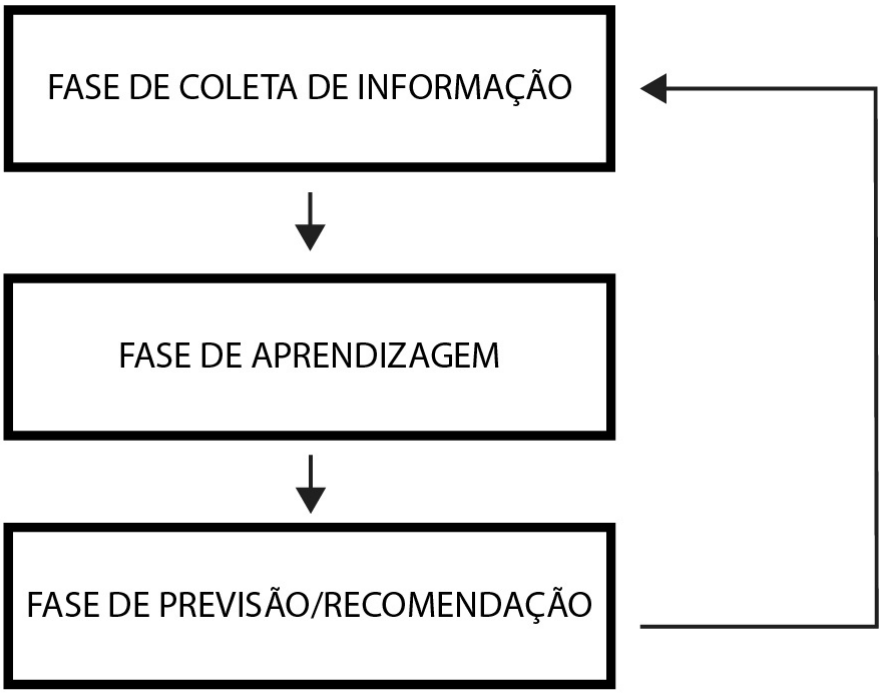


Figura 2.6 – Fases de recomendação de conteúdo.
Fonte: [Isinkaye et al. 2015] - Adaptado

- **Fases de coleta de informações**

  Esse é o primeiro passo executado pelos sistemas recomendadores. Nesse passo é feita uma coleta de informações relevantes sobre o usuário, para que seja criado um perfil ou modelo.

  Um agente recomendador não irá ter bom funcionamento caso o perfil/modelo do usuário não seja bem definido. O sistema precisa saber o máximo possível sobre o usuário, para fazer recomendações efetivas desde o início da utilização do sistema [Huang et al. 2014].

- **Fase de aprendizagem**

Nessa fase é feita a filtragem dos dados coletados na fase anterior. Para isso, um algoritmo de aprendizagem é aplicado sobre os dados do usuário [Isinkaye et al. 2015].

- **Fase de previsão/recomendação**

  Nessa fase é feita a recomendação ou predição dos conteúdos que o usuário pode preferir. Esse processo pode ser feito de acordo com o conjunto de dados coletados na fase de coleta de informações, com base em memória ou com base no perfil do usuário modelado pelo sistema [Isinkaye et al. 2015].

2.4 Técnicas de filtragem de recomendação

Para que as recomendações feitas pelos sistemas recomendadores sejam eficazes, é importante o uso de boas técnicas de recomendação na fase de aprendizagem do sistema de recomendação. A Figura 2.7 mostra as principais técnicas de recomendação citadas por [Isinkaye et al. 2015].

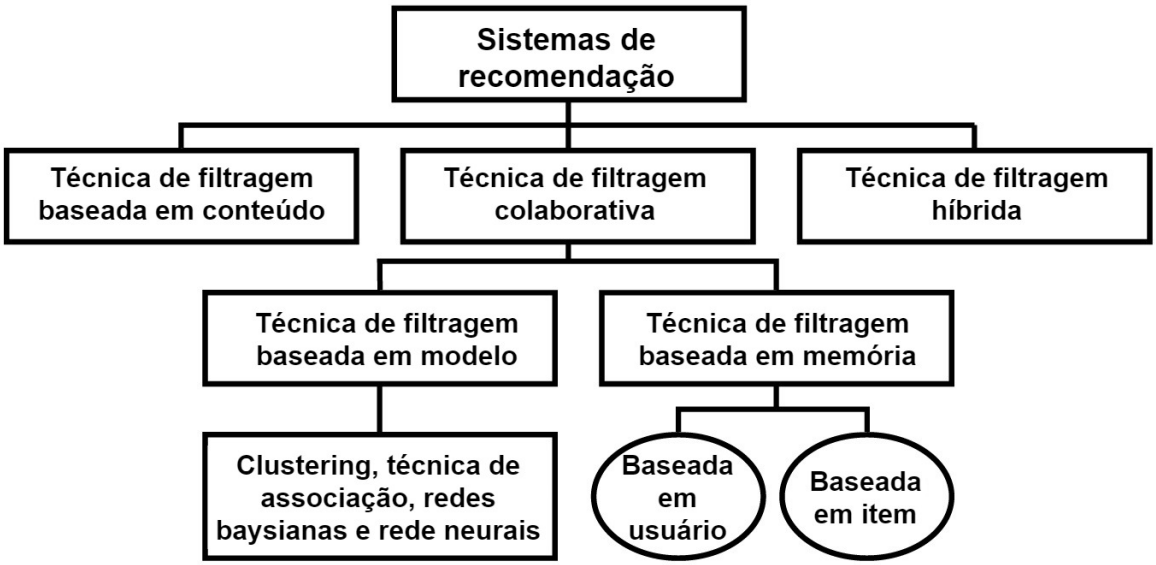


Figura 2.7 – Técnicas de Recomendação.
Fonte: [Isinkaye et al. 2015] - Adaptado

2.4.1 Filtragem baseada em conteúdo

A técnica de filtragem baseada em conteúdo analisa os dados dos itens consumidos no passado pelo usuário para gerar novas recomendações. Essa técnica funciona muito bem quando os itens recomendados são páginas da web ou notícias, por exemplo.

Os itens bem avaliados consumidos anteriormente pelo usuário são adicionados em seu perfil. Adiante, é recomendado para o usuário itens semelhantes aos existentes em seu perfil.

2.4.2 Filtragem colaborativa

A filtragem colaborativa utiliza o histórico de preferências do usuário para recomendar novos itens. Essas preferências podem ser compostas de *links* visitados ou produtos comprados.

O histórico de preferências do usuário é mantido com o objetivo de encontrar outros usuários semelhantes. Quando isso acontece, temos um relacionamento de interesses. O relacionamento de interesses pode ser determinado de duas formas: recomendação baseada no usuário ou baseada no item.

Na recomendação baseada no usuário, as recomendações são feitas através da relação entre usuários e sua forma de consumo. Essa relação é complexa, pois usuários possuem uma natureza dinâmica e podem mudar seus gostos e preferências no decorrer do tempo. Uma das principais estratégias para contornar a natureza dinâmica dos usuários é considerar um histórico recente de consumo.

A recomendação baseada em itens não tem características tão dinâmicas quanto na baseada em usuário. Por isso, pode ser calculada em intervalos maiores de tempo. Esse caso geralmente acontece quando existem mais usuários do que itens [Ricci et al. 2015].

2.4.2.1 Técnicas baseadas em memória

As técnicas de filtragem baseadas em memória são uma subdivisão da técnica de filtragem colaborativa. Nas técnicas de filtragem baseadas em memória os itens que já foram classificados anteriormente por algum usuário exercem um papel importante, pois eles podem ser recomendados para outros usuários semelhantes aos usuários que já classificaram os itens anteriormente. Para isso, algoritmos baseados em clusterização são aplicados a fim de identificar usuário similares.

As técnicas de recomendação baseada em memória podem ser divididas de duas formas [Isinkaye et al. 2015]: técnicas baseadas em usuário, que são aquelas que recomendam itens através de um cálculo que diz qual o nível de similaridade entre os usuários; e

as técnicas baseadas em itens, que calculam a similaridade dos itens para fazer recomendação para os usuários.

2.4.2.2 Técnicas baseadas em modelo

Essa técnica utiliza as classificações anteriores e, através de algoritmos de aprendizagem de máquina ou mineração de dados, constrói um modelo que aumenta o desempenho da técnica de filtragem colaborativa. Por usar um modelo pré-calculado, itens são recomendados com resultados semelhantes às técnicas que utilizam similaridade de usuários.

Algoritmos de mineração de regras de associação, *clustering*, redes neurais artificiais e redes bayesianas são algumas formas de montar o modelo para recomendação [Mobasher et al. 2004].

2.4.3 Filtragem híbrida

A filtragem híbrida possui esse nome porque combina várias técnicas de recomendação para melhorar o desempenho. Através da combinação de vários algoritmos, busca-se obter recomendações mais efetivas, pois as fraquezas de um algoritmo podem ser compensadas pelas vantagens de outro.

Basicamente, existem duas formas de fazer a combinação de algoritmos: na primeira forma, é utilizada a filtragem baseada em conteúdo na abordagem colaborativa; já na segunda, é feita a abordagem colaborativa na abordagem baseada em conteúdo. A união das duas abordagens gera a filtragem híbrida [Brusilovsky and Millán 2007].

2.5 Coeficiente de Kendall

Para fazer a validação das recomendações feitas neste trabalho, foi utilizado o coeficiente de Kendall para comparar a ordem de exercícios aplicada aos alunos com e sem a utilização do método aqui proposto.

O coeficiente de correlação de postos de Kendall, ou, coeficiente de Kendall, é uma estatística usada para medir a correlação de postos entre duas quantidades medidas, ou seja, verifica a semelhança entre as ordens dos dados quando classificados por cada uma das quantidades [KENDALL 1938].

Por definição o coeficiente de Kendall é calculado da seguinte maneira:

Dado (x_1, y_1), (x_2, y_2),...,(x_n, y_n) um conjunto de observações das variáveis X e Y respectivamente, e todos os valores pertencentes a (x_i) e (y_i) sejam únicos. Qualquer par de observações (x_i, y_i) e (x_j, y_j), em que $i \neq j$, é concordante se as classificações de ambos os elementos concordarem uma com a outra, isto é, se $x_i > x_j$ e $y_i > y_j$ ou se $x_i < x_j$ e $y_i < y_j$. Elas são discordantes se $x_i > x_j$ e $y_i < y_j$ ou se $x_i < x_j$ e $y_i > y_j$. Se $x_i = x_j$ ou $y_i = y_j$, o par não é nem concordante, nem discordante. O coeficiente τ de Kendall é definido pela Equação 2.1:

$$\tau = \frac{pc - pd}{tp(tp-1)/2} \tag{2.1}$$

Onde:

- *pc* é a quantidade de pares concordantes
- *pd* é a quantidade de pares discordantes
- *tp* é a quantidade total de pares

Assim, obtemos as seguintes propriedades:

- o denominador é o número total de combinações de pares, então, o coeficiente deve estar no intervalo $-1 \leq \tau \leq 1$.
- Se a concordância entre as duas classificações forem iguais, τ = 1.
- Se a discordância entre as duas classificações forem iguais, ou seja, uma classificação for o reverso da outra, τ = -1.

Neste trabalho para comparar uma lista de exercícios com outra lista de exercícios é aplicado o coeficiente de Kendall, τ, considerando a lista de exercício inicial como variável X e a lista de exercício modificada como variável Y.

2.5.1 Considerações finais

Neste capítulo foram apresentados conceitos teóricos básicos relacionados a esta pesquisa. Inicialmente foi feita uma breve explanação sobre juízes online, que podem ser usados como ferramentas de auxílio ao professor e ao aluno no processo de ensino e aprendizagem de algoritmos e linguagens de programação.

Alguns juízes online possuem em seu repositório milhares de exercícios, o que pode causar uma sobrecarga de informação para o usuário desses sistemas. É desafiador selecionar a ordem adequada para o aluno resolver, sendo assim, a integração de sistemas de recomendação aos juízes online pode ser benéfica para os usuários.

Existem diversos sistemas de recomendação aplicados em várias áreas, por exemplo: marketing, negócios na web, entretenimento digital e recentemente o uso no contexto educacional. Para cada área em que o sistema de recomendação é utilizado, existem formas e técnicas diferentes para fazer a coleta de informação do usuário, a aprendizagem e recomendação.

Nesta pesquisa é utilizada a técnica de filtragem colaborativa para fazer a recomendação de exercícios para os alunos. As recomendações são feitas de acordo com o índice de dificuldade de cada exercício resolvido anteriormente pelos alunos. Para isso, os dados sobre os índices de dificuldade dos exercícios são armazenados em uma estrutura de dados, denominada Perfil de Programação. Para avaliar a qualidade das recomendações, utilizando o coeficiente de Kendall, é comparado os índices de dificuldades dos exercícios revolvidos pelos alunos com e sem a utilização do método proposto.

Capítulo 3

Trabalhos Relacionados

Após a delimitação do tema da pesquisa, definição de objetivos e questões de pesquisas, elaborou-se uma Revisão Sistemática da Literatura (RSL) seguindo as orientações gerais propostas por [Kitchenham 2004], com intuito de identificar o estado da arte relacionado a métodos de recomendação de conteúdo para disciplinas de introdução a programação em ambientes virtuais de aprendizagem ou juízes online. Nesse sentido, serão apresentados os resultados da RSL e os principais trabalhos encontrados nela, que estão fortemente relacionados a este trabalho.

3.1 Revisão Sistemática da Literatura

Esta RSL foi feita em três etapas: planejamento, execução e resultados. A Figura 3.1 ilustra as etapas desta RSL.

3.1.1 Planejamento

A RSL conduzida teve por intento investigar e analisar pesquisas que modelam o comportamento de estudantes de programação, utilizando dados coletados em juízes online ou ambientes virtuais de aprendizagem, a fim de recomendar conteúdos relativos a disciplinas CS1. Especificamente, procurou-se por métodos de filtragem de dados, técnicas de recomendação de conteúdos e métricas para validação de sistemas de recomendação.

Foram estabelecidos critérios para a escolha das bibliotecas digitais para a condução da busca por produções científicas relacionadas ao tema aqui exposto e, além disso, foram pesquisados títulos no idioma inglês. A pesquisa foi realizada nas bibliotecas digitais:

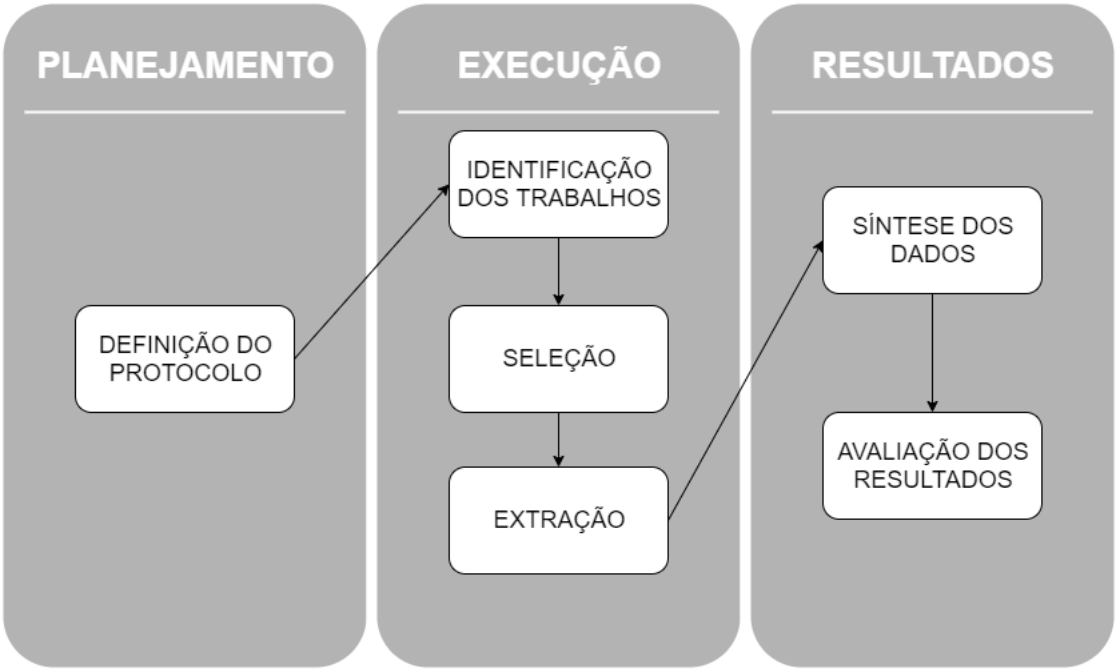


Figura 3.1 – Etapas da RSL
Fonte: Adaptado de [Francisco et al. 2016].

Scopus[1], ACM[2] e IEEE Xplore[3]. Destaca-se que a Scopus indexa publicações de muitos periódicos como, por exemplo: Springer, ScienceDirect/Elsevier e British Computer Society. Outra ferramenta utilizada foi o Start versão 3.3[4], desenvolvido pelo Laboratório de Engenharia de Software (LAPES) da Universidade de São Carlos (UFSCAR), que é específico para dar suporte ao processo da RSL. O software Mendeley Desktop[5] 1.17.13 foi utilizado para a catalogação, leitura e compartilhamento dos artigos.

Para atingir o objetivo, questões de pesquisa foram estabelecidas e respondidas para cada artigo na fase de extração dos dados. A seguir, são apresentadas as questões:

Q1: Como recomendar questões de forma individualizada para alunos de disciplinas CS1 em um ambiente de correção automática de códigos, utilizando um perfil de programação baseado nos *logs* de exercícios resolvidos?

Essa questão foi dividida em outras subquestões, listadas abaixo. A subdivisão foi feita para facilitar a extração dos dados dos trabalhos selecionados nesta RSL.

Q1-a. O trabalho aborda a recomendação de conteúdo?

[1]http://www.scopus.com
[2]https://dl.acm.org/
[3]http://ieeexplore.ieee.org
[4]Disponível para download em http://lapes.dc.ufscar.br/tools/start_tool
[5]Disponível para download em https://www.mendeley.com/download-desktop-new/

Q1-b. Que tipo de conteúdo é recomendado?

Q1-c. A recomendação é destinada a que grupo de usuários?

Q1-d. Em que ambiente os conteúdos são recomendados?

Q1-e. Se perfis são criados com vistas à recomendação, que base de dados é utilizada na criação dos perfis?

Q1-f. Que técnicas são utilizadas pelos ambientes de correção automática de códigos para recomendação de conteúdo?

Q2: Quais métodos são utilizados pelos sistemas de recomendação de conteúdo?

Q3: Quais métricas são utilizadas para validar a eficiência e a eficácia de um sistema de recomendação?

Com o objetivo de responder as questões Q1, Q2 e Q3 elaborou-se uma string de busca, que foi norteada com palavras chaves extraídas de artigos de controle obtidos através de uma busca exploratória.

Para a realização da busca dos artigos nas bases científicas, foi necessária a construção de uma *String* de busca que retornasse trabalhos relevantes e adequados para essa RSL. A *String* de busca utilizou na sua formulação operadores booleanos (OR, AND), além de *wildcards*. A Tabela 3.1 mostra a *string* de busca da RSL.

Tabela 3.1 – String de Busca da RSL

(virtual learning environment OR e-learn* OR online judge) AND recommend* AND (program* OR code)

3.1.2 Execução

O processo de busca foi realizado em 10/12/2018. O quantitativo de trabalhos que entraram no 1º filtro da fase de seleção é dado na Tabela 3.2, isto é, a quantidade de resultados das buscas já removidos os resultados duplicados. Além disso, é mostrado na tabela o quantitativo de artigos relevantes que passaram do 1º filtro e que foram selecionados conforme os critérios de inclusão.

Tabela 3.2 – Sumário dos resultados da busca

Base de dados	**Artigos encontrados**	**Artigos aceitos na fase de seleção**	**Artigos aceitos na fase de extração**
Scopus	189	66	10
ACM	1328	41	4
IEEE	120	5	5
Totais	**1637**	**112**	**19**

3.1.2.1 Seleção dos Estudos

Os artigos que foram selecionados para a revisão de literatura obedeceram aos seguintes critérios:

- Inclusão dos artigos **(I)**

I-a. Artigos que abordam sistemas de recomendação;

I-b. Artigos que incluem uma das palavras chaves da *string* de busca no resumo, título ou lista de palavras chaves;

I-c. Artigos publicados e disponíveis integralmente em bases de dados científicas;

- Exclusão dos artigos **(E)**

E-a. Artigos que não tratam de recomendação de conteúdo educacional na área de programação;

E-b. Artigos que não abordam recomendação de conteúdo como foco principal da pesquisa;

E-c. Artigos que não apresentam os métodos utilizados para recomendação de conteúdo;

E-d. Trabalhos publicados como artigos curtos ou pôsteres;

E-e. Capítulos de livros, teses e dissertações;

E-f. Revisões da literatura e *surveys*;

Processo de seleção: Nessa fase, foram lidos os títulos, lista de palavras-chave e resumo dos trabalhos. Depois de aplicar os critérios de inclusão e exclusão, a qualidade

dos resultados de busca dos artigos foi aprimorada, visto que artigos que não faziam parte do objetivo dessa revisão sistemática da literatura foram removidos. As informações de referências foram importadas das fontes de pesquisa e salvas.

Para os artigos que foram selecionados como relevantes, uma cópia digital do artigo foi armazenada em repositório próprio para posterior leitura. Como passo seguinte, as cópias digitais catalogadas foram analisadas em detalhes. Esse passo final foi útil para averiguar se havia algum artigo selecionado que se enquadrava ainda em alguns dos critérios de exclusão mencionados anteriormente.

Após o processo de seleção final, 93 artigos foram excluídos. Os 19 artigos restantes, que foram aprovados para essa revisão, são listados na Tabela 3.3. Assim, foram obtidas 19 publicações bem relacionadas com o tema proposto desta pesquisa. A Figura 3.2 mostra a distribuição temporal dos artigos selecionados para esta RSL.

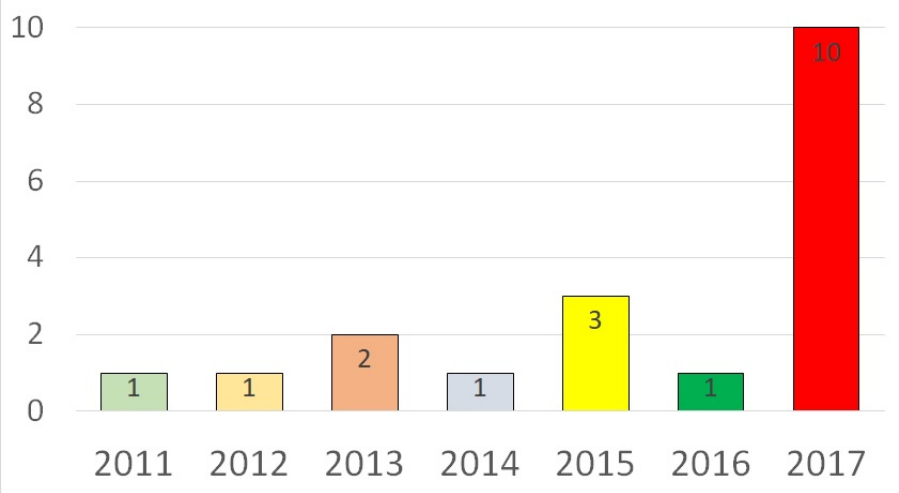


Figura 3.2 – Distribuição temporal das publicações aceitas no segundo filtro da RSL.
Fonte: Própria.

Tabela 3.3 – Artigos selecionados após 2º filtro

ID	Artigo
S01	Enhancing e-learning systems with personalized recommendation based on collaborative tagging techniques [Klašnja-Milićević et al. 2017]

Continua na próxima página

Tabela 3.3 – Artigos selecionados após 2º filtro

S02	A study of concept-based similarity approaches for recommending program examples [Hosseini and Brusilovsky 2017]
S03	Content wizard: Concept-based recommender system for instructors of programming courses [Chau et al. 2017]
S04	A method for personalized c programming learning contents recommendation to enhance traditional instruction [Yan et al. 2017]
S05	A recommendation system to support the students performance in programming contests [De Paula et al. 2015]
S06	Mining an online judge system to support introductory computer programming teaching [Francisco and Ambrosio 2015]
S07	Applying recommender systems and adaptive hypermedia for e-learning personalization [Vesin et al. 2013]
S08	Recommendation of programming activities by multi-label classification for a formative assessment of students [De Oliveira et al. 2013]
S09	Protus 2.0: Ontology-based semantic recommendation in programming tutoring system [Vesin et al. 2012]
S10	A recommendation approach for programming online judges supported by data preprocessing techniqes [Yera and Martínez 2017]
S11	Rule-based reasoning for altering pattern navigation in Programming Tutoring System [Vesin et al. 2011]
S12	CodERS: A hybrid recommender system for an E-learning system [Ansari et al. 2017]

Continua na próxima página

Tabela 3.3 – Artigos selecionados após 2º filtro

S13	Research on three-layer collaborative filtering recommendation for Online Judge [Yu and Chen 2016]
S14	Real-time Learning Analytics for C Programming Language Courses [Fu et al. 2017]
S15	An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges [Toledo 2014]
S16	Case-Based Recommendation for Online Judges Using Learning Itineraries [Sánchez-Ruiz et al. 2017]
S17	Classification and Recommendation of Competitive Programming Problems Using CNN [Sudha et al. 2017]
S18	Similar users or similar items? Comparing similarity-based approaches for recommender systems in online judges [Caro-Martinez and Jimenez-Diaz 2017]
S19	The research of the recommendation algorithm in online learning [Yu et al. 2015]

3.1.2.2 Extração de Dados

Após o conjunto final de estudos primários terem sidos definidos e suas qualidades avaliadas, a fase de extração de dados do processo de Revisão Sistemática da Literatura foi realizada. A seção a seguir especifica o formulário de extração de dados criado para registrar dados e detalha a estratégia para usá-los.

3.1.2.3 Formulário de extração de dados

Um formulário de extração de dados foi projetado para registrar todas as informações relevantes dos estudos primários necessários para responder às perguntas da pesquisa. A Tabela 3.4 representa o Formulário de Extração de Dados usado para essa RSL.

Tabela 3.4 – Formulário de Extração de Dados

Informações sobre o artigo		
ID	Identificador único no formato: S<número>	
Título		
Ano de Publicação		
Publicador		
Informações relevantes para responder as questões de pesquisa		
Q1-a	O trabalho aborda recomendação de conteúdo?	( ) Sim ( ) Não
Q1-b	Que tipo de conteúdo é recomendado?	( ) Exercícios ( ) Conteúdo na Web ( ) Material didático ( ) Códigos de Linguagem de programação
Q1-c	A recomendação é destinada a que grupo de usuários?	( ) Alunos ( ) Professores
Q1-d	Em que ambientes os conteúdos são recomendados?	( ) Juiz online ( ) AVA ( ) Sistema próprio desenvolvido na pesquisa
Continua na próxima página		

Tabela 3.4 – Formulário de Extração de Dados

Q1-e	Se perfis são criados com vistas à recomendação, que base de dados é utilizada na criação dos perfis?	( ) Logs de exercícios/Códigos ( ) Questionário em formulário ( ) Logs de busca na Web
Q2	Quais métodos são utilizados pelos sistemas de recomendação de conteúdo?	( ) Filtragem colaborativa ( ) Marcação colaborativa de Tags ( ) Baseada em conceito
Q3	Quais métricas são utilizadas para validar a eficiência e eficácia de um sistema de recomendação?	( ) FolkRank ( ) Expressividade da Etiqueta ( ) Recall ( ) Clássica de Precisão ( ) Comparação com opinião de humanos ( ) F1 Score ( ) Hamming Loss ( ) One-Error

3.1.3 Resultados

A fase de síntese de dados do processo de Revisão Sistemática da Literatura envolve a compilação dos dados extraídos dos estudos primários de modo a abordar cada uma das questões da pesquisa.

Os dados sintetizados para cada pergunta foram tabulados facilitando qualquer análise futura necessária e os dados extraídos dos estudos foram compilados e dirigidos para cada questão de pesquisa.

3.1.3.1 Questão de pesquisa 1

A questão 1 busca responder como recomendar questões de forma individualizada para alunos de disciplinas de CS1 em um ambiente de correção automática de códigos, utilizando um perfil de programação baseado nos *logs* de exercícios resolvidos.

A referida questão foi subdividida em várias outras questões que terão os resultados detalhados a seguir:

Q1-a: O trabalho aborda a recomendação de conteúdo?

Todos os 19 trabalhos selecionados abordam recomendação de conteúdo.

Q1-b: Que tipo de conteúdo é recomendado?

A Figura 3.3 mostra que a grande maioria dos trabalhos selecionados fazem recomendação de exercícios voltados para programação de computadores (S05, S06, S07, S08, S09, S10, S11, S12, S13, S15, S16, S17, S18 e S19), seguidos por recomendação de material didático (S01, S03, S04, S07, S09, S11 e S14). Somente o artigo S02 faz recomendação de códigos de linguagem de programação.

Q1-c: A recomendação é destinada a que grupo de usuários?

A Figura 3.4 mostra que em 18 artigos selecionados (S01, S02, S04, S05, S06, S07, S08, S09, S10, S11, S12, S13, S14, S15, S16, S17, S18 e S19) a recomendação é destinada aos alunos, apenas 2 trabalhos (S03 e S14) fazem a recomendação para professores. É importante frisar, que em um único trabalho (S14) a recomendação é feita para alunos e professores.

Isso mostra que o desenvolvimento de ambientes que recomendam itens personalizados para o aluno é uma tendência e que esta pesquisa está alinhada com outras pesquisas da área. Já que pretende-se recomendar exercícios personalizados para alunos que estão estudando conceitos sobre linguagem de programação e algoritmos.

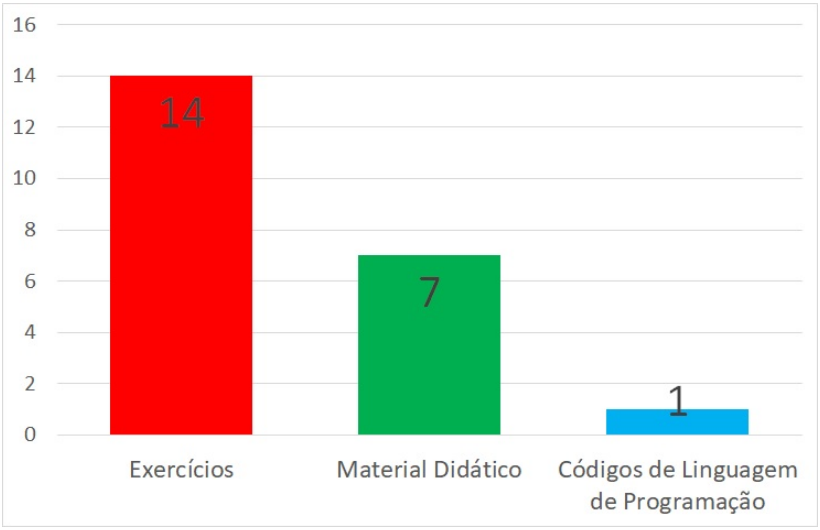


Figura 3.3 – Tipo de conteúdo recomendado.
Fonte: Própria

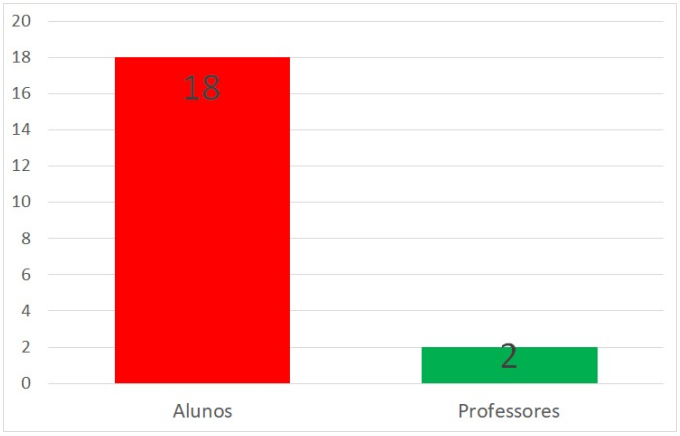


Figura 3.4 – Grupo de usuário alvo da recomendação.
Fonte: Própria

Q1-d: Em que ambiente os conteúdos são recomendados?

A Figura 3.5 mostra que os principais ambientes utilizados para fazer recomendação de conteúdo educacional para aprendizagem de programação são os Ambientes Virtuais de Aprendizagem (AVA) e os juízes online.

Em dez trabalhos (S05, S06, S08, S10, S13, S15, S16, S17, S18 e S19), as recomendações de conteúdos são feitas em juízes online e em nove trabalhos (S01, S02, S03, S04, S07, S09, S11, S12 e S14), as recomendações são feitas em AVAs.

Para os AVAs o mais utilizado nas pesquisas é o *Moodle*[6]. Os outros AVAs utilizados são ambientes desenvolvidos pelos próprios autores dos trabalhos.

[6]https://moodle.org/

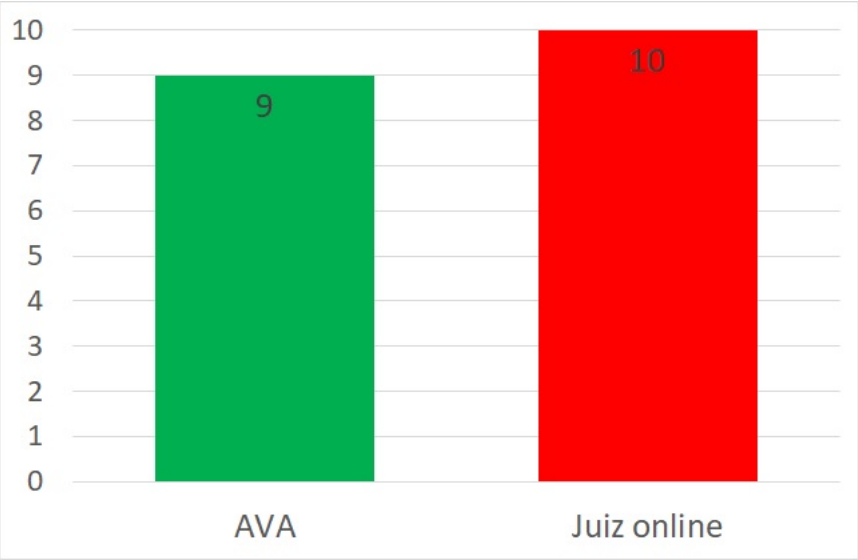


Figura 3.5 – Ambientes em que são feitas as recomendações.
Fonte: Própria

Nas pesquisas que recomendam conteúdo em juízes online, na maioria dos casos os pesquisadores utilizam juízes onlines que disponibilizam a base de dados para estudos.

Q1-e: Se perfis são criados com vistas à recomendação, que base de dados é utilizada na criação dos perfis?

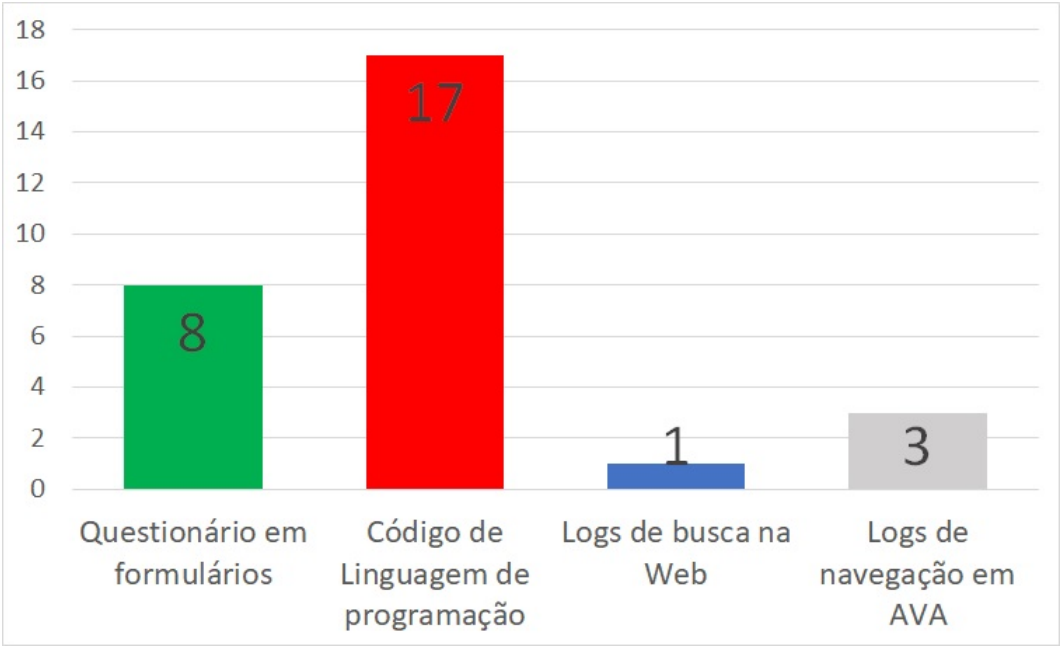


Figura 3.6 – Fonte de base de dados utilizada para criação de perfil para recomendação de conteúdo.
Fonte: Própria

A Figura 3.6 é de muita importância para este trabalho, pois responde uma das perguntas de maior relevância para esta pesquisa: qual a fonte dos dados utilizados para criação dos perfis para recomendação de conteúdo?

Aqui, é possível notar que a maioria dos trabalhos (S03, S04, S05, S06, S07, S08, S09, S10, S11, S12, S13, S14, S15, S16, S17, S18 e S19) utilizam códigos de linguagem de programação para criar o perfil do usuário alvo da recomendação. Sendo que no caso dos juízes online, todos utilizam códigos de programação para fazer a recomendação. O trabalho S06, além de utilizar códigos de programação, utiliza questionários em formulários para identificar o conhecimento prévio das habilidades de programação do aluno.

Um total de oito trabalhos (S01, S02, S03, S06, S07, S09, S11 e S12) utilizam questionários em formulário para criar o perfil do usuário. Porém, nesse caso, a grande maioria desses trabalhos recomendam itens em AVAs.

Três trabalhos (S07, S09 e S11) utilizam *logs* de navegação de AVA para auxiliar na montagem do perfil do usuário. É importante frisar que, nesses trabalhos, também são utilizados outros dados conforme já mostrado anteriormente.

Finalmente, o único trabalho que utiliza *logs* de busca na *web* para fazer recomendações de itens é o S07, que também utilizou outros tipos de dados para montar o perfil do usuário.

Sendo assim, é possível notar que a base de dados mais adequada para se recomendar conteúdo dentro de juízes online são os códigos submetidos pelos usuários. Neste trabalho, serão coletados os dados que possuem indícios sobre a dificuldade do aluno enquanto ele utiliza o ambiente integrado de desenvolvimento de um juiz online para montar o perfil de programação. Esses dados serão extraídos dos códigos submetidos pelo aluno e dados sobre seu comportamento enquanto utiliza o juiz online.

3.1.3.2 Questão de pesquisa 2

A questão 2 busca responder quais métodos são utilizados pelos sistemas de recomendação de conteúdo?

A Figura 3.7 mostra os métodos utilizados nos trabalhos selecionados para recomendar conteúdo aos usuários. O método mais utilizado é a filtragem colaborativa (utilizada nos trabalhos S01, S05, S06, S07, S08, S09, S10, S11, S13, S14, S15, S18 e S19). Em resumo, a filtragem colaborativa relaciona os usuários, verifica quais características esses

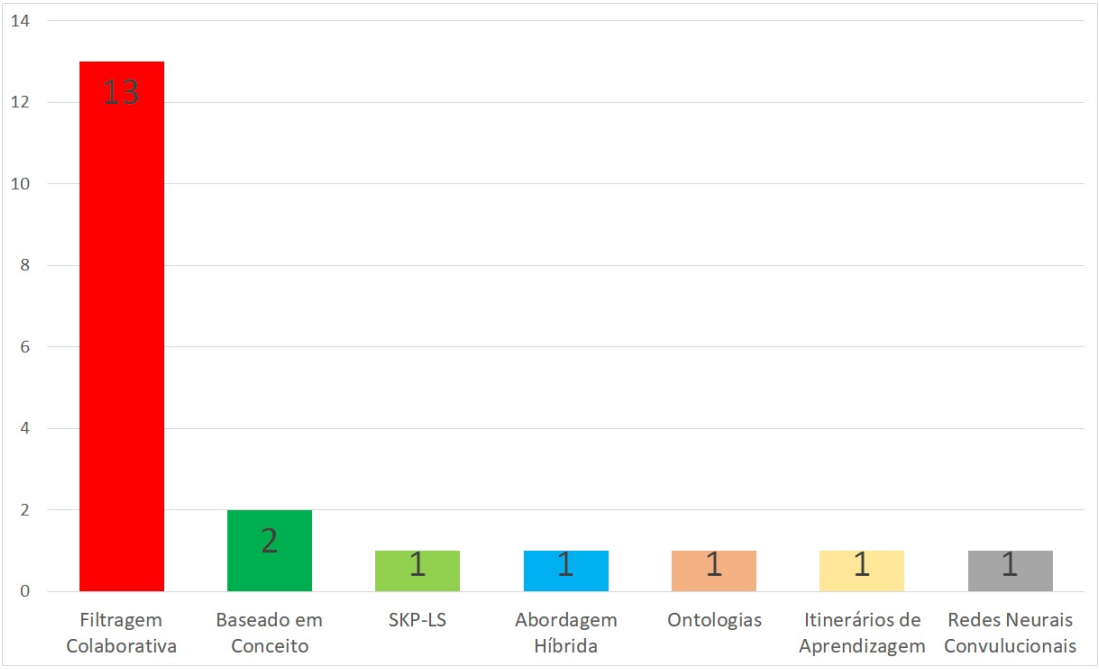


Figura 3.7 – Métodos utilizados para recomendação de conteúdo.
Fonte: Própria

usuários possuem em comum e recomenda itens para esses usuários de acordo com essas semelhanças.

É importante frisar que, somente em dois trabalhos (S16 e S17) que recomendam conteúdos em juiz online a filtragem colaborativa não é utilizada.

O trabalho S16 utiliza a recomendação baseada em itinerário de aprendizagem. Para isso, os autores consideram os exercícios resolvidos com um nó em um grafo, e as arestas são a sequência de exercícios resolvidos. A análise de similaridade de grafos é feita com métricas especificas de similaridade entre grafo. Depois, é verificado quais exercícios o aluno alvo da recomendação não possui no grafo, então é recomendada a sequência para o aluno em questão.

Já o trabalho S17 usa Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks* (CNN), para classificar códigos de programas do juiz online CodeForces[7]. Os códigos são mapeados em classes de assuntos (árvores, busca binária, grafos e programação dinâmica). Depois, quando um aluno responde um exercício, a CNN classifica o exercício resolvido em uma classe de assunto, calcula a distância euclidiana entre os outros exercícios da mesma classe, e recomenda exercícios semelhantes ao resolvido pelo aluno.

[7]https://codeforces.com/

Os outros métodos, SKP-LS (utilizado no trabalho S04), abordagem híbrida (utilizado no trabalho S12), ontologias (utilizado no trabalho S14) e baseado em conceito (utilizado no trabalho S02), são todos utilizados em AVAs. Sendo assim, é possível notar que, no caso dos juízes online, o método mais adequado para recomendar itens é a filtragem colaborativa.

3.1.3.3 Questão de pesquisa 3

A questão 3 busca responder quais métricas são utilizadas para validar a eficiência e eficácia de um sistema de recomendação?

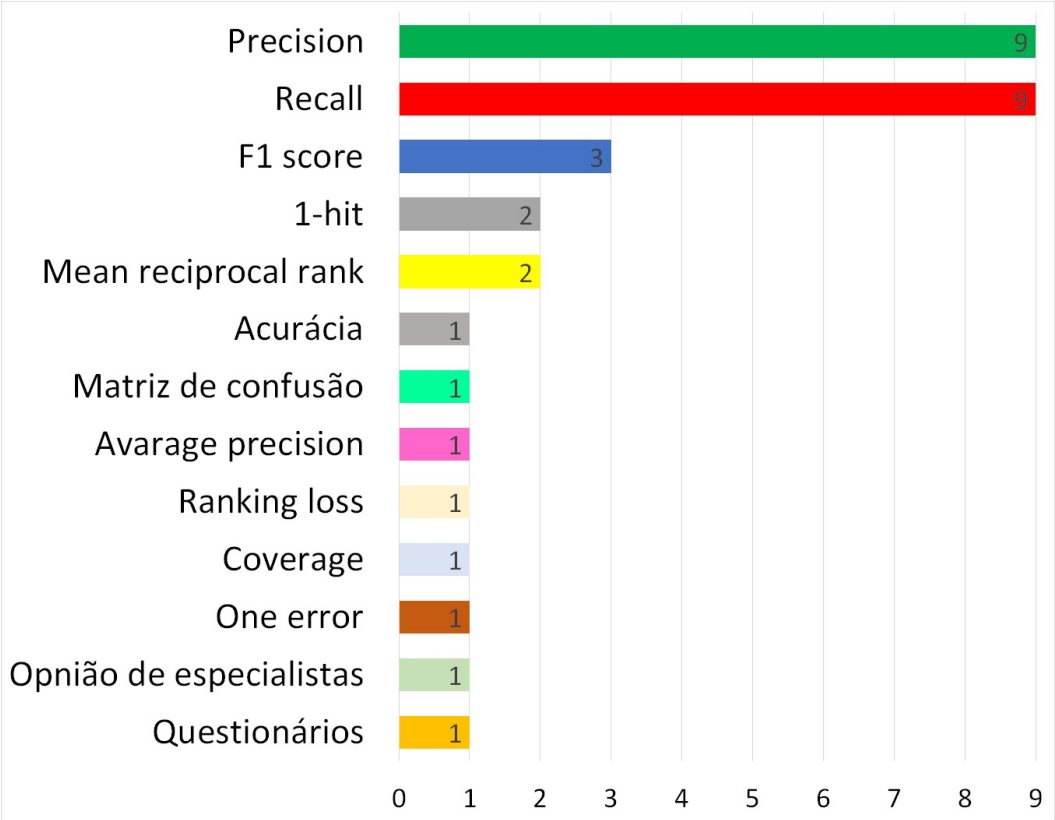


Figura 3.8 – Métricas para validação de Sistemas de Recomendação.
Fonte: Própria

A Figura 3.8 mostra que os sistemas de recomendação são avaliados por diversas métricas simultaneamente. Vários trabalhos utilizaram mais de um método para avaliar as recomendações feitas.

Outro dado importante da Figura 3.8 é que as métricas mais utilizadas são Recall e Precision. Somente o trabalho S08 utiliza a métrica de Precision e não utiliza a métrica de Recall como complemento. Todos os outros que utilizam Precision, também utilizam

Recall. Isso acontece pois os dados avaliados pela métrica de Recall são complementados pela métrica de Precision.

A seguir, serão listadas as métricas utilizadas em cada trabalho:

- Precision - número de recomendações corretas dividido pelo número total de recomendações feitas. Utilizada nos artigos S01, S03, S05, S08, S14, S15, S16, S17 e S18;
- Recall - número de recomendações corretas dividido pelo número de resultados que deveriam ter sido recomendados. Utilizada nos artigos S01, S03, S05, S14, S15, S16, S17, S18 e S19;
- F1 Score - média harmônica entre Precision e Recall. Utilizada nos artigos S05, S16 e S18;
- Avarage precision - valor médio de Precision para valores de Recall acima de 0 a 1. Utilizado no artigo S08.
- Mean Reciprocal Rank - medida para avaliar sistemas que retorna uma lista classificada de respostas às consultas. Utilizada nos artigos S16 e S18;
- 1-hit - número de recomendações dividido pelo número de usuários que "consumiram" os itens recomendados. Utilizada nos artigos S16 e S18;
- Questionários - aplicação de questionários de satisfação sobre as recomendações. Utilizado no artigo S12;
- Comparação com opinião de especialistas humanos - pessoas com conhecimento na área verificam a qualidade das recomendações manualmente. Utilizada no artigo S02;
- One Error - avalia a quantidade de vezes que uma recomendação importante não aparece no conjunto de recomendações. Quanto menor o valor de One Erro, melhor é o desempenho do sistema de recomendação. Utilizado no artigo S08;
- Coverage - avalia até que ponto deve-se descer a classificação do classificador das classes recomendadas para cobrir todas as classes pertinentes de uma instância. Utilizada no artigo S08;

- Ranking Loss - avalia quantas vezes no ranking de recomendações feitas a um usuário, as recomendações pertinentes foram definidas como não pertinentes. Utilizado no artigo S08;
- Matriz de confusão - uma tabela que mostra as frequências de classificação para cada classe do modelo. Utilizada no artigo S17;
- Acurácia - razão entre o número de recomendações corretas e o número total de amostras de entrada. Utilizada no artigo S17.

Para validação das recomendações feitas neste trabalho, serão feitas comparações das recomendações para os alunos, a fim de verificar a semelhança entre as listas de exercícios. Uma vez que os alunos são diferentes, as recomendações também devem ser diferentes. Outro dado analisado será o índice de dificuldade dos exercícios recomendados pelo método. Aqui, o objetivo é verificar se a ordem dos exercícios mantém um nível de dificuldade crescente.

3.2 Estado da arte

Nesta seção serão apresentados resumidamente os principais trabalhos desta RSL que estão correlacionados com esta pesquisa. Os trabalhos a seguir utilizam dados coletados em juízes online e ambientes virtuais de aprendizagem a fim de recomendar conteúdo (exercícios, materiais didáticos dentre outros) para melhorar a experiência de aprendizagem do usuário.

Esses trabalhos usam filtragem colaborativa, abordagem Baseada em Conceito, ontologias, método SKP-LS, similaridade de conteúdo e abordagem Híbrida para fazer recomendação.

3.2.1 Recomendação de conteúdo com abordagem de filtragem colaborativa

Em sistemas de recomendação baseados em filtragem colaborativa é feito um cálculo para identificar usuários com preferências similares ao usuário para o qual deseja-se fazer a recomendação. Logo após, são verificados quais itens foram consumidos pelos usuários

similares ao usuário alvo da recomendação que não foram consumidos pelo usuário alvo. Os itens mais bem avaliados pelos usuários semelhantes são então recomendados para o usuário alvo.

Nesse sentido, no trabalho S06, [Francisco and Ambrosio 2015] desenvolveram um sistema de tutoria de programação adaptável e inteligente baseado na web - Protus. O Protus aplica técnicas de recomendação e hipermídia adaptativa. Esse sistema visa orientar automaticamente as atividades do aluno e recomendar a ele links e ações relevantes durante o processo de aprendizagem.

Para isso, é criado um perfil do aluno, que utiliza dados pessoais, dados da performance do aluno e um histórico de aprendizagem. Uma vez que isso é feito, os alunos são agrupados por semelhança (clustering), então é recomendado um "caminho" a percorrer no curso.

No trabalho S10, [Yera and Martínez 2017] fazem recomendação de atividades de programação em juízes online usando filtragem colaborativa. Para isso, é construído um perfil do aluno, que basicamente é uma matriz com dados binários contendo as questões feitas pelo aluno (0 não resolveu, 1 resolveu) e a quantidade de tentativa para cada questão.

Em todos os trabalhos que usam filtragem colaborativa, é construído um perfil do aluno, seja ele contendo dados extraídos da resolução de exercícios, dados de navegação, questionários em formulários ou *logs* de busca na *web*.

O perfil do aluno é um item de grande importância na abordagem de filtragem colaborativa, pois é com ele que conseguimos identificar as preferências do usuário e com base nisso fazer o cálculo de similaridade de usuário, para assim, finalmente fazer a recomendação.

3.2.2 Recomendação de conteúdo com abordagem baseada em conceito

No trabalho S03, [Chau et al. 2017] utilizaram a abordagem baseada em conceito para recomendar conteúdo para professores estruturar cursos online de programação. A ideia do sistema é deduzir a estrutura dos cursos online analisando exemplos de código de programação apresentados em cada unidade pelo professor. Para isso, três passos são executados:

Primeiro, o professor cria uma unidade de aprendizagem, e nessa unidade ele faz o *upload* do conjunto de códigos de linguagem de programação (JAVA) que serão utilizados. Após isso, o sistema extrai automaticamente os conceitos associados a cada código (classes, herança, polimorfismo, etc.) e finalmente, um módulo de recomendação sugere ao professor diversos conteúdos relativos aos conceitos extraídos.

É importante frisar que, nessa pesquisa, o *upload* de códigos de linguagem de programação é feito de forma *offline*. Ou autores sugerem para que em trabalhos futuros seja feita a coleta de códigos de programação *online* de alunos e professores, assim a recomendação pode ser atualizada constantemente.

3.2.3 Recomendação de conteúdo com uso de Ontologias

No trabalho S11, através do uso de ontologias, [Vesin et al. 2011] criaram um curso personalizado de Java para os alunos. Nesse curso, são recomendados exercícios e material didático em um Ambiente Virtual de Aprendizagem.

Para fazer a personalização é criado o perfil do aluno através da coleta de dados explícita (formulários) e implícita (ações no AVA, tais como navegação, tempo em determinado assunto, logs de busca na WEB e etc). O uso de ontologias e linguagens de regras SWRL (Semantic Web Rule Language) para construir o modelo do aprendiz no sistema de tutoria Java foi criado através do Software Protegé.

3.2.4 Recomendação de conteúdo pelo método SKP-LS

No trabalho S04, [Yan et al. 2017] propõe um método intitulado de descrição do status de aprendizado do aluno baseado em SKP (Syntax Knowledge Point). Em primeiro lugar, é reunido todo conhecimento sintático que deve ser aprendido pelos alunos, extraindo o SKP de códigos-fonte em materiais didáticos ou respostas de modelos de exercícios.

Em seguida, para cada aluno, é coletado o SKP dos códigos fonte que são submetidos nas resoluções de exercícios em um ambiente virtual de aprendizagem. Finalmente, para cada aluno, a compreensão de cada SKP é estimada com base nos dados coletados e o status de aprendizado do aluno pode ser descrito pelo seu entendimento de todos os SKPs. Após o processamento dos dados de aprendizagem do aluno, são feitas recomendações somente para assuntos onde o SKP foi mais baixo.

3.2.5 Recomendação de conteúdo com abordagem de similaridade de conteúdo

No trabalho S05, [De Paula et al. 2015] fazem recomendação de exercícios de programação em juiz online utilizando a abordagem de similaridade de conteúdo. Nesse trabalho, os exercícios são classificados por assuntos abordados (estruturas condicionais, estruturas de repetição, vetores, matrizes, listas, filas, pilhas, ordenação, árvores, busca, grafos e combinatórios) e categorias (iniciante, *ad hoc*, strings, estruturas, matemática, paradigmas de programação, grafos e geometria computacional).

O perfil do usuário é montado de acordo com os exercícios que o usuário resolve. Sendo assim, as recomendações são feitas com base em exercícios de assuntos e categorias semelhantes resolvidos anteriormente pelo usuário. É importante frisar, que nesse trabalho específico, os autores sugerem que um trabalho futuro é fazer a recomendação de conteúdo com a abordagem de filtragem colaborativa.

3.2.6 Recomendação de conteúdo com abordagem híbrida

No trabalho S08, [De Oliveira et al. 2013] fazem recomendação utilizando a abordagem de similaridade de conteúdo e abordagem de filtragem colaborativa (a união dessas duas abordagens é dita Abordagem Híbrida). Nesse caso, é feita a criação do perfil do aluno, onde são registrados vários atributos utilizados para resolver os exercícios (estruturas de decisão e repetição, comandos de entrada e saída, funções, operadores aritméticos e lógicos dentre outros).

Nesse trabalho também é feita a classificação de classes de atividades por assunto (entrada e saída de dados, estruturas, repetição, comparação e etc). Para fazer a recomendação é levado em conta à similaridade de usuários e à similaridade das questões resolvidas. Primeiro é feito o cálculo de similaridade de usuários, após isso, são recomendados exercícios de acordo com a semelhante de assuntos entre as questões (similaridade de itens).

3.2.7 Síntese dos trabalhos relacionados

Com o objetivo de analisar os trabalhos apresentados neste capítulo, foi criada a Figura 3.9, que mostra os pontos em comum e diferenças em relação a:

AUTOR	TIPO DE CONTEÚDO RECOMENDADO	GRUPO DE USUÁRIO QUE RECEBE A RECOMENDAÇÃO	AMBIENTE	BASE UTILIZADA PARA CRIAÇÃO DO PERFIL DO USUÁRIO	ABORDAGEM DE RECOMENDAÇÃO	MÉTRICAS PARA VALIDAÇÃO DA RECOMENDAÇÃO
[Klasnja-Milicevic et al., 2017]	Material didático	Alunos	AVA	Questionário em Formulários	Filtragem Colaborativa	Recall / Precision
[Hos, 2017]	Códigos de linguagem de Programação	Alunos	AVA	Questionário em Formulários	Baseado em Conceito	Comparação com opnião de especialista humano
[Chau et al., 2017]	Material didático	Professores	AVA	Código de linguagem de programação / Questionários em Formulários	Baseado em Conceito	Recall / Precision
[Yan et al., 2017a]	Material didático	Alunos	AVA	Código de linguagem de programação	SKP-LS	Não utilizou
[Francisco e Ambrosio, 2015]	Exercícios	Alunos	Juiz online	Código de linguagem de programação / Questionários em Formulários	Filtragem Colaborativa	Não utilizou
[Vesin et al., 2013b]	Exercícios/Material Didático	Alunos	AVA	Código de linguagem de programação / Questionários em Formulários / Logs de navegação no AVA	Filtragem Colaborativa	Não utilizou
[De Oliveira et al., 2013]	Exercícios	Alunos	Juiz online	Código de linguagem de programação	Filtragem Colaborativa	Comparação com opnião de especialista humano / One-Error / Coverage / Ranking Loss / Avarage Precision
[Vesin et al., 2012]	Exercícios/Material Didático	Alunos	AVA	Código de linguagem de programação / Questionários em Formulários / Logs de navegação no AVA	Filtragem Colaborativa	Não utilizou
[Yera e Martínez, 2017]	Exercícios	Alunos	Juiz online	Códigos de linguagem de programação	Filtragem Colaborativa	Comparação com trabalhos correlatos
[Vesin et al., 2011]	Exercícios/Material Didático	Alunos	AVA	Código de linguagem de programação / Questionários em Formulários / Logs de navegação no AVA	Filtragem Colaborativa	Não utilizou
[Ansari et al.,2017]	Exercícios	Alunos	AVA	Código de linguagem de programação /Questionários em Formulários / Logs de Busca na Web	Abordagem Híbrida	Questionários destinados aos usuários
[Yu e Chen, 2016]	Exercícios	Alunos	Juiz online	Código de linguagem de programação	Filtragem Colaborativa	Não utilizou
[Fu et al.,2017]	Material didático	Alunos / Professores	AVA	Códigos de linguagem de programação	Filtragem Colaborativa / Ontologias	Recall / Precision
[Toledo, 2014]	Exercícios	Alunos	Juiz online	Códigos de linguagem de programação	Filtragem Colaborativa	Recall / Precision
[De Paula et al., 2015]	Exercícios	Alunos	Juiz online	Códigos de linguagem de programação	Filtragem Colaborativa	Recall / Precision / F1 Score
[Sánchez-Ruiz et al. 2017]	Exercícios	Alunos	Juiz online	Códigos de linguagem de programação	Itinerários de aprendizagem (Grafos diferionados)	Recall / Precision / F1 Score /Mean Reciprocal Rank / 1-Hit
[Sudha et al. 2017]	Exercícios	Alunos	Juiz online	Códigos de linguagem de programação	Redes Neurais Convulucionais	Matriz de Confusão / Recall / Precision / Acurácia
[Caro-Martinez e Jimenez-Diaz 2017]	Exercícios	Alunos	Juiz online	Códigos de linguagem de programação	Filtragem Colaborativa	Precision / Recall / F1 Score / Mean Reciprocal Rank / 1-hit
[Yu et al. 2015]	Exercícios	Alunos	Juiz online	Códigos de linguagem de programação	Filtragem Colaborativa	Recall
Proposta do Trabalho	**Exercícios**	**Alunos**	**Juiz online**	**Códigos de linguagem de programação**	**Filtragem Colaborativa**	**Simililaridade entre as recomendações / Médias das dificuldades**

Figura 3.9 – Comparação entre os trabalhos relacionados.
Fonte: Própria

- **Tipo de conteúdo recomendado:** qual conteúdo foi recomendado?
- **Grupo de usuário que recebe a recomendação:** qual o público alvo em que a recomendação de conteúdo está sendo direcionada?
- **Ambiente:** em que ambiente as recomendações são realizadas aos usuários?
- **Base utilizada para criação do perfil do usuário:** de onde são extraídos os dados para criar o perfil do usuário?

- **Abordagem de recomendação:** qual o método/abordagem de recomendação é utilizado no trabalho?

- **Métricas para validação da recomendação:** quais métricas são utilizadas para validar os resultados das recomendações feitas aos usuários pelo método utilizado?

3.2.8 Contribuição deste trabalho em relação aos trabalhos correlatos

Como é visível na Figura 3.9, a maioria dos trabalhos que fazem recomendação em juízes online utiliza a filtragem colaborativa como abordagem de recomendação, somente dois trabalhos não usam esse método.

Acredita-se que o trabalho proposto nesta pesquisa trará avanços significativos, pois comparado aos trabalhos relacionados, será o único que irá fazer recomendações em um juiz online utilizando a utilizando a filtragem colaborativa com o objetivo de mapear o índice de dificuldade dos exercícios de programação resolvidos pelos alunos.

Sendo assim, este trabalho inova no sentido de ser o único dentre os trabalhos relacionados na Tabela 3.9, que utilizará da técnica de *Scaffolding* citada por [Souza et al. 2016] como uma das possíveis soluções para minimizar o número de reprovações de alunos de programação.

Neste trabalho, o *Scaffolding* será aplicado no sentido de proporcionar ao aluno uma lista de exercícios adaptada ao seu nível de conhecimento, na medida do possível, sempre listando exercícios em ordem crescente de dificuldade. Assim, o aluno não se sentirá desestimulado ao resolver inicialmente exercícios considerados difíceis para ele.

Os assuntos relativos aos exercícios serão classificados pelos professores, de forma manual. O foco deste trabalho é, uma vez que os professores relacionam uma determinada lista de exercícios para os alunos resolverem, gerar a melhor ordem de resolução. Aqui, a melhor ordem de resolução é classificada pela dificuldade prevista para cada aluno de forma individual. O Capítulo 4 possui maiores informações sobre o método proposto nesta pesquisa.

3.3 Considerações finais

Foi apresentada uma RSL sobre os sistemas de recomendação de conteúdo para disciplinas de CS1 em plataformas de aprendizagem online. O processo de revisão foi constituído de três etapas: planejamento, execução e resultados.

A pesquisa retornou um total 1637 estudos, entre os quais foram selecionados 112 e 19 trabalhos após o 1º e 2º filtros, respectivamente. Os estudos excluídos durante essas fases ou eram artigos duplicados ou não atendiam aos critérios de inclusão.

A respeito dos 19 estudos que entraram na fase de extração, esses foram devidamente lidos, analisados e fichados, a fim de responder às questões de pesquisa dessa RSL. As informações extraídas de todas as fichas foram sintetizadas e apresentadas neste capítulo.

Foi possível verificar, com base nos estudos, que a maioria dos trabalhos utilizam *logs* de exercícios resolvidos em alguma linguagem de programação para criar o perfil de programação. Esses *logs* incluem dados tais como estruturas utilizadas para resolução do exercício (Estruturas de repetição, seleção, operadores, vetores e etc), quantidade de exercícios resolvidos e número de tentativas.

A abordagem de recomendação mais utilizada é a de filtragem colaborativa. Porém, durante esta pesquisa, percebeu-se que os trabalhos que utilizam tal abordagem sugerem como trabalhos futuros que seja feita o uso de abordagem híbrida para melhorar os resultados das recomendações feitas.

Com base nos resultados desta RSL, este trabalho propõe recomendação de exercícios em um juiz online. Para isso, é implementado um perfil de programação que contém atributos extraídos dos *logs* de exercícios resolvidos anteriormente. Para analisar a similaridade entre os alunos é utilizada a abordagem de filtragem colaborativa.

Capítulo 4

Método Proposto

A proposta desta pesquisa é utilizar dados de um juiz online para recomendar exercícios de programação ordenados pelo nível de dificuldade a alunos de disciplinas CS1. Para isso, deve-se criar um Perfil de Programação (PP). O PP contém atributos que quantificam a dificuldade que o aluno sente ao responder determinado exercício de programação.

Com os dados do PP utiliza-se a filtragem colaborativa para identificar alunos similares e fazer recomendações de exercícios com base na similaridade entre os alunos.

Em resumo, neste capítulo será explanado o PP e o método de recomendação baseado em filtragem colaborativa.

4.1 Arquitetura do método proposto

A Figura 4.1 mostra a arquitetura do método de recomendação de exercícios, proposto nesta pesquisa. Em resumo, o principal objetivo desse método é recomendar exercícios de programação para o aluno em uma ordem crescente de dificuldade.

As recomendações feitas ao aluno são sempre de exercícios de um tema previamente cadastrado por um professor da disciplina de CS1. Para recomendar exercícios ao aluno, é necessário que ele tenha resolvido outros exercícios sobre o mesmo tema, pois assim, é possível identificar as dificuldades do aluno.

Enquanto o aluno utiliza o ACAC para resolver exercícios, coletam-se dados com o objetivo de mapear a dificuldade sentida pelo aluno em cada exercício. Alguns dos dados coletados são: a quantidade de vezes que o aluno testa o código, a quantidade de erros

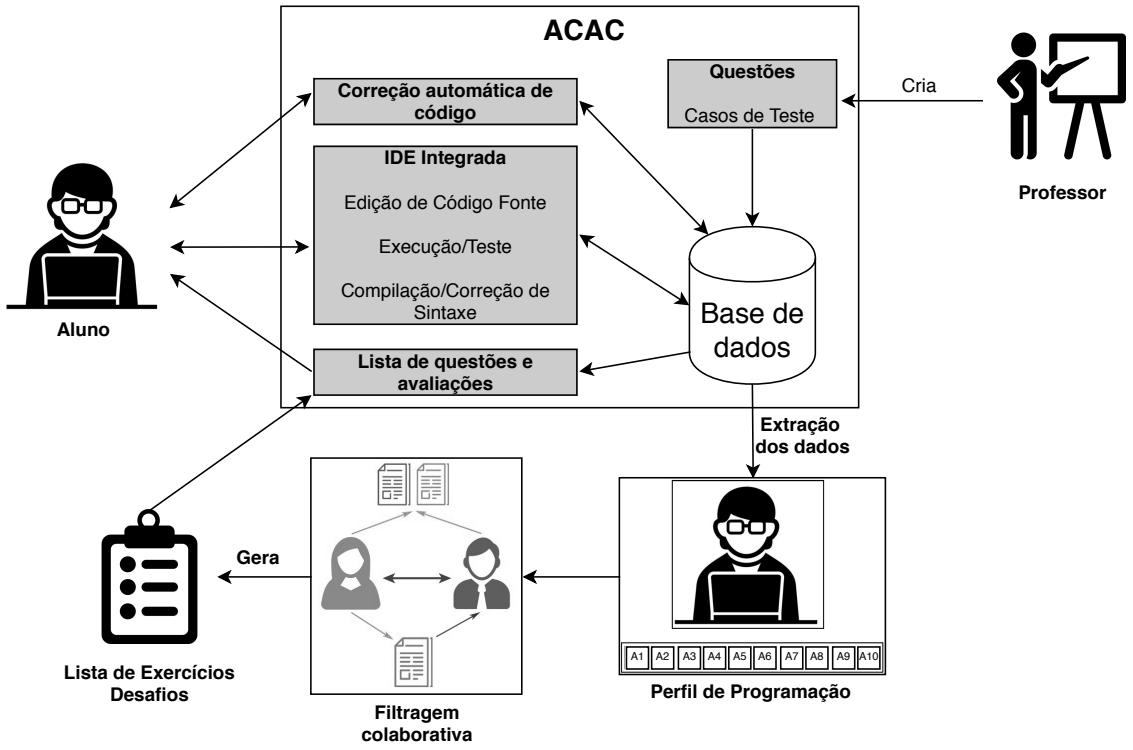


Figura 4.1 – Arquitetura proposta.

cometidos, erros de codificação cometidos, o tamanho do código do aluno, a quantidade de vezes que o aluno aperta a tecla *delete* ou *backspace*, dentre outros.

Com os dados devidamente coletados, cria-se o Perfil de Programação (PP)[1]. O PP é importante nesse método, pois é com ele que é identificado o índice de dificuldade do exercício resolvido pelo aluno. Executa-se esse procedimento para todos os alunos pertencentes à disciplina de CS1. Então, dividem-se os alunos da disciplina em dois grupos, sendo o grupo de controle e experimental.

Utilizando os dados do PP, aplica-se a filtragem colaborativa[2]. Aqui, utilizam-se dados de alunos do grupo de controle para identificar a ordem ideal de exercícios para os alunos do grupo experimental. Finalmente, para cada aluno do grupo experimental, gera-se uma lista de exercícios, ordenada pelo grau de dificuldade para um determinado tema.

[1]Na Seção 4.2, os dados do PP serão explicados com mais detalhes.
[2]Na Seção 4.3, a filtragem colaborativa utilizada neste método será explicada com mais detalhes.

4.2 Perfil de Programação

[Dwan et al. 2017] utiliza os dados de *logs* de juízes online para tentar prever se um aluno será aprovado ou não em disciplinas de CS1. Para isso, foi necessário criar um perfil de programação para cada aluno. Esse perfil foi elaborado a partir de evidências baseadas nos códigos fontes submetidos pelos alunos à medida que eles resolviam listas de exercícios de disciplinas básicas de programação.

Nesta pesquisa, adaptou-se o Perfil de Programação proposto por [Dwan et al. 2017]. Aqui, utilizam-se 10 atributos contendo os índices de dificuldade de cada exercício resolvido que são coletados no momento em que o aluno está resolvendo algum exercício direto no juiz online. Segue a descrição de todos os atributos coletados e utilizados.

Atributos do perfil de programação utilizados nesta pesquisa:

- Errors (A1): quantidade de submissões de códigos que não são validadas como corretas pelo juiz online para um determinado exercício;
- Submissions (A2): quantidade de vezes que o aluno submete o código do exercício até o juiz online validá-lo como certo;
- Tests (A3): quantidade de vezes que o aluno testa o código de um determinado exercício sem submetê-lo para validação do juiz online;
- Log size (A4): para cada exercício que o aluno resolve é gerado um *log* contendo os códigos que o aluno submeteu para avaliação e as saídas dos casos testes executados pelo juiz online. O tamanho do *log* é um número inteiro contendo a quantidade de linhas do arquivo de *log* do exercício submetido;
- Backspace (A5): quantidade vezes em que o aluno aperta a tecla "*backspace*" no código que foi validado como certo pelo juiz online em um determinado exercício;
- Type Error (A6): quantidade de vezes que uma operação ou função é aplicada a um objeto de um tipo inadequado em um determinado exercício. Exemplo: na operação "2" + 2 é gerado um Type Error, pois no Python 3 não é possível somar objetos do tipo *string* ("2") com objetos do tipo inteiro (2);
- Syntax Error (A7): quantidade de vezes que o analisador do *Python 3* encontra um erro de sintaxe em um determinado exercício. Isso pode ocorrer em uma importação

de instrução, em uma chamada para as funções internas, ou ao ler uma entrada padrão.

- Value Error (A8): quantidade de vezes que é o argumento de uma função é passado com um tipo inadequado em um determinado exercício. Exemplo: a instrução int("2") lança um Value Error, pois a função int() deve receber atributos do tipo int e não do tipo *string*;

- Name Error (A9): quantidade de vezes que um objeto não pode ser encontrado por não ter sido declarado no código de um determinado exercício.

- Indentation Error (A10): quantidade de vezes que é há um recuo incorreto no código de um determinado exercício.

O índice de dificuldade do exercício não pode ser representado nos casos em que o aluno copia a solução do exercício de outra pessoa ou outro local. Para que a dificuldade possa ser mapeada o aluno necessariamente deve utilizar e responder o exercício no IDE do juiz online.

Os atributos A1, A2 e A3 têm o objetivo de indicar a dificuldade do aluno em encontrar a solução para o exercício em questão. Quanto menor for a quantidade de erros, submissões e testes, menor é a dificuldade que o aluno sente ao responder um exercício.

O atributo A4 tem o objetivo de indicar a dificuldade do aluno em propor uma boa solução para o problema, pois quanto menor for o tamanho do *log* gerado pelo IDE do juiz online enquanto o aluno responde o exercício, menor é o tamanho do código, menos mensagens de erros são mostradas, ou seja, menor foi a dificuldade do aluno ao responder o exercício.

O atributo A5 tem o objetivo de indicar a dificuldade do aluno em compreender e utilizar corretamente os recursos da linguagem de programação, utilizada para responder o exercício, pois ao utilizar um comando de forma errada o IDE notifica o aluno que o comando não é válido. Nesses casos, o aluno pode apertar a tecla *backspace* para corrigir seu erro.

Os atributos A6, A7, A8, A9 e A10 têm o objetivo de indicar a dificuldade do aluno em relação a conceitos da linguagem de programação, utilizada para resolver o exercício. Quanto menor for a quantidade desses tipos de erros cometidos, menos dificuldade o aluno tem ao resolver o exercício.

A maioria dos dados presentes no PP pode ser extraída de qualquer juiz online que possua um IDE integrado, com poucas adaptações. Isso mostra que o método proposto pode ser adaptado em outros ambientes. Além disso, os dados do PP podem ser coletados com o objetivo de mapear características que não sejam o índice de dificuldade.

Após a coleta dos dados do PP, realiza-se a Normalização Min-Max, onde o menor valor (Min) de cada atributo é mapeado como 0 e o maior valor (Max) de cada atributo é mapeado como 1.

Com os dados do PP já normalizados, é possível usar o cálculo de distância euclidiana para verificar quão parecido ou diferente um aluno é do outro. Na seção seguinte será apresentado como é feita a recomendação de exercícios com o uso da técnica de recomendação baseada em Filtragem Colaborativa.

4.3 Filtragem Colaborativa

Nesta pesquisa, cada aluno é representado pelo PP. Sendo assim, é possível verificar quão similar um aluno é do outro utilizando a técnica de filtragem colaborativa, através da distância euclidiana multidimensional [Venturi 2015].

A Tabela 4.1 ilustra os dados presentes no PP de 3 alunos (A, B, C), cada exercício é representado por um número inteiro e os atributos do PP são identificados por A1, A2, ..., A10 e contêm valores que variam de 0 a 1. Note que o aluno A só respondeu o exercício 1800, enquanto os alunos B e C resolveram os exercícios 1800, 1801 e 1802. Supondo que o exercício 1800 é um exercício avaliativo feito em sala de aula, ou seja, não é um exercício opcional ou um item de uma prova, e o exercício 1801 e 1802 são exercícios opcionais, podemos fazer uma simulação de como funciona o método de recomendação proposto nesta pesquisa.

Para fim de simulação, iremos assumir que os exercícios 1801 e 1802 pertencem ao mesmo tema. Pretendemos encontrar qual a ordem ideal para o aluno resolver, que nesse caso pode ser 1801, 1802 ou 1802, 1801.

No método de recomendação proposto nesta pesquisa, o primeiro passo é identificar a similaridade entre os alunos. Para isso, aplica-se a distância euclidiana multidimensional

Tabela 4.1 – Dados dos PP dos alunos fictícios A, B e C

Aluno	Exercício	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A	1800	0.00	0.06	0.00	0.04	0.01	0.00	0.00	0.00	0.00	0.00
B	1800	0.08	0.41	0.10	0.00	0.03	0.14	0.00	0.00	0.00	0.00
B	1801	0.02	0.00	0.12	0.06	0.08	0.00	0.23	0.00	0.50	0.00
B	1802	0.00	0.04	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00
C	1800	0.00	0.05	0.00	0.04	0.01	0.00	0.00	0.00	0.00	0.00
C	1801	0.01	0.30	0.08	0.12	0.07	0.00	0.22	0.48	0.00	0.00
C	1802	0.02	0.00	0.12	0.06	0.08	0.00	0.23	0.00	0.50	0.00

adaptada para o modelo de recomendação de exercícios desta pesquisa, dada pela Fórmula 4.1.

$$\delta(Aa,Ao) = \sqrt{\sum_{i=1}^{10}(Ma_i - Mo_i)^2} \tag{4.1}$$

Onde temos:

- Aa - id de um aluno;
- Ao - id de outro aluno;
- Ma - atributo do PP do Aluno Aa;
- Mo - atributo do PP do Aluno Ao;
- i - índice do atributo do PP;

Utilizando os dados dos Alunos A, B e C, podemos fazer os seguintes testes:

$$\delta(A,B) = 0,4 \tag{4.2}$$

$$\delta(A,C) = 0,1 \tag{4.3}$$

Para facilitar todos os experimentos, a distância euclidiana entre dois alunos será mostrada em porcentagem. Na similaridade de 100% temos alunos com o perfil de programação iguais. Nos casos em que os alunos não possuem nenhum exercício em comum, não é possível calcular a similaridade, a similaridade atribuída será 0%.

A conversão da distância euclidiana para similaridade é dada pela Fórmula 4.4.

$$S(Aa,Ao) = 1/(1+\sqrt{\delta(Aa,Ao)}) * 100 \tag{4.4}$$

Aplicando a Fórmula 4.4 nos exemplos anteriores, teremos:

$$S(A,B) = 61\% \tag{4.5}$$

$$S(A,C) = 75\% \tag{4.6}$$

Nesse caso, as recomendações do Aluno C para o Aluno A terão mais influência do que as recomendações do Aluno B para o Aluno A. Isso irá ocorrer porque $S(A,C) > S(A,B)$. Na Seção 4.4 será explicado com mais detalhes como o método desta pesquisa utiliza a Fórmula 4.4 no processo de recomendação de exercícios.

4.4 Recomendações de exercícios para o aluno

Para fazer as recomendações, utiliza-se a base de dados de um juiz online. Dividem-se os alunos em dois grupos, sendo o grupo de controle, a fim de aplicar a filtragem colaborativa, e alunos do grupo experimental, que receberão as recomendações. Chamaremos os alunos do grupo de controle de GC e os alunos do grupo experimental de GE.

Para que um aluno pertencente ao GE receba a recomendação de exercícios, cria-se uma função que verifica a similaridade que o aluno possui com todos os outros alunos pertencentes ao GC. É importante frisar que, para evitar o problema de *Cold-Start*[3], o cálculo de similaridade entre os alunos não utiliza dados de exercícios de opcionais ou provas, mas somente exercícios avaliativos.

Após isso, verifica-se qual o índice de dificuldade de cada exercício opcional respondido por cada aluno pertencente ao GC. O índice de dificuldade é dado pelo somatório dos

[3]Problema causado por novos usuários ou itens que não tenham apresentado quaisquer avaliações, e, portanto não podem ser comparados a outros.

10 atributos do PP, a Fórmula 4.7 representa o cálculo feito para determinar a dificuldade que um aluno sente em cada exercício.

$$D(Exercicio) = \sum_{i=1}^{10} Ma_i \tag{4.7}$$

- Ma - atributo do PP;
- i - índice do atributo do Perfil de Programação;

A Tabela 4.2 ilustra essa etapa do processo com os alunos B e C, com base nos mesmos dados utilizados para exemplificar as etapas anteriores.

Tabela 4.2 – Tabela ilustrativa do índice de dificuldade dos exercício e similaridades com aluno alvo da recomendação

Aluno GC	**Exercício**	**Índice de dificuldade**	**Similaridade aluno com o Aluno A**
B	1801	1.01	61%
B	1802	0.07	61%
C	1801	1.27	75%
C	1802	1.01	75%

Uma vez que o cálculo da similaridade e dificuldade são feitos, calcula-se a estimativa da dificuldade que o aluno do GE sentiria nos exercícios mais complexos ou exercícios de provas. Para isso, aplica-se a Fórmula 4.8.

$$\omega(Exercicio) = \frac{D_1.S_1 + D_2.S_2 + ... + D_n.S_n}{S_1 + S_2 + ... + S_n} \tag{4.8}$$

Onde temos:

- Dn - Índice de dificuldade do exercício em questão resolvido pelo aluno “n”;
- Sn - Similaridade entre o aluno “n” e o aluno alvo da recomendação.

Utilizando os dados da Tabela 4.2 estima-se a dificuldade que o Aluno A sentiria ao resolver o exercício 1801. A Equação 4.9 mostra os cálculos utilizados neste método de recomendação para estimar a dificuldade do aluno A ao resolver o exercício 1801:

$$\omega(Exercicio1801) = \frac{1,01.0,61 + 1,27.0,75}{0,61 + 0,75} \tag{4.9}$$

A Equação 4.10 representa a dificuldade estimada obtida para o exercício 1802:

$$\omega(Exercicio1802) = \frac{0,07.0,61 + 1,01.0,75}{0,61 + 0,75} \tag{4.10}$$

Após isso ser feito em todos os exercícios opcionais, registram-se os resultados em uma estrutura de dados representada pela Tabela 4.3. Note que, nessa estrutura, os exercícios são ordenados pela dificuldade estimada, sempre da menor dificuldade para maior.

Tabela 4.3 – Lista de dificuldades estimadas para os exercícios opcionais recomendados para o aluno A

Aluno alvo	**Exercício**	**Dificuldade estimada**
A	1802	0.59
	1801	1.15

Nesse caso hipotético, o primeiro exercício opcional recomendado para o Aluno A resolver seria o exercício 1802, logo após, seria recomendado o exercício 1801. Executa-se esse procedimento para cada aluno que se deseja obter recomendação da ordem de exercício. Assim, os exercícios são sempre recomendados pela estimativa de dificuldade que o aluno em questão sentiria.

4.5 Considerações finais

Nesse capítulo foi apresentado o modelo de recomendação de exercícios. Para fazer as recomendações, coleta-se dados através do IDE integrado de um juiz online. Após a coleta, para cada aluno, cria-se um Perfil de Programação, e dividem-se os alunos em dois grupos, sendo o grupo de controle (GC) e o grupo experimental (GE).

Para cada aluno do GE aplica-se a filtragem colaborativa entre todos alunos do GC. Faz-se isso para medir a similaridade entre o aluno que irá receber a recomendação e os alunos do GC.

Uma vez feito o cálculo de similaridade, faz-se a estimativa da dificuldade que o aluno que receberá a recomendação sentiria para cada um dos exercícios. Finalmente, recomenda-se exercícios ao aluno, sempre do exercício com a menor dificuldade estimada para a maior dificuldade estimada.

Capítulo 5

Experimentos e resultados

Neste capítulo serão apresentados os resultados dos experimentos realizados nesta pesquisa. Os experimentos serão divididos em duas partes. A primeira está relacionada com as etapas do método proposto, como a extração dos dados dos alunos, criação do Perfil de Programação, o cálculo de similaridade entre os alunos e a recomendação de exercícios. A segunda parte tem a ver com a validação do método, na qual será feita a análise dos resultados obtidos sobre as recomendações feitas aos alunos.

5.1 Contexto Educacional

Nesta pesquisa foi utilizado o conjunto de dados dos alunos de turmas de Introdução à Programação (CS1), matriculados no ano de 2018, da Universidade Federal do Amazonas (UFAM). Os alunos do semestre 2018-1 foram classificados como Grupo de Controle (GC) e os alunos do semestre 2018-2 foram classificados como Grupo Experimental (GE). Nesse caso, os alunos do GE receberam recomendações de exercícios.

A disciplina de CS1 é dividida em 7 módulos. Seguem os tópicos estudados em cada módulo.

1. Variáveis e estrutura de programação sequencial;
2. Estrutura condicional simples e composta;
3. Estrutura condicional encadeada;
4. Estrutura de repetição por condição;

5. Vetores e Strings;

6. Estrutura de repetição por contagem;

7. Matrizes;

A Figura 5.1 mostra a estrutra de atividades desenvolvidas pelos alunos das disciplinas de CS1 em cada módulo. Os alunos resolvem uma lista de exercícios avaliativos, uma lista de exercícios opcionais (exercícios com maior grau de complexidade) e uma prova.

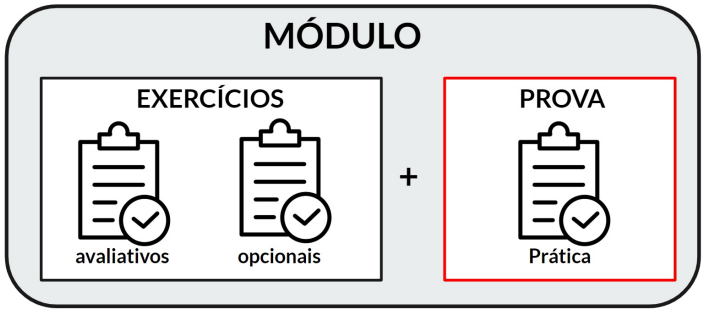


Figura 5.1 – Estrutura de atividades praticadas pelos alunos em cada módulo nas disciplinas de CS1 na UFAM

A lista de exercícios avaliativos possui o objetivo de auxiliar o processo de ensino e aprendizagem dos assuntos estudados em cada módulo. A lista de exercícios opcionais, por possuir maior grau de dificuldade, possui o objetivo de fixar o assunto estudado, elevando assim o grau de conhecimento do aluno. A prova possui o caráter avaliativo. Na média cada lista de exercícios avaliativos e exercícios opcionais possuem 10 questões práticas. A prova contém 2 questões práticas.

Atualmente, a ordem de resolução dos exercícios opcionais é proposta pelo professor da disciplina. Essa ordem é a mesma para todos os alunos. Por isso, pode acontecer que os alunos sintam diferentes níveis de dificuldade ao resolver os exercícios. Sendo assim, o aluno resolvendo primeiro os exercícios mais difíceis, pode ser desestimulado.

Nesta pesquisa os exercícios feitos em sala de aula foram utilizados como base para identificar a similaridade entre os alunos. Após isso, foi recomendado uma lista de exercícios opcionais personalizada aos alunos, onde foi sugerido a resolução de exercícios ordenados por dificuldade crescente.

5.2 Coleta de dados e Perfis de Programação

Os dados sobre os alunos e exercícios foram coletados no sistema CodeBench[1]. O CodeBench é um Juiz Online desenvolvido pelo Instituto de Computação (IComp) da Universidade Federal do Amazonas, Brasil. Através do CodeBench, os professores podem fornecer listas de exercícios de programação para seus alunos, que, por sua vez, devem desenvolver soluções para cada exercício através de um IDE integrado. Quando um aluno envia um código-fonte para um determinado exercício, o sistema notifica imediatamente o aluno se a solução está correta ou não.

O CodeBench registra automaticamente todas as ações executadas pelos alunos no IDE incorporado durante suas tentativas de resolver os exercícios propostos.

No total, 645 alunos resolveram sete listas de exercícios avaliativos e sete listas de exercícios opcionais utilizando a linguagem de programação Python 3.

A Tabela 5.1 mostra os dados coletados e utilizados nesta pesquisa.

Tabela 5.1 – Informações coletadas do juiz online CobeBench durante o ano letivo de 2018.

Informações Registradas	**2018-1 (GC)**	**2018-2 (GE)**	**Total**
Classes CS1	9	5	14
Número de alunos	465	180	645
Exercícios práticos	1550	893	2.443
Códigos (Desenvolvidos por alunos)	47.969	13.458	61.427
Testes e submissões de códigos	49.993	14.536	64.529

Para cada aluno é gerada uma série de informações que foram extraídas e utilizadas nesta pesquisa para propor as recomendações de forma individualizada.

No momento em que o aluno codifica a solução do exercício utilizando o CodeBench, são coletados dados sobre as submissões de código, tais como erros de sintaxe. Além disso, dados de funções especiais do teclado e mouse são coletados. A Figura 5.2 mostra os dados coletados do exercício 820 codificado pelo aluno 1085.

Como já foi apresentado na Seção 4.2, o primeiro passo para gerar a recomendação de exercícios opcionais ordenados por nível de dificuldade para um determinado aluno é coletar os dados sobre os exercícios avaliativos revolvidos e gerar uma estrutura de dados, aqui denominada de Perfil de Programação (PP).

[1] http://codebench.icomp.ufam.edu.br

```
2018-6-30 17:10:09.669#viewportChange#0
2018-6-30 17:10:20.134#viewportChange#0
2018-6-30 17:11:19.678#mousedown#{"isTrusted":true}
2018-6-30 17:11:19.698#focus#
2018-6-30 17:11:25.926#mousedown#{"isTrusted":true}
2018-6-30 17:11:26.742#mousedown#{"isTrusted":true}
2018-6-30 17:11:28.026#copy#{"isTrusted":true}
2018-6-30 17:11:28.228#copy#{"isTrusted":true}
2018-6-30 17:11:28.501#keyHandled#"Backspace"
2018-6-30 17:11:28.707#keyHandled#"Backspace"
2018-6-30 17:11:28.843#keyHandled#"Backspace"
2018-6-30 17:11:29.009#keyHandled#"Backspace"
2018-6-30 17:11:29.176#keyHandled#"Backspace"
2018-6-30 17:11:29.334#keyHandled#"Backspace"
2018-6-30 17:11:29.503#keyHandled#"Backspace"
2018-6-30 17:11:29.671#keyHandled#"Backspace"
2018-6-30 17:11:29.837#keyHandled#"Backspace"
2018-6-30 17:11:29.998#keyHandled#"Backspace"
```

Figura 5.2 – Dados adicionais de teclado e mouse coletados no momento da codificação do exercício 820 do aluno 1085

Os dados do PP de um aluno para uma questão são calculados a partir dos dados coletados pelo CodeBench. A Tabela 5.2 apresenta uma amostra do total de exercícios desenvolvidos pelo Aluno 3702, que foi escolhido aleatoriamente para exemplificar os experimentos feitos nesta pesquisa. Cada aluno é representado por um número inteiro, rotulado como “IdAluno”.

Tabela 5.2 – Amostra de 5 exercícios presentes no PP do aluno 3702

idExercício	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**
1326	0,00	0,00	0,00	0,05	0,02	0,00	0,00	0,00	0,00	0,00
1327	0,00	0,01	0,14	0,30	0,16	0,00	0,00	0,00	0,00	0,00
1328	0,00	0,00	0,00	0,08	0,05	0,00	0,00	0,00	0,00	0,00
1329	0,00	0,06	0,00	0,05	0,03	0,00	0,00	0,00	0,00	0,00
1330	0,06	0,04	0,01	0,10	0,04	0,00	0,09	0,00	0,00	0,00

É importante frisar que essa tabela não contém nenhum dado sobre exercício opcional. Para fins de cálculo de similaridade, nenhum exercício opcional ou questão de prova entra no PP. Já nos alunos do GC são coletados os dados de todos os exercícios, pois eles foram utilizados no momento em que foi feita a filtragem colaborativa. A Seção 5.2.1 mostra os dados de similaridade entre o Aluno 3702 pertencente ao GE e todos os alunos do GC.

5.2.1 Cálculo de similaridade entre os alunos do GC e o aluno alvo da recomendação

Como já foi descrito na Seção 4.3, no método proposto nesta pesquisa, as recomendações para um dado aluno são feitas com base na similaridade desse aluno com outros alunos do grupo controle. Quanto maior for a similaridade entre os alunos, mais "peso" um aluno terá sobre o outro no momento da recomendação. Sendo assim, serão apresentados resultados obtidos no método proposto usando como exemplo um aluno pertencente ao GE. Nesse caso em específico, os resultados para o aluno 3702.

A Figura 5.3 mostra a similaridade em porcentagem entre aluno 3702 e todos os alunos do GC. É possível notar que a grande maioria dos alunos do GC possue similaridade inferior a 50% com o aluno alvo da recomendação. Nesse caso, esses dados não influenciam diretamente na qualidade da recomendação. Isso ocorre pois, para fazer a estimativa do índice de dificuldade de cada exercício opcional recomendado para o aluno alvo da recomendação, a similaridade é utilizada como peso ponderado. Nesse caso, os alunos com maiores similaridades (90% a 100%) exercem maior influência.

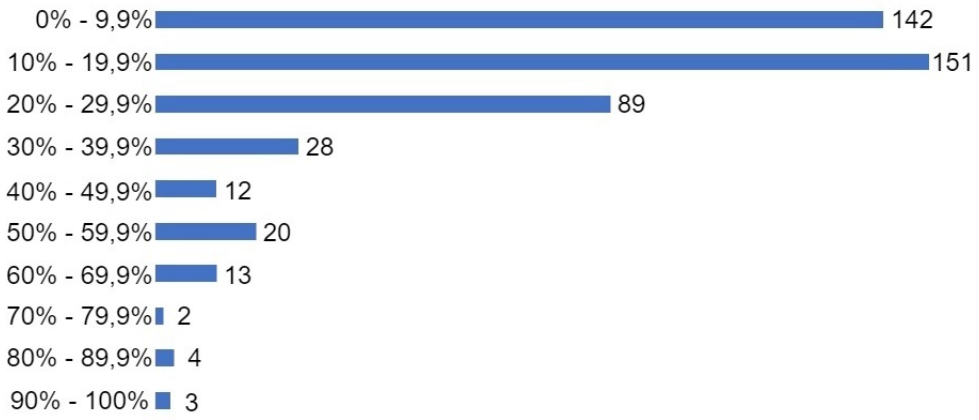


Figura 5.3 – Similaridade entre os alunos do GC o aluno alvo da recomendação (Aluno 3702).

Vale a pena frisar que esses dados são apenas do aluno 3702 em específico. Podem ocorrer casos que exista uma quantidade maior de outros alunos similares. Também podem ocorrer casos que o aluno por não possuir uma quantidade relevante de exercícios resolvidos, terá uma quantidade baixa de outros alunos semelhantes a si. Para esses casos, acredita-se que trabalhos futuros que não utilizam a filtragem colaborativa podem melhorar a qualidade da recomendação, pois esse atualmente é um problema em aberto na área de sistemas de recomendações que utilizam somente filtragem colaborativa.

5.3 Recomendações de Exercícios opcionais

Para exemplificar o processo de recomendação de exercícios opcionais, foram feitas recomendações para o Aluno 3702, que é um aluno pertencente ao GE. Este processo pode ser aplicado aos demais alunos da disciplina de CS1.

Atualmente, cada um dos 7 módulos da disciplina de CS1 da UFAM contém uma ordem fixa de exercícios opcionais. No Apêndice A, encontram-se os exercícios referentes ao primeiro módulo da disciplina (Variáveis e estrutura de programação sequencial), dispostos na ordem de resolução proposta pelos professores.

Utilizando o método de recomendação aplicado ao aluno 3702, foi gerada uma ordem diferente da proposta pelos professores. É importante frisar que o método de recomendação não recomenda novos exercícios, mas os mesmos exercícios propostos pelos professores, porém em uma ordem adaptada para cada aluno, de acordo com a dificuldade estimada nos exercícios opcionais.

A Tabela 5.3 apresenta a comparação entre a ordem dos exercícios propostos pelos professores e a ordem proposta pelo método de recomendação.

Tabela 5.3 – Tabela comparativa da ordem original dos exercícios opcionais e a ordem proposta pelo método de recomendação para o aluno 3702.

Primeira lista de exercícios opcionais		
-	Ordem original	Ordem proposta
1º	1800	2426
2º	2049	1446
3º	1446	1805
4º	2043	1803
5º	2425	2049
6º	2426	1800
7º	1801	1804
8º	1802	1806
9º	1803	1802
10º	1805	2425
11º	1804	1801
12º	1806	2043

Nesse caso, percebe-se que houve mudanças significativas na ordem das questões. Para exemplificar, veja que o primeiro exercício proposto pelos professores (Exercício

1800) é o sexto exercício proposto pelo método de recomendação. O último exercício proposto pelos professores (Exercício 1806) é o oitavo exercício proposto pelo método de recomendação. Isso ocorreu devido o fato de os exercícios propostos pelo método serem recomendados de acordo com a dificuldade estimada para cada exercício. Nesse caso, sempre da menor dificuldade estimada, para maior dificuldade estimada.

A Seção 5.4 mostra a médias das dificuldades dos exercícios opcionais na ordem proposta pelos professores e na ordem propostas pelo método de recomendação. Será possível perceber que, na maioria dos casos, a ordem proposta pelo método de recomendação sugere aos alunos exercícios com dificuldade menor, seguidos de exercícios com maior dificuldade.

5.3.1 Similaridade entre a ordem original dos exercícios opcionais e a ordem de exercícios opcionais recomendados aos alunos

Nesta seção será mostrado que a ordem de exercícios recomendada pelo método proposto nesta pesquisa tende a ser diferente da ordem proposta pelo professor da disciplina.

As listas de exercícios opcionais propostas pelo professor da disciplina é única, ou seja, possui a mesma ordem de resolução para todos os alunos. Sendo assim, na maioria dos casos, não atende as necessidades individuais do aluno em relação ao nível de dificuldade.

Então, a fim de mostrar a lista de exercícios recomendadas para os alunos tendem a ser diferentes da lista original proposta, foram escolhidos aleatoriamente 100 alunos do GE. Para cada aluno escolhido foi gerado uma lista de exercícios opcionais para cada um dos sete módulos da disciplina de CS1 da UFAM. Utilizando o coeficiente de Kendall, verificou-se a similaridade das listas de exercícios opcionais recomendados com a lista de exercícios na ordem original.

A correlação de Kendall ou coeficiente de Kendall[2] é uma medida da correspondência entre dois *rankings*. Valores do coeficiente de Kendall próximos a 1 indicam forte concordância, valores próximos a -1 indicam forte discordância [Kendall 1970]. Para viabilizar esse experimento, foi utilizado a biblioteca Python *scipy.stats.kendalltau*[3] para calcular os coeficientes de Kendall.

[2] Apresentado na Seção 2.5

[3] Documentação disponível em:
https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.kendalltau.html

Serão apresentados os valores dos Coeficientes de Kendall para as recomendações geradas para o aluno 3702. Para calcular o coeficiente de Kendall, foram utilizados dois rankings: sendo o primeiro a ordem da lista de exercícios opcionais proposta pelos professores, aqui chamado de Ordem Original (OO); e o segundo ranking é a ordem da lista de exercícios opcionais proposta pelo método de recomendação, aqui chamado de Ordem Recomendada (OR). No caso do aluno 3702 foi obtido:

- OO = [1800, 2049, 1446, 2043, 2425, 2426, 1801, 1802, 1803, 1805, 1804, 1806]
- OR = [2426, 1446, 1805, 1803, 2049, 1800, 1804, 1806, 1802, 2425, 1801, 2043]

Aplicando o cálculo do coeficiente de Kendall chegou-se ao seguinte resultado:

- τ(OO, OR) = -0.27

Esse procedimento foi aplicado com 100 alunos pertencentes ao GE escolhidos aleatoriamente. A Figura 5.4 mostra os resultados obtidos nesse experimento.

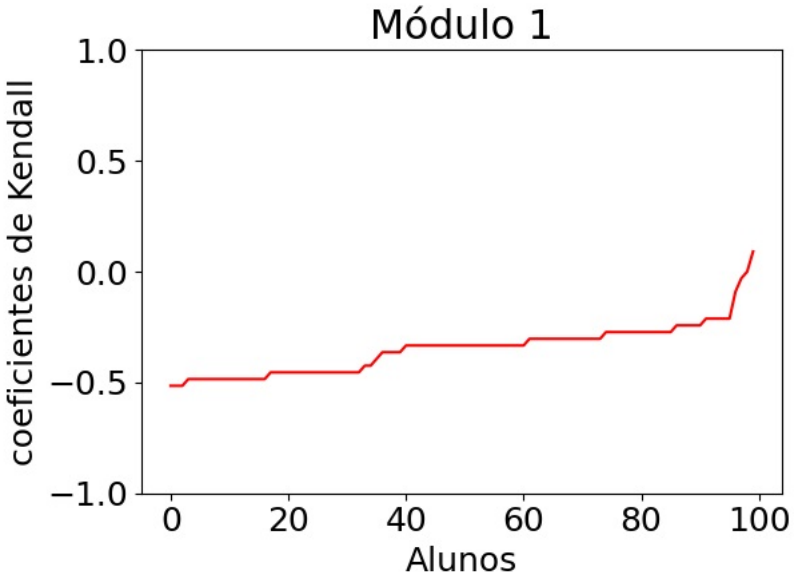


Figura 5.4 – Similaridade entre a ordem de exercício normal e exercícios recomendados.
Fonte: Própria

É possível notar que as similaridades variam de valores negativos (que por definição, mostra uma correlação com forte discordância) para valores positivos (que por definição, mostra uma correlação com forte concordância).

Isso mostra que a ordem dos exercícios recomendados pelo método de recomendação proposto nesta pesquisa realmente sugere uma lista de exercícios personalizada para cada aluno. Vale a pena lembrar que, no formato original, todos os alunos respondiam a lista de exercícios na mesma ordem.

Usando o método de recomendação aqui proposto, pode ocorrer casos em que a ordem de resolução exercícios opcionais sejam iguais, como também é possível notar no gráfico. Porém, isso não acontece na maioria das vezes. Caso a lista de exercícios opcionais possua um número maior de exercícios, a probabilidade da ordem ser mesma para os alunos diminui.

5.4 Comparativo do índice de dificuldade dos exercícios opcionais na ordem original e na ordem recomendada

Um dos objetivos deste trabalho é proporcionar ao aluno de programação um ambiente na qual a resolução dos exercícios possa ser de forma gradual de dificuldade. Em um cenário ideal, deseja-se que o aluno resolva sempre exercícios mais fáceis para os mais difíceis. Então, foi feito um experimento para tentar medir o nível de dificuldade que o aluno sentiria ao resolver a lista de exercícios proposta pelos professores e a lista de exercícios proposta pelo método de recomendação desta pesquisa.

A amostra desta pesquisa é composta por alunos dos semestres 2018.1 (GC) e 2018.2 (GE) da disciplina de introdução à programação da UFAM. Todos os alunos já passaram pela avaliação, que é composta pela resolução de exercícios avaliativos, exercícios opcionais e exercícios exames. Para fins de simulação, nos experimentos anteriores, assumia-se que os alunos do GE não tinham resolvido nenhum exercício opcional (mas na verdade, por ser tratar de alunos que já cursaram a disciplina em questão, os exercícios opcionais já tinham sido resolvidos).

Então, como os alunos resolveram os exercícios opcionais, utilizando os métodos de coleta de dados já apresentados nas Seções 4.2 e 5.1, foi possível mensurar a dificuldade que cada aluno sentiu ao resolver os exercícios opcionais.

Um experimento foi conduzido com os seguintes passos:

- Para manter a uniformidade dos dados, somente alunos do GE que concluíram a lista de exercícios opcionais foram selecionados (total de 66 alunos). Esse critério de escolha ocorreu devido ao fato de questões de contexto educacional que não fazem parte desta pesquisa, tais como: evasão, desistência, dentre outros; poderem influenciar os resultados desse experimento;

- Para os alunos selecionados foram coletados o índice de dificuldade dos exercícios opcionais. O índice de dificuldade é calculado aplicando a Fórmula 4.7 para cada exercício resolvido pelo aluno.

- Após a coleta dos dados, foi feita a comparação do índice de dificuldade dos exercícios na ordem original (ordem proposta pelos professores) e o índice de dificuldade na ordem proposta pelo método desta pesquisa. A Tabela 5.4 mostra a coleta dos dados feita para primeira lista de exercícios opcionais (Módulo 1 da disciplina de CS1) para o aluno 3702.

A Figura 5.5 mostra os mesmos dados da Tabela 5.4. Porém somente com o índice de dificuldade dos exercícios na ordem proposta pelos professores (linha vermelha) e o índice de dificuldade dos exercícios opcionais na ordem proposta pelo método de recomendação (linha azul).

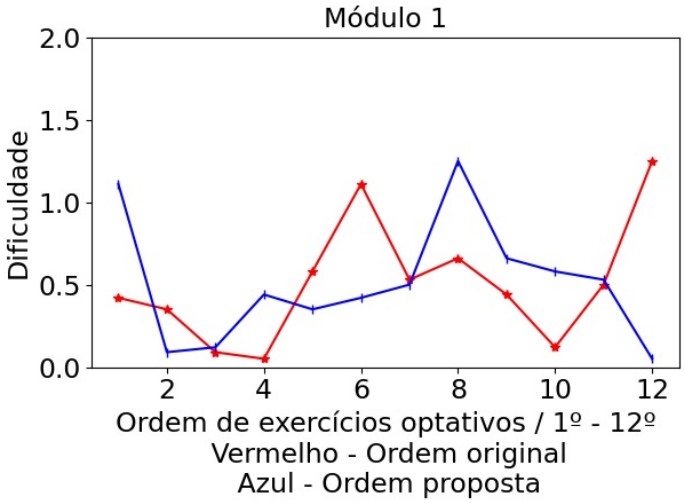


Figura 5.5 – Comparativo dos índices de dificuldade dos exercícios do aluno 3702.
Fonte: Própria

É possível notar nesse caso em específico, que essa situação ideal não é a ideal. Para isso, a linha azul do gráfico deveria ser estritamente crescente. Isso ocorre devido o fato

Tabela 5.4 – Comparativo dos índices de dificuldade dos exercícios resolvidos pelo aluno 3702

Primeira lista de exercícios opcionais				
-	Exercícios Ordem original	Dificuldade Ordem original	Exercícios Ordem proposta	Dificuldade Ordem proposta
1º	1800	0.42	2426	1.11
2º	2049	0.35	1446	0.09
3º	1446	0.09	1805	0.12
4º	2043	0.05	1803	0.44
5º	2425	0.58	2049	0.35
6º	2426	1.11	1800	0.42
7º	1801	0.53	1804	0.50
8º	1802	0.66	1806	1.25
9º	1803	0.44	1802	0.66
10º	1805	0.12	2425	0.58
11º	1804	0.50	1801	0.53
12º	1806	1.25	2043	0.05

das recomendações serem feitas com base na estimativa do índice de dificuldade que o aluno sentiria em cada exercício opcional, e para o caso do aluno 3702, a estimativa não foi satisfatória.

No entanto, quando se analisa todos os alunos que fazem parte desse experimento (aqueles que conseguiram resolver todos os exercícios opcionais do módulo 1) os resultado são satisfatórios.

A Figura 5.6 mostra a média dos índices de dificuldade dos exercícios opcionais na ordem proposta pelos professores e na ordem proposta pelo método de recomendação.

Os dados utilizados são os seguintes: verificou-se o índice de dificuldade do primeiro ao último exercício proposto pelos professores e, após isso, foi tirado a média dos índices de dificuldade para cada exercício isoladamente e plotado na linha vermelha. O mesmo procedimento foi feito com os exercícios propostos pelo método de recomendação, esses dados foram plotados na linha azul.

Percebe-se que, apesar da linha azul não ser estritamente crescente, na grande maioria dos casos os exercícios são recomendados para os alunos da menor dificuldade para maior dificuldade. Os casos específicos em que isso não ocorre podem ser estudados em

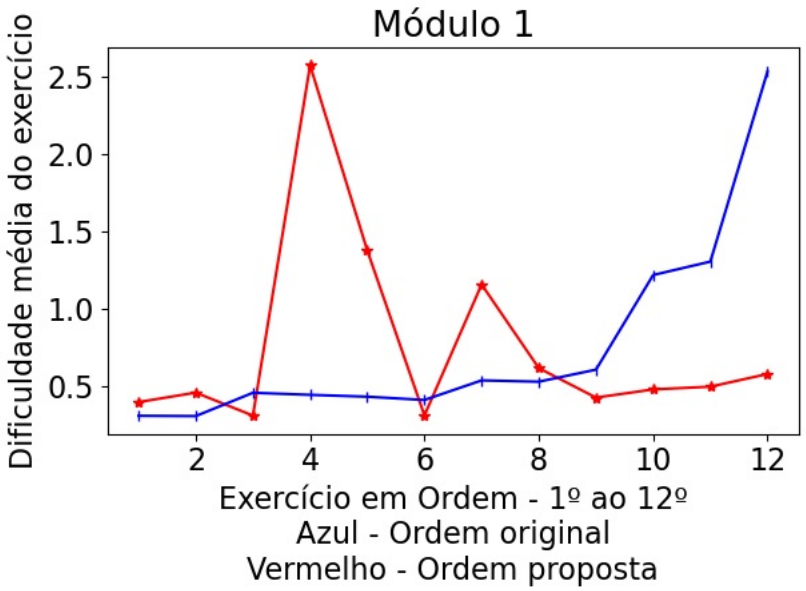


Figura 5.6 – Comparação da média dos índices de dificuldade dos exercícios resolvidos pelos alunos.

Fonte: Própria

pesquisas futuras, pois existem diversas soluções possíveis para esses casos, que podem variar desde a melhoria da implementação do método utilizado nesta pesquisa à questões de metodologias pedagógicas.

5.4.1 Comparação entre as dificuldades de exercícios na Ordem Original, Ordem Proposta e Ordem Ideal

Um último experimento foi conduzido para verificar a similaridade dos rankings das dificuldades. Aqui, as dificuldades dos exercícios são classificadas em 3 classes: dificuldades na Ordem Original (OO), Ordem Proposta (OP) e Ordem Ideal (OI).

Para calcular a dificuldade na Ordem Original, coletou-se a dificuldade que cada aluno sentiu na lista de exercícios proposta pelo professor da disciplina. Para isso, aplicou-se a Fórmula 4.7 para cada exercício resolvido pelo aluno.

Para calcular a dificuldade na Ordem Proposta, aplicou-se a Fórmula 4.7 nos exercícios que o método de recomendação desta pesquisa sugere ao aluno.

Para calcular a dificuldade na Ordem ideal, foi utilizado as dificuldades mapeadas na Ordem Original, porém ordenadas em ordem crescente.

A Tabela 5.5 mostra um exemplo das dificuldades nas ordens: original, proposta e ideal, coletadas nesse experimento para o aluno 3702.

Tabela 5.5 – Ordem de dificuldades original, proposta e ideal do aluno 3702

Primeira lista de exercícios opcionais						
-	**Ordem Original**		**Ordem Proposta**		**Ordem Ideal**	
	Exercícios	Dificuldade	Exercícios	Dificuldade	Exercícios	Dificuldade
1º	1800	0.42	2426	1.11	2043	0.05
2º	2049	0.35	1446	0.09	1446	0.09
3º	1446	0.09	1805	0.12	1805	0.12
4º	2043	0.05	1803	0.44	2049	0.35
5º	2425	0.58	2049	0.35	1800	0.42
6º	2426	1.11	1800	0.42	1803	0.44
7º	1801	0.53	1804	0.50	1804	0.50
8º	1802	0.66	1806	1.25	1801	0.53
9º	1803	0.44	1802	0.66	2425	0.58
10º	1805	0.12	2425	0.58	1802	0.66
11º	1804	0.50	1801	0.53	2426	1.11
12º	1806	1.25	2043	0.05	1806	1.25

O propósito desse experimento é comparar a similaridade da Ordem Original $\times$ Ordem Proposta e a Ordem Original $\times$ Ordem Ideal. Para isso, foi utilizado o coeficiente de Kendall. Lembrando que, em uma situação ideal de aprendizagem, o aluno deve sempre iniciar a lista resolvendo os exercícios mais fáceis e ir avançando nos exercícios seguindo uma ordem crescente de dificuldade.

Ao comparar a Ordem Original com a Ordem Ideal, e a Ordem Proposta com a Ordem Ideal, é possível verificar qual ordem de exercício está mais próxima desse ambiente ideal.

Aplicando os valores das dificuldades na Ordem Original, Ordem Proposta e Ordem Ideal para o aluno 3702, apresentados na Tabela 5.5 obtemos o seguinte resultado:

- $\tau(OO, OI) = 0.27$
- $\tau(OP, OI) = 0.15$

Novamente, como já tinha acontecido na Seção 5.4, os resultados para o Aluno 3702 não são os melhores. Pois o $\tau(OP, OI) < \tau(OO, OI)$. Nesse caso, a Ordem Original é mais similar a Ordem Ideal.

Porém, aplicando o experimento aos alunos que resolveram toda a lista de exercícios opcionais do módulo 1, obtemos resultados mais satisfatórios.

- Média aritmética de $\tau(OO, OI) = 0.10$
- Média aritmética de $\tau(OP, OI) = 0.30$

A Tabela 5.6 representa os dados das médias dos índices de dificuldade dos exercícios da primeira lista de exercícios.

Tabela 5.6 – Médias de dificuldades das Ordens: original, proposta e ideal

Primeira lista de exercícios opcionais			
Ordem	**Dificuldade média Ordem original**	**Dificuldade média Ordem proposta**	**Dificuldade média Ordem ideal**
1º	0.02	3.91	0.02
2º	1.93	0.20	0.20
3º	0.57	0.57	0.57
4º	8.96	1.93	1.32
5º	5.49	6.44	1.44
6º	3.91	0.02	1.93
7º	1.44	6.55	3.19
8º	1.32	3.19	3.91
9º	6.44	1.32	5.49
10º	0.20	1.44	6.44
11º	6.55	5.49	6.55
12º	3.19	8.96	8.96

Nesse caso, para maioria dos alunos, a Ordem Proposta é mais similar a Ordem Ideal. Ou seja, aqui existe um ambiente de aprendizagem que na maioria dos casos recomenda exercícios na ordem gradual de dificuldade, recomendando exercícios mais fáceis para mais difíceis.

5.4.2 Respostas às Questões de pesquisa

A seguir, os resultados dos experimentos serão discutidos a fim de responder as questões de pesquisa.

5.4.2.1 Respondendo à Questão de Pesquisa 1

É possível desenvolver um método que forneça aos alunos uma sequência de exercícios com níveis crescente de dificuldade?

A Figura 5.7 mostra que o método proposto nesta pesquisa na maioria das vezes recomenda exercícios aos alunos em uma sequência crescente de dificuldade.

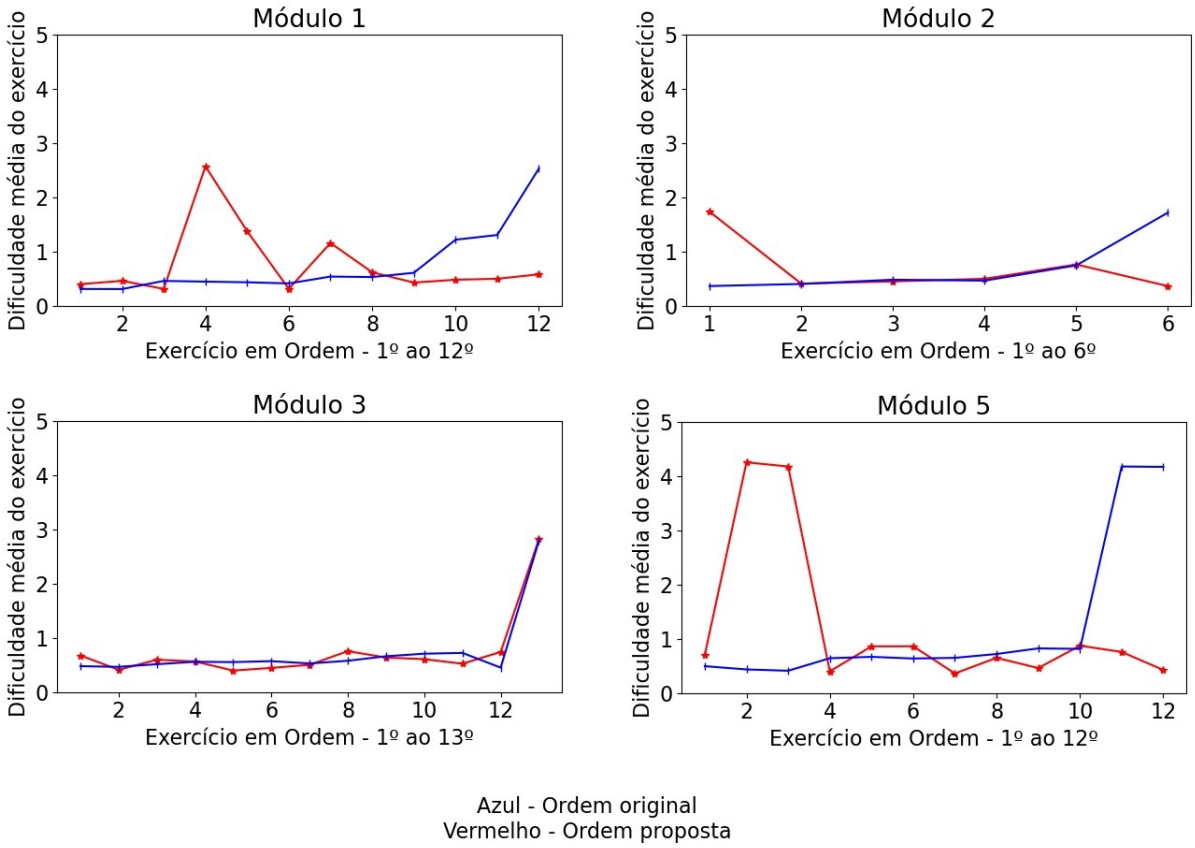


Figura 5.7 – Comparativo entre as médias de dificuldade dos exercícios na ordem proposta pelos professores e ordem recomendada pelo método da pesquisa.

Na Figura 5.7 é apresentado o comparativo entre as médias dos índices de dificuldade das listas de exercícios opcionais 1, 2, 3 e 5. Somente essas listas são apresentadas, pois como já foi mencionado na Seção 5.4, somente os alunos que conseguiram resolver a lista de exercício completa de cada módulo entrou nesse experimento.

É possível notar em todas as listas, que a linha azul, representante das dificuldades que os alunos sentiram ao resolver os exercícios na ordem proposta pelo método de recomendação desta pesquisa, na maioria das vezes é crescente. Em algumas poucas exceções isso não ocorre. Como é o caso da Lista de Exercícios opcionais 3, onde a questão 12 possui um nível de dificuldade médio maior do que a questão 11.

Ao comparar os exercícios na ordem original dos exercícios, isso é, na ordem proposta pelos professores (representados pela linha vermelha), a vantagem em usar questões na ordem proposta pelo método de recomendação desta pesquisa é enorme.

Veja por exemplo, o primeiro exercício da segunda lista de exercícios opcionais propostos pelos professores é o que possui maior dificuldade média. Nesse caso, quando a ordem é a proposta pelo método desta pesquisa, os exercícios sempre possuem uma ordem crescente de dificuldade, salvo no caso do exercício 3 para o 4, pois o exercício 4 possui dificuldade média inferior ao exercício 3. Mesmo assim, a diferença de dificuldade é menor do que 0,2. Em uma escala onde o exercício mais fácil é de 0,4 e o maior difícil é 1,7, essa diferença não é tão significante.

Sendo assim, pode-se afirmar que o método de recomendação de exercícios desta pesquisa fornece uma sequência de exercícios com níveis crescente de dificuldade, em poucas exceções isso não ocorre.

5.4.2.2 Respondendo à Questão de Pesquisa 2

É possível desenvolver um método de recomendação de exercícios que leve em consideração as individualidades dos alunos?

Com base nos dados da Figura 5.8, pode-se afirmar que sim, as recomendações feitas pelo método de recomendação proposto nesta pesquisa levam em consideração as individualidades dos alunos.

Ao utilizar o coeficiente de Kendal para comparar a similaridade entre a Ordem Original de exercícios com a Ordem Proposta pelo método de recomendação, para cada um dos sete módulos, é possível notar que existe uma variação de valores negativos para valores positivos em todas as 7 listas de exercícios opcionais.

Nos gráficos que representam as listas de exercícios opcionais 1 e 2, a imagem da função para alguns intervalos de X é constante em Y. Isso representa que um mesmo exercício foi recomendado em uma mesma ordem para vários alunos.

No caso em específico da lista de exercícios opcionais do módulo 2, um mesmo exercício foi recomendado na mesma ordem de resolução para mais de 65 alunos. Isso geralmente ocorre quando o número de exercícios opcionais é pequeno. A lista de exercícios opcionais do segundo módulo da disciplina, contém somente 6 exercícios opcionais.

Então, analisando em um contexto geral, pode-se afirmar que o método de recomendação proposto nesta pesquisa leva em consideração as individualidades dos alunos, pois quando a quantidade de exercícios opcionais nos módulos é grande as recomendações não

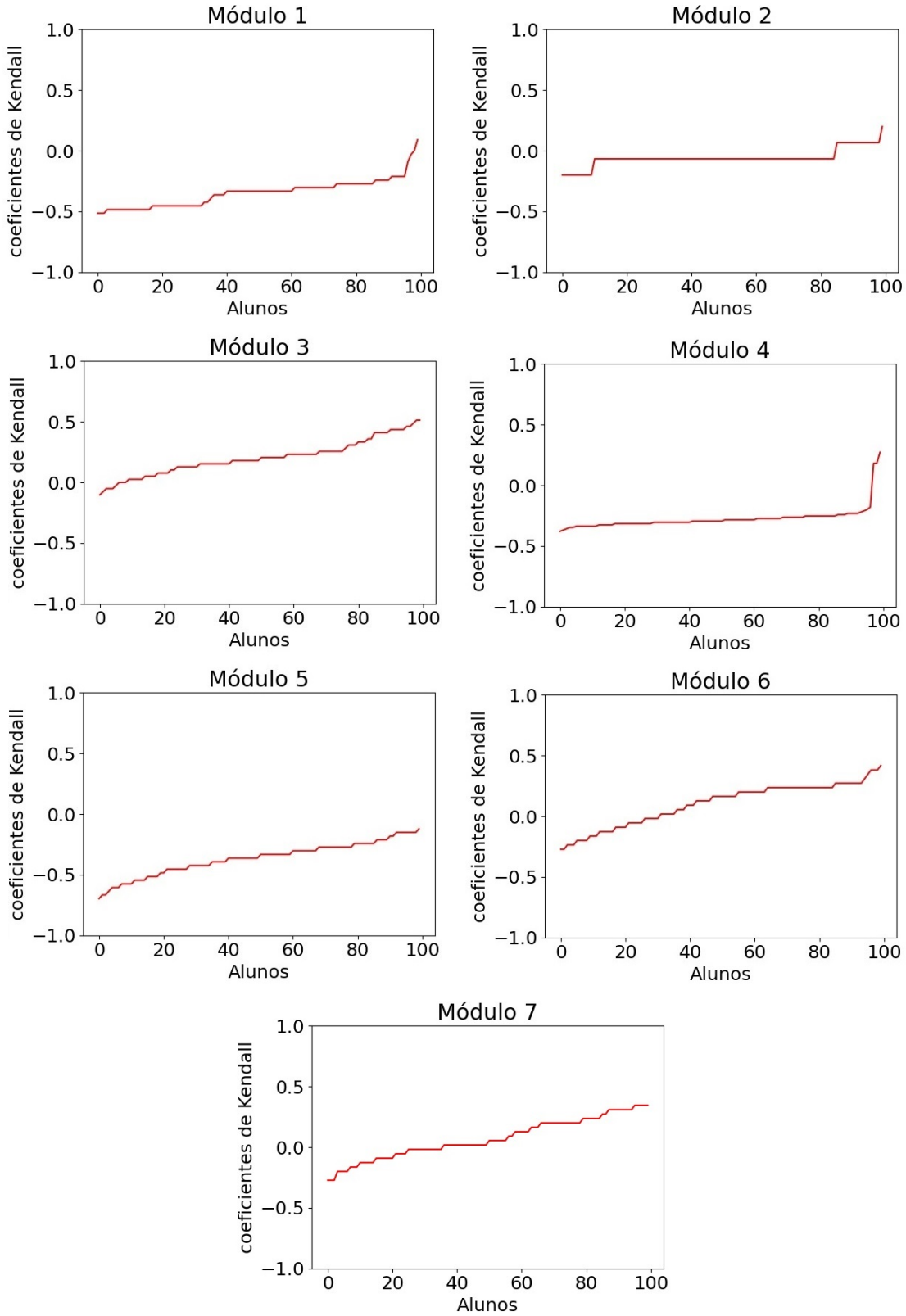


Figura 5.8 – Similaridade entre as listas de exercícios recomendadas aos alunos

serão similares. Ou seja, as listas de exercícios opcionais possuem uma ordem diferente para cada aluno em questão.

É importante frisar que, sem o uso do método de recomendação proposto nesta pesquisa, a estratégia de *scaffolding* não é utilizada, pois é proposto uma mesma ordem de resolução de exercícios para todos os alunos.

5.4.2.3 Respondendo à Questão de Pesquisa 3

É possível desenvolver algum mecanismo automático de recomendação de exercícios que disponha as questões em uma ordem mais adequada do que aquela recomendada pelos professores?

Para responder essa questão de pesquisa, uma variação do experimento realizado na Seção 5.4.1 foi aplicado para cada um dos sete módulos da disciplina de CS1 do semestre 2018.2 da UFAM. Nesse caso, diferente do experimento da Seção 5.4.1, foi feita a coleta dos dados de todos os alunos do GE, incluindo os que não resolveram todos exercícios. Esse procedimento foi feito assim, pois nenhum aluno resolveu por completo as listas de exercícios opcionais dos módulos 4, 6 e 7.

Nesse experimento foi coletada a dificuldade que cada aluno sentiu ao resolver cada exercício opcionais. A dificuldade do aluno é dada pelo somatório dos dez atributos do PP, mencionado anteriormente na Seção 4.2.

Após isso, foi feito o cálculo de média aritmética para cada exercício por ordem de resolução. Ou seja, verificou-se na ordem de exercício proposta pelos professores, qual foi o primeiro exercício resolvidos pelos alunos, e para esse exercício foi calculado a média aritmética dos índices de dificuldade dos exercícios. Isso foi feito para o segundo, terceiro e assim sucessivamente para todos os exercícios dos sete módulos da disciplina. Isso gerou um ranking de dificuldades, aqui denominado de **Dificuldades médias na Ordem Original**.

Após isso, foi gerada a recomendação de exercícios opcionais para todos os alunos do GE. Então, foi calculado a média de dificuldades que os alunos sentiriam caso resolvessem os exercícios na ordem proposta pelo método desta pesquisa. Isso gerou um ranking de dificuldades, aqui denominado de **Dificuldades médias na Ordem Recomendada**.

Finalmente, foi feita a ordenação crescente dos índices de dificuldade na ordem original dos exercícios opcionais, o que gerou um ranking de dificuldades aqui denominado de **Dificuldades médias na Ordem Ideal**.

No Apêndice B são apresentadas 7 tabelas, uma para cada módulo da disciplina de CS1, semestre 2018.2 da UFAM. Essas tabelas representam os dados das dificuldades nas ordens: Original, Recomendada e Ideal; coletados nesse experimento.

A Figura 5.9 mostra o comparativo das dificuldades nas ordens: original, recomendada e ideal; para cada um dos 7 módulos da disciplina de CS1, semestre 2018.2 da UFAM.

Como é possível notar, a ordem de resolução de exercícios proposta pelo método de recomendação desta pesquisa, para os módulos 1, 4 e 6 são muito similares a ordem ideal de exercícios. No caso dos módulos 2, 5 e 7, apesar da ordem de resolução de exercícios não serem tão parecidas com a ordem ideal de exercícios, ainda assim, é uma ordem mais adequada do que a ordem de exercícios proposta pelos professores.

Somente no caso do módulo 3 o método de recomendação proposto nesta pesquisa não recomendou uma ordem de exercícios mais adequada para os alunos resolverem. É importante frisar que, nesse módulo, poucos alunos resolveram os exercícios. O número máximo de resolução de exercícios foi de 44 alunos (primeiro exercício da lista referente ao módulo 3).

Um pequeno número de resoluções pode afetar a fidelidade da avaliação. O ideal é que o método proposto nesta pesquisa seja replicado em um base de dados com mais alunos.

A fim de quantificar a similaridade entre os rankings dos índices de dificuldade dos exercícios e responder a Questão de pesquisa 3, utilizando os dados apresentados no Apêndice B, foi calculado o coeficiente de Kendall entre as médias dos índices de dificuldade na Ordem Original e as médias dos índices de dificuldade na Ordem Ideal, e as médias dos índices de dificuldade na Ordem Recomendada e as médias de dificuldade na Ordem Ideal. A Tabela 5.7 mostra os resultados desse experimento.

A Tabela 5.7 confirma que nos casos das recomendações feitas pelo método desta pesquisa, para os módulos 1, 2, 4, 5, 6 e 7 são mais adequadas do que a ordem de exercícios proposta pelos professores.

5.4.3 Considerações Finais

Neste capítulo foram apresentados os resultados obtidos nos experimentos feitos durante esta pesquisa. Inicialmente, foi apresentado o PP do aluno 3702. Logo após, foi feito o cálculo da similaridade entre os alunos do GC com o aluno 3702.

Tabela 5.7 – Similaridade calculada pelo coeficiente de Kendall para os exercícios nas ordens: original, recomendada e ideal.

Módulo	**coeficiente de Kendall Original $\times$ Ideal**	**coeficiente de Kendall Recomendada $\times$ Ideal**	**Resposta à Q3**
1	-0,09	0,7	Sim
2	-0,2	0,33	Sim
3	0,42	-0,03	Não
4	-0,18	0,75	Sim
5	-0,47	0,25	Sim
6	0,09	0,74	Sim
7	-0,24	0,09	Sim

Após isso, foi apresentado o resultado das recomendações feitas para o aluno 3702. No caso em específico do aluno 3702, as recomendações não foram as melhores possíveis. O aluno 3702 foi considerado um *outlier*, pois os valores dos índices de dificuldade dos exercícios opcionais propostos não seguem uma ordem crescente.

Porém, quando o mesmo experimento foi aplicado em todos os alunos que conseguiram realizar toda a lista de exercício, o experimento foi muito satisfatório, pois na grande maioria dos casos, a ordem recomendada de exercícios opcionais apresentavam dificuldade crescente.

Outro experimento foi conduzido a fim de verificar a similaridade das listas de exercícios recomendados para os alunos. Para mostrar a similaridade entre as listas, foi utilizado cálculo do coeficiente de Kendall. Esse experimento foi satisfatório, pois, apesar de algumas listas de exercícios terem sido iguais para alguns alunos, as ordens dos exercícios variavam para os alunos.

Um último experimento foi conduzido com o objetivo de verificar qual ordem de exercícios é mais similar à ordem ideal de exercícios, sendo que a ordem ideal de exercícios é aquela que o aluno responde primeiro os exercícios mais fáceis seguidos pelo mais difíceis. Nesse caso, novamente para o aluno 3702, os resultados não foram satisfatórios, pois a ordem original dos exercícios opcionais foi mais similar a ordem ideal de exercícios opcionais. Porém, quando esse experimento foi aplicado aos alunos que concluíram a lista de exercícios opcionais, os resultados novamente foram satisfatórios, pois a ordem de exercícios propostos pelo método de recomendação foi mais similar à ordem ideal de exercícios.

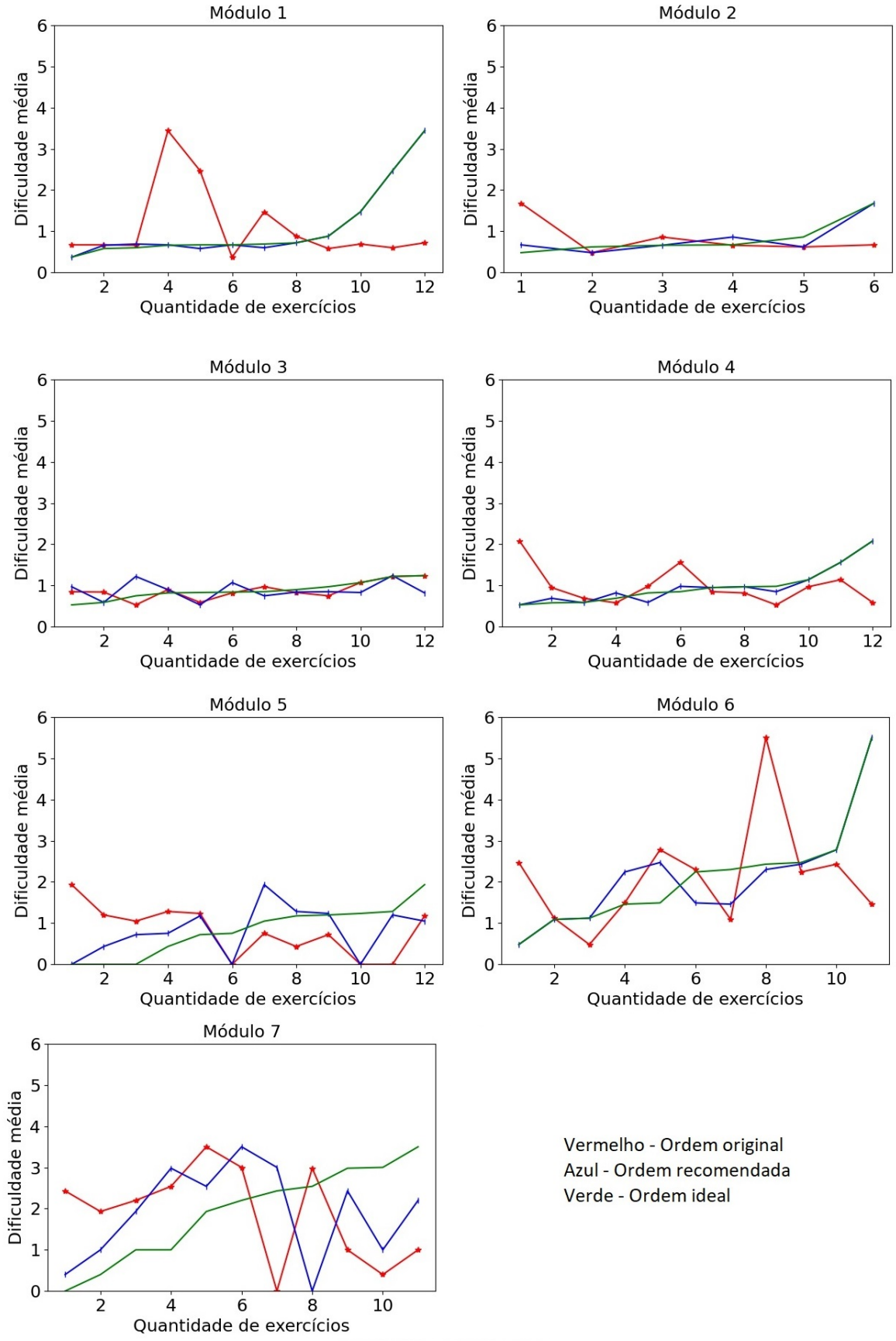


Figura 5.9 – Comparativo das médias de dificuldade dos exercícios dos 7 módulos nas ordens: original, proposta e ideal.

Capítulo 6

Considerações finais

Neste capítulo serão explicitadas as conclusões, as limitações e os trabalhos futuros almejados.

6.1 Conclusões

Este trabalho propôs-se e validou-se um método de recomendação de exercícios baseado em filtragem colaborativa. Para tanto, a) coletaram-se dados sobre os exercícios resolvidos pelos aluno no ambiente de programação do CodeBench; b) com os dados coletados, construiu-se um perfil de programação para cada aluno dos semestres 2018.1 e 2018.2 das turmas de CS1 da UFAM; c) dividiram-se os alunos em 2 grupos, sendo eles: Grupo de Controle (alunos do semestre 2018.1) e Grupo Experimental (alunos do semestre 2018.2); d) para cada aluno do GE, calculou-se a similaridade entre todos os do GC; e) usando as similaridades entre os alunos e os índices de dificuldades dos exercícios opcionais, gerou-se um lista de recomendações de exercícios opcionais. Essa lista tem como principal atributo o valor estimado de dificuldade que os alunos alvos da recomendação sentiriam ao resolver os exercícios opcionais; f) Verificou-se a similaridade entre as recomendações feitas aos alunos GE; g) Verificou-se as médias dos índices de dificuldade dos exercícios opcionais na ordem original e na ordem recomendada pelo método; h) Verificou-se a similaridade entre a ordem de exercícios opcionais originais, ordem de exercícios opcionais proposta pelo método e ordem de exercícios opcionais ideal.

Os objetivos desse trabalho foram alcançados e as questões de pesquisa foram respondidas. Além disso, as soluções para minimizar os índices de reprovação propostas

por [Souza et al. 2016], que são: desenvolvimento de ambientes pedagógicos, resolução de problemas reais e *Scaffolding*; estão presentes no método de recomendação proposto nesta pesquisa.

Os resultados mostram que é possível mapear o índice de dificuldade dos exercícios resolvidos pelo aluno através do perfil de programação. Também é possível fazer recomendações de exercícios na ordem personalizada para os alunos usando a abordagem de recomendação com filtragem colaborativa. Foi possível perceber que na maioria dos casos, as recomendações feitas pelo método proposto neste trabalho oferecem ao aluno um caminho gradual de dificuldade, e que esse caminho também é na maioria das vezes individualizado.

6.2 Limitações

Este trabalho utilizou a filtragem colaborativa para fazer recomendações de exercícios opcionais aos alunos. Por isso, nos casos que o aluno tiver resolvido poucos exercícios, ou seja, o perfil de programação do aluno possuir poucos dados, a qualidade da recomendação não será de boa qualidade. Esse é um problema clássico da filtragem colaborativa.

Outra limitação deste trabalho é que a qualidade da recomendação pode ser afetada quando se tem poucos dados sobre os exercícios opcionais. Isso acontece quando poucos alunos conseguem resolver esses exercícios ou quando o professor cadastra um novo exercício opcional. Para fazer a recomendação outros alunos já devem ter resolvido os exercícios opcionais.

Este trabalho não leva em consideração fatores de contexto pedagógicos, tais como: ambiente de sala de aula, o professor da disciplina, a base de conhecimento que o aluno trouxe do ensino médio, a motivação do aluno com a disciplina de CS1, dentre outros. Tais fatores, podem influenciar os alunos na resolução dos exercícios. Isso pode fazer os dados utilizados para recomendação (Perfil de Programação) ficarem fora do padrão, o que também pode alterar a qualidade da recomendação.

A base de dados utilizada neste trabalho contém o total 645 alunos, todos alunos da UFAM, do ano letivo de 2018. É interessante que o método de recomendação proposto nesta pesquisa seja testado em outras bases de dados de outras instituições de ensino,

pois fatores educacionais relativos a instituições de ensino podem alterar a qualidade da recomendação pelo método proposto.

Outro fato sobre a base de dados é que o GE possui somente 180 alunos. Para que a eficácia do método possa ser comprovada, é recomendado que a quantidade de alunos seja maior. Ainda relativo à base de dados, o número de questões opcionais também pode afetar a qualidade das recomendações. O ideal é que o mesmo experimento seja testado em uma base de dados com o número maior de exercícios opcionais.

6.3 Trabalhos Futuros

Como trabalhos futuros, pretende-se verificar quais os atributos do Perfil de Programação são mais relevantes no momento a recomendação. Nesta pesquisa, todos os atributos possuem o mesmo peso no momento de determinar o índice de dificuldade dos exercícios revolvidos pelos alunos. Uma sugestão é fazer uso de algoritmos de aprendizagem de máquina para determinar os atributos mais relevantes.

Este trabalho utilizou somente a filtragem colaborativa para fazer as recomendações. Para que a qualidade das recomendações possa aumentar, outras abordagens de recomendação devem ser adicionadas ao método aqui proposto, modificando assim o método para abordagem híbrida de recomendação.

Nesta pesquisa, somente dados sobre o aluno foram coletados. Assim como foi feito um perfil de programação do aluno, deveria ser feito um perfil do exercício. Nesse perfil deveria conter informações relevantes sobre o exercício, com isso, a abordagem baseada em conteúdo poderia ser adicionada ao método de recomendação.

As recomendações de exercícios opcionais feitas neste trabalho são *off-line*, ou seja, foram utilizados dados reais dos alunos, porém as recomendações não são feitas em tempo real. Pretende-se utilizar o método aqui proposto em uma turma em andamento, de forma *on-line*. Sendo assim, pode-se coletar a satisfação do aluno sobre a recomendação. Sistemas de avaliação que utilizem por exemplo: *"likes"*, avaliação por estrelas, e outras formas de verificar a satisfação da recomendação comumente utilizados nos sistemas de recomendação podem ser adicionadas caso a recomendação seja em tempo real.

Referências

[Alves and Jaques 2014] Alves, F. P. and Jaques, P. (2014). Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação. *Anais do Simpósio Brasileiro de Informática na Educação*, 25(1):1078–1082.

[Ansari et al. 2017] Ansari, M. H., Moradi, M., Nikrah, O., and Kambakhsh, K. M. (2017). CodERS: A hybrid recommender system for an E-learning system. *Proceedings - 2016 2nd International Conference of Signal Processing and Intelligent Systems, ICSPIS 2016*.

[Brusilovsky and Millán 2007] Brusilovsky, P. and Millán, E. (2007). User models for adaptive hypermedia and adaptive educational systems. In *The adaptive web*, pages 3–53. Springer.

[Caro-Martinez and Jimenez-Diaz 2017] Caro-Martinez, M. and Jimenez-Diaz, G. (2017). Similar users or similar items? comparing similarity-based approaches for recommender systems in online judges. In *International Conference on Case-Based Reasoning*, pages 92–107. Springer.

[Cazella et al. 2009] Cazella, S. C., Reategui, E. B., Machado, M., and Barbosa, J. L. V. (2009). Recomendação de objetos de aprendizagem empregando filtragem colaborativa e competências. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 1.

[Chau et al. 2017] Chau, H., Barria-Pineda, J., and Brusilovsky, P. (2017). Content wizard:concept-based recommender system for instructors of programming courses. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization - UMAP '17*, UMAP '17, pages 135–140, New York, NY, USA. ACM.

[De Oliveira et al. 2013] De Oliveira, M. G., Marques Ciarelli, P., and Oliveira, E. (2013). Recommendation of programming activities by multi-label classification for a formative assessment of students. *Expert Systems with Applications*, 40(16):6641–6651.

[De Paula et al. 2015] De Paula, L. C., De Oliveira Fassbinder, A. G., and Barbosa, E. F. (2015). A recommendation system to support the students performance in programming contests. *Proceedings - Frontiers in Education Conference, FIE*, 2015-February(February).

[Drachsler et al. 2015] Drachsler, H., Verbert, K., Santos, O. C., and Manouselis, N. (2015). Panorama of Recommender Systems to Support Learning. In *Recommender Systems Handbook*, pages 421–451. Springer.

[Dwan et al. 2017] Dwan, F., Oliveira, E., and Fernandes, D. (2017). Predição de Zona de Aprendizagem de Alunos de Introdução à Programação em Ambientes de Correção Automática de Código. (Cbie):1507.

[Francisco et al. 2016] Francisco, R., Júnior, C. P., and Ambrósio, A. P. (2016). Juiz online no ensino de programaçao introdutória-uma revisao sistemática da literatura. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 27, page 11.

[Francisco and Ambrosio 2015] Francisco, R. E. and Ambrosio, A. P. (2015). Mining an online judge system to support introductory computer programming teaching. In *CEUR Workshop Proceedings*, volume 1446.

[Fu et al. 2017] Fu, X., Shimada, A., Ogata, H., Taniguchi, Y., and Suehiro, D. (2017). Real-time learning analytics for C programming language courses. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference on - LAK '17*, LAK '17, pages 280–288. ACM.

[Galvão et al. 2016] Galvão, L., Fernandes, D., and Gadelha, B. (2016). Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 27.

[Gomes et al. 2008] Gomes, A., Henriques, J., and Mendes, A. (2008). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educação, Formação & Tecnologias*, 1:[93–103].

[Hosseini and Brusilovsky 2017] Hosseini, R. and Brusilovsky, P. (2017). A study of concept-based similarity approaches for recommending program examples. *New Review of Hypermedia and Multimedia*, 23(3):161–188.

[Hu and Pu 2009] Hu, R. and Pu, P. (2009). Acceptance issues of personality-based recommender systems. In *Proceedings of the third ACM conference on Recommender systems - RecSys '09*, pages 221–224. ACM.

[Huang et al. 2014] Huang, C.-L., Yeh, P.-H., Lin, C.-W., and Wu, D.-C. (2014). Utilizing user tag-based interests in recommender systems for social resource sharing websites. *Knowledge-Based Systems*, 56:86–96.

[Isinkaye et al. 2015] Isinkaye, F. O., Folajimi, Y. O., and Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16(3):261–273.

[Kendall 1970] Kendall, M. (1970). *Rank Correlation Methods*. Theory and applications of rank order-statistics. Griffin.

[KENDALL 1938] KENDALL, M. G. (1938). A NEW MEASURE OF RANK CORRELATION. *Biometrika*, 30(1-2):81–93.

[Kitchenham 2004] Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

[Klašnja-Milićević et al. 2017] Klašnja-Milićević, A., Ivanović, M., Vesin, B., and Budimac, Z. (2017). Enhancing e-learning systems with personalized recommendation based on collaborative tagging techniques. *Applied Intelligence*, pages 1–17.

[Kurnia et al. 2001] Kurnia, A., Lim, A., and Cheang, B. (2001). Online judge. *Computers & Education*, 36(4):299–315.

[Laisa et al. 2018] Laisa, J., Medeiros, T., Aranha, E., and da Silva, T. R. (2018). Uma revisão sistemática da literatura sobre sistemas de recomendação educacional. *Anais do Computer on the Beach*, pages 751–760.

[Mobasher et al. 2004] Mobasher, B., Jin, X., and Zhou, Y. (2004). Semantically enhanced collaborative filtering on the web. In *Web Mining: From Web to Semantic Web*, pages 57–76. Springer.

[Ng and Bereiter 1991] Ng, E. and Bereiter, C. (1991). Three levels of goal orientation in learning. *Journal of the Learning Sciences*, 1(3-4):243–271.

[Oliveira et al. 2016] Oliveira, M. V., Rodrigues, L. C., and Queiroga, A. (2016). Material didático lúdico: uso da ferramenta scratch para auxílio no aprendizado de lógica da programação. In *Anais do Workshop de Informática na Escola*, volume 22, page 359.

[Pu et al. 2011] Pu, P., Chen, L., and Hu, R. (2011). A user centric evaluation framework for recommender systems. In *Procs. of RecSys'11*, pages 157–164. ACM.

[Ricci et al. 2015] Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2015). *Recommender systems handbook*. Springer.

[Sánchez-Ruiz et al. 2017] Sánchez-Ruiz, A. A., Jimenez-Diaz, G., Gómez-Martín, P. P., and Gómez-Martín, M. A. (2017). Case-based recommendation for online judges using learning itineraries. In *International Conference on Case-Based Reasoning*, pages 315–329. Springer.

[Souza et al. 2016] Souza, D. M., Batista, M. d. S., and Barbosa, E. F. (2016). Problemas e dificuldades no ensino e na aprendizagem de programação: Um mapeamento sistemático. *Revista Brasileira de Informática na Educação*, 24(1):39–52.

[Sudha et al. 2017] Sudha, S., Kumar, A. A., Nagappan, M. M., and Suresh, R. (2017). Classification and recommendation of competitive programming problems using cnn. In *International Conference on Intelligent Information Technologies*, pages 262–272. Springer.

[Toledo 2014] Toledo, R. Y. (2014). An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges. 12(June):51–65.

[Venturi 2015] Venturi, J. J. (2015). *Álgebra vetorial e geometria analítica*. Livrarias Curitiba, www.geometriaanalitica.com.br, 10 edition.

[Vesin et al. 2011] Vesin, B., Ivanovic, M., Klasnja-Milicevic, a., and Budimac, Z. (2011). Rule-based reasoning for altering pattern navigation in Programming Tutoring System. *System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference on*, (January):1–6.

[Vesin et al. 2012] Vesin, B., Ivanović, M., Klašnja-Milićević, A., and Budimac, Z. (2012). Protus 2.0: Ontology-based semantic recommendation in programming tutoring system. *Expert Systems with Applications*, 39(15):12229–12246.

[Vesin et al. 2013] Vesin, B., Klašnja-Milićević, A., Ivanović, M., and Budimac, Z. (2013). Applying recommender systems and adaptive hypermedia for e-learning personalization. *Computing and Informatics*, 32(3):629–659.

[Wasik et al. 2018] Wasik, S., Antczak, M., Badura, J., Laskowski, A., and Sternal, T. (2018). A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)*, 51(1):1–34.

[Yan et al. 2017] Yan, Y., Hara, K., Kazuma, T., and He, A. (2017). A method for personalized c programming learning contents recommendation to enhance traditional instruction. In *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pages 320–327.

[Yera and Martínez 2017] Yera, R. and Martínez, L. (2017). A recommendation approach for programming online judges supported by data preprocessing techniques. *Applied Intelligence*, 47(2):277–290.

[Yu et al. 2015] Yu, R., Cai, Z., Du, X., He, M., Wang, Z., Yang, B., and Chang, P. (2015). The research of the recommendation algorithm in online learning. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):71–80.

[Yu and Chen 2016] Yu, X. and Chen, W. (2016). Research on three-layer collaborative filtering recommendation for online judge. In *Green and Sustainable Computing Conference (IGSC0< 2016 Seventh International*, pages 1–4. IEEE.

[Zaíane 2002] Zaíane, O. R. (2002). Building a recommender agent for e-learning systems. In *Computers in education, 2002. proceedings. international conference on*, pages 55–59. IEEE.

[Ziegler et al. 2005] Ziegler, C.-N. C., McNee, S. M. S., Konstan, J. a. J., and Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web WWW 05*, number January, page 22. ACM.

Apêndice A

Lista de exercícios opcionais

Nesta Seção de Apêndice, encontra-se a lista de exercícios opcionais de disciplina de CS1 da UFAM, referente ao primeiro ao módulo (Variáveis e estrutura de programação sequencial) na ordem de resolução proposta pelos professores.

Os exercícios opcionais propostos pelos professores seguem a ordem:

- Exercício opcional 1800

  Escreva um programa que leia três variáveis reais a, b, c, nesta ordem.

  Como saída, o programa deve imprimir o resultado da seguinte fórmula matemática:

$$\frac{(a^2+b^2+c^2)}{(a+b+c)}$$

  Arredonde os resultados com 07 casas decimais.

- Exercício opcional 2049

  Elementos da divisão inteira

  Escreva um programa que leia dois números inteiros, nesta ordem:

  - X (dividendo)
  - Y (divisor)

  Como saída, imprima os quatro elementos da divisão inteira, nesta ordem:

  - dividendo

  - divisor
  - quociente
  - resto

- Exercício opcional 1446

  Polpa de frutas Michael comprou alguns litros de polpa de frutas, com o objetivo de doar dois terços a uma instituição de caridade e ficar com a terça parte.

  Escreva um programa que leia quantos litros foram comprados. Como saída, apresente a quantidade que ficará com Michael.

  Arredonde os valores com até 03 casas decimais de precisão.

- Exercício opcional 2043

  Média ponderada

  Escreva um programa que leia quatro notas de um aluno. Como saída, imprima a média ponderada, sabendo que os pesos das avaliações são respectivamente: 1, 2, 3 e 4.

  Arredonde os resultados com até 02 casas decimais de precisão.

- Exercício opcional 2425

  Consumo de combustível

  A partir da distância total percorrida (em km) e do total de combustível gasto (em litros), calcule o consumo médio de um automóvel.

  Apresente o valor que representa o consumo médio do automóvel com até 3 casas após a vírgula, seguido da mensagem "km/l".

- Exercício opcional 2426

  Viagem de carro

  Escreva um programa que leia:

  1. O tempo gasto em uma viagem de carro.
  2. A velocidade média do veículo.

  Como saída, o programa deve calcular e apresentar:

  1. A distância percorrida.

2. A quantidade de litros de combustível gastos na viagem.

A autonomia do veículo é de 12.0 km com um litro de combustível.

- Exercício opcional 1801

  Quatro provas com média 5

  Cleosméria fez quatro provas neste período. Escreva um programa que leia as quatro notas obtidas.

  Como saída, imprima, nesta ordem:

  O valor da média aritmética das quatro notas, arredondado em até duas casas decimais.

  A mensagem Aprovação, caso a média seja igual ou superior a 5,0, ou a mensagem Reprovação, caso contrário.

- Exercício opcional 1803

  Dígito verificador

  Objetivo: calcular o dígito verificador de uma conta bancária

  As contas bancárias do banco Gringotes são identificadas por uma sequência de quatro dígitos, mais um dígito verificador.

  Para calcular o dígito verificador, procede-se assim:

  1. Começando da direita para a esquerda, cada dígito da conta é multiplicado por um número, sendo o primeiro por 2, depois por 3, depois por 4, e depois por 5.
  2. O somatório dessas multiplicações é dividido por 11.
  3. O resto desta divisão (módulo 11) é o dígito verificador.

  Escreva um programa que leia um número inteiro de quatro dígitos, correspondente à conta bancária. Como saída, imprima o dígito verificador.

- Exercício opcional 1085

  Ponto médio entre dois pontos no plano cartesiano

  Sejam A e B dois pontos sobre plano cartesiano. As coordenadas do ponto M, o ponto médio do segmento de reta formado por A e B, são dadas por:

$$x_M = \frac{(x_B + x_A)}{2}$$

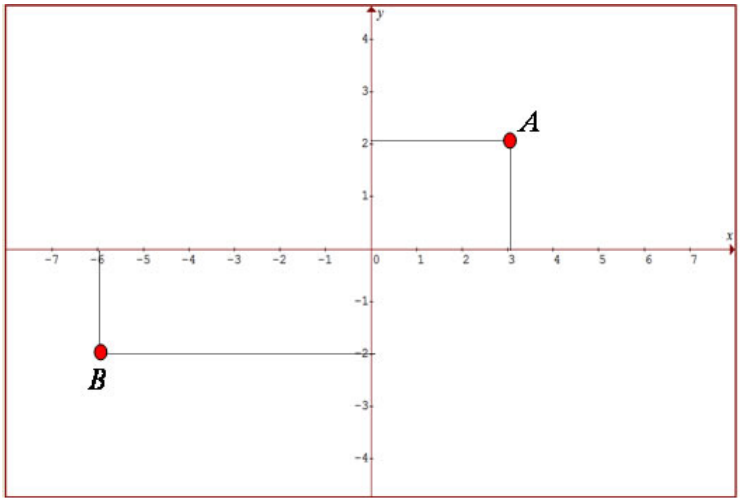
$$y_M = \frac{(y_B + y_A)}{2}$$

Elabore um programa que leia, nesta ordem:

1. as coordenadas do ponto A (xA, yA)
2. as coordenadas do ponto B (xB, yB)

Como saída, imprima o valor das coordenadas xM e yM, nessa ordem, com até uma casa decimal de precisão.

- Exercício opcional 1084
  Distância entre dois pontos no plano cartesiano
  Sejam A e B dois pontos sobre o plano cartesiano, como ilustrado na figura a seguir.



A distância d entre A e B é dada pela seguinte fórmula:

$$d_{AB} = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2}$$

Elabore um programa que leia as coordenadas do ponto A (XA,YA) e as do ponto B (XB,YB), nessa ordem, e imprima no console o valor da distância entre eles com até três casas decimais de precisão.

- Exercício opcional 1806

  Problema do aniversário

  Em um grupo de n pessoas, a probabilidade de que duas delas façam aniversário no mesmo dia do ano é dada pela seguinte expressão:

$$p(n) = 1 - \frac{365!}{(365-n)!} \cdot \frac{1}{365^n}$$

  Escreva um programa que leia o número de pessoas em um grupo e, como saída, informe a probabilidade (em %) de duas pessoas desse grupo fazerem aniversário juntas.

  Apresente o resultado com até duas casas decimais de precisão.

  Curiosidade: Para um grupo maior que 23 pessoas, como uma sala de aula, a probabilidade de que duas pessoas façam aniversário juntas é maior que 50%. Será se isso acontece na sua turma?

Apêndice B

Tabelas com dificuldades médias

Nesta Seção de Apêndice, encontram-se as tabelas geradas a partir da coleta da média dos índices de dificuldades dos exercícios opcionais coletadas para responder à Questão de Pesquisa 3.

Tabela B.1 – Dificuldades médias dos exercícios nas ordens: original, recomendada e ideal - Módulo 1

Exercício	Quant. de alunos que revolveram	Dificuldade média Ordem original	Dificuldade média Ordem recomendada	Dificuldade média Ordem ideal
1º	96	0,67	0,37	0,37
2º	88	0,67	0,66	0,58
3º	86	0,66	0,69	0,60
4º	85	3,44	0,67	0,66
5º	80	2,47	0,58	0,67
6º	77	0,37	0,67	0,67
7º	70	1,47	0,60	0,69
8º	65	0,88	0,72	0,72
9º	63	0,58	0,88	0,88
10º	65	0,69	1,47	1,47
11º	64	0,60	2,47	2,47
12º	64	0,72	3,44	3,44

Tabela B.2 – Dificuldades médias dos exercícios na ordem original e na ordem proposta pelo método - Módulo 2

Exercício	Quant. de alunos que resolveram	Dificuldade média Ordem original	Dificuldade média Ordem recomendada	Dificuldade média Ordem ideal
1º	50	1,68	0,67	0,48
2º	53	0,48	0,48	0,62
3º	147	0,86	0,66	0,66
4º	53	0,66	0,86	0,67
5º	53	0,62	0,62	0,86
6º	52	0,67	1,68	1,68

Tabela B.3 – Dificuldades médias dos exercícios na ordem original e na ordem proposta pelo método - Módulo 3

Exercício	Quant. de alunos que resolveram	Dificuldade média Ordem original	Dificuldade média Ordem recomendada	Dificuldade média Ordem ideal
1º	44	0,85	0,97	0,53
2º	39	0,84	0,59	0,59
3º	40	0,53	1,22	0,75
4º	42	0,90	0,90	0,82
5º	43	0,59	0,53	0,83
6º	29	0,82	1,07	0,84
7º	32	0,97	0,75	0,85
8º	35	0,83	0,84	0,90
9º	40	0,75	0,85	0,97
10º	28	1,07	0,83	1,07
11º	26	1,22	1,24	1,22
12º	27	1,24	0,82	1,24

Tabela B.4 – Dificuldades médias dos exercícios na ordem original e na ordem proposta pelo método - Módulo 4

Exercício	Quant. de alunos que resolveram	Dificuldade média Ordem original	Dificuldade média Ordem recomendada	Dificuldade média Ordem ideal
1º	12	2,08	0,53	0,53
2º	7	0,95	0,69	0,58
3º	2	0,69	0,58	0,59
4º	3	0,58	0,82	0,69
5º	6	0,98	0,59	0,82
6º	11	1,56	0,98	0,85
7º	9	0,85	0,95	0,95
8º	4	0,82	0,97	0,97
9º	1	0,53	0,85	0,98
10º	8	0,97	1,14	1,14
11º	10	1,14	1,56	1,56
12º	5	0,59	2,08	2,08

Tabela B.5 – Dificuldades médias dos exercícios na ordem original e na ordem proposta pelo método - Módulo 5

Exercício	Quant. de alunos que resolveram	Dificuldade média Ordem original	Dificuldade média Ordem recomendada	Dificuldade média Ordem ideal
1º	7	1,93	0,00	0,00
2º	11	1,20	0,43	0,00
3º	11	1,05	0,72	0,00
4º	8	1,28	0,75	0,43
5º	9	1,23	1,18	0,72
6º	1	0,00	0,00	0,75
7º	4	0,75	1,93	1,05
8º	2	0,43	1,28	1,18
9º	2	0,72	1,23	1,20
10º	6	0,00	0,00	1,23
11º	10	0,00	1,20	1,28
12º	2	1,18	1,05	1,93

Tabela B.6 – Dificuldades médias dos exercícios na ordem original e na ordem proposta pelo método - Módulo 6

Exercício	Quant. de alunos que resolveram	Dificuldade média Ordem original	Dificuldade média Ordem recomendada	Dificuldade média Ordem ideal
1º	5	2,47	0,48	0,48
2º	13	1,12	1,09	1,09
3º	13	0,48	1,12	1,12
4º	13	1,49	2,24	1,46
5º	5	2,78	2,47	1,49
6º	5	2,30	1,49	2,24
7º	6	1,09	1,46	2,30
8º	2	5,50	2,30	2,43
9º	4	2,24	2,43	2,47
10º	4	2,43	2,78	2,78
11º	3	1,46	5,50	5,50

Tabela B.7 – Dificuldades médias dos exercícios na ordem original e na ordem proposta pelo método - Módulo 7

Exercício	Quant. de alunos que resolveram	Dificuldade média Ordem original	Dificuldade média Ordem recomendada	Dificuldade média Ordem ideal
1º	4	2,43	0,40	0,00
2º	4	1,93	1,00	0,40
3º	3	2,20	1,93	1,00
4º	3	2,54	2,98	1,00
5º	2	3,50	2,54	1,93
6º	2	3,00	3,50	2,20
7º	1	0,00	3,00	2,43
8º	4	2,98	0,00	2,54
9º	1	1,00	2,43	2,98
10º	39	0,40	1,00	3,00
11º	1	1,00	2,20	3,50