Classificação de dificuldade de questões de programação com base na inteligibilidade do enunciado

**Pedro H. C. dos Santos, Leandro S. G. Carvalho,
Elaine H. T. Oliveira, David B. F. de Oliveira**

Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Av. General Rodrigo Octávio, 6200 – Coroado I 69077-000 – Manaus, AM – Brasil

{phcs,galvao,elaine,david}@icomp.ufam.edu.br

Resumo. *Em grandes turmas de disciplinas de programação mediadas por juízes online, é preciso aleatorizar as questões selecionadas para testes de avaliação, a fim de evitar comportamento desonesto. O problema de uma aleatorização ingênua é a falta de equidade com respeito ao grau de dificuldade das questões. Assim, este artigo propõe um método para classificar a dificuldade de questões de programação com base em métricas de inteligibilidade de texto extraídas a partir de seus enunciados. Foram analisadas 450 questões, respondidas por 800 alunos em uma disciplina de introdução à programação entre os períodos letivos de 2017 a 2018. Verificou-se que uma abordagem baseada apenas nas métricas de inteligibilidade é subótima para a classificação de dificuldade de questões, porém não irrelevante.*

Abstract. *In large programming classes mediated by online judges, one must randomize the chosen questions for evaluation tests, in order to avoid dishonest behavior. The problem of a naive randomization is the lack of equity in respect to the degree of difficulty of the questions. Thus, this paper proposes a method to classify the difficulty of programming questions based on textual intelligibility metrics extracted from their statements. In total, 450 questions were analyzed, and those were answered by 800 students in a introduction to programming discipline between the academic periods of 2017 to 2018. It was verified that an approach based only on the intelligibility metrics is suboptimal for the classification of difficulty of questions, but not irrelevant.*

1. Introdução

Juízes online, ou ambientes de correção automática de código (ACAC), são sistemas online que testam códigos de programação submetidos por estudantes e os avaliam quanto à sua corretude. Dessa forma, eles proveem feedback rápido e automático ao estudante a respeito de problemas de programação cadastrados por professores. Por conta disso, eles têm sido cada vez mais adotados por universidades do Brasil e do mundo.

Porém, o uso desses juízes online para avaliação de aprendizagem em disciplinas requer um grande número de questões disponíveis em seus bancos, a fim de garantir que alunos diferentes resolvam questões distintas, a fim de se evitar conluio entre os estudantes. Além disso, essas questões devem estar classificadas de acordo com a dificuldade, a fim de promover equidade entre os exercícios atribuídos a cada aluno.

DOI: 10.5753/cbie.sbie.2019.1886

Uma forma de medir a dificuldade de questões em juízes online é pedir que o instrutor classifique a dificuldade da questão no momento do cadastro. O problema com essa abordagem é que ela consome tempo e introduz o viés da experiência pessoal de cada instrutor no ensino de programação [Barbosa et al. 2017]. Ainda que essa experiência fosse homogênea, a dificuldade de uma questão pode variar também conforme o tipo de turma: turmas avançadas ou iniciantes, turmas de cursos que usam a computação como atividade meio ou como atividade fim, entre outros fatores.

Outra abordagem seria estimar a dificuldade a partir de métricas de software extraídas do código de solução cadastrado pelos instrutores [Whalley and Kasto 2014, Elnaffar 2016]. Porém, muitas questões cadastradas no juiz online utilizado como base deste estudo possuem apenas o enunciado, sem soluções de referência.

Desse modo, este trabalho buscou elaborar um método para classificar a dificuldade de questões em uma disciplina de introdução a programação usando como base o texto do enunciado da questão. Esse é um dos campos obrigatórios em uma questão de programação, que nenhum instrutor pode se furtar a preencher.

As seguintes questões de pesquisa nortearam este trabalho:

1. **QP1:** Como métricas baseadas em processamento de texto podem ser usadas para classificar a dificuldade de questões de introdução a programação?
2. **QP2:** Existe correlação estatística entre as informações extraídas do texto dos enunciados das questões com métricas de uso do juiz online durante o processo de resolução das questões por parte dos estudantes?

2. Trabalhos relacionados

Na literatura, algumas abordagens utilizam a opinião de professores experientes para classificar a dificuldade de questões de programação. Por exemplo, [Llana et al. 2012] utiliza a resolução dos exercícios pelos professores para medir a dificuldade de problemas, enquanto que [Meisalo et al. 2004] compara a dificuldade estimada pelo instrutor à dificuldade real sentida pelos alunos ao resolver os problemas. Apesar de válidos, esses métodos possuem o problema natural de inserir viés na classificação das questões.

O problema da subjetividade na classificação é demonstrado por [Sheard et al. 2011], em que vários tutores classificaram, de maneira independente, um conjunto de exercícios de programação de acordo com o grau de dificuldade percebida (baixo, médio, alto). Após isso, os tutores discutiram entre si para debater as classificações conflitantes. Como resultado, verificou-se pouca concordância entre os tutores (cerca de 43%), de modo que não foi possível chegar a um consenso sobre a dificuldade de cada questão.

Por outro lado, trabalhos que utilizam métodos objetivos para tentar categorizar questões de programação tendem a realizar uma análise manual das questões, por meio da separação dos exercícios entre os conceitos de programação envolvidos na resolução do problema. Essa abordagem é adotada por [Cherenkova et al. 2014], em que as taxas de acerto dos estudantes por questão são comparadas com os conceitos exigidos em cada problema. Nesse trabalho, observou-se uma dificuldade acentuada sentida por alunos de disciplinas introdutórias em problemas envolvendo laços de repetição com manipulação de vetores.

O trabalho de [Cherenkova et al. 2014] destaca também a possibilidade de a dificuldade de uma questão de programação vir da compreensão textual do enunciado. Foram consideradas métricas quantitativas, como o comprimento dos textos, e outras características, como o vocabulário; porém, conclusões não foram extraídas devido à falta de resultados. O presente trabalho buscou também encontrar relações entre a complexidade dos enunciados das questões e a dificuldade encontrada pelos alunos durante a resolução. Porém, foram utilizadas outras métricas quantitativas para categorizar os enunciados.

O uso de métricas de inteligibilidade para agrupamento de textos quanto à sua complexidade é tema do artigo de [Scarton and Aluisio 2009], em que textos de diferentes fontes (e consequentemente de diferentes complexidades) são comparados utilizando diversas métricas de processamento de linguagem natural. Esse trabalho revela que é possível distinguir textos com público alvo infantil (ou seja, textos mais simples) daqueles com público alvo adulto. Entre as métricas de inteligibilidade textual abordadas, destaca-se a Flesch-Kincaid, que foi adaptada para o português pelos autores. Assim, o trabalho atual buscou utilizar esse arcabouço de métricas de inteligibilidade de textos em geral para classificar a dificuldade de exercícios de programação a partir de seus enunciados.

A abordagem focada no texto dos enunciados utilizada no presente trabalho é semelhante à utilizada por [Cherenkova et al. 2014], porém se difere quanto às métricas de pesquisa e as ferramentas utilizadas. A pesquisa atual buscou abordar os textos dos enunciados a partir de métricas quantitativas mais profundas e comparar essas métricas com dados de uso de estudantes, a fim de validar o uso dessas estatísticas. Essa abordagem não sofre do problema de subjetividade encontrado por [Sheard et al. 2011] e também utiliza a ideia de agrupar textos a partir de métricas de inteligibilidade do trabalho de [Scarton and Aluisio 2009].

3. Métricas de Inteligibilidade Textual

3.1. Índice Flesch

A abordagem selecionada nesta pesquisa para uma classificação das questões quanto à dificuldade foi uma baseada em calcular a complexidade dos textos dos enunciados de cada questão. Essa complexidade foi calculada utilizando métricas de inteligibilidade textual definidas na literatura, com destaque para a *Flesch Reading Ease*. Essa métrica está entre as mais divulgadas no Brasil [Scarton and Aluisio 2009], de modo que possui uma adaptação já estabelecida para a língua portuguesa. O índice Flesch é calculado segundo a adaptação de [Martins et al.], em que o resultado da fórmula é adicionado a 42, o número médio em que o índice Flesch varia entre os textos em português dos textos em inglês [Scarton and Aluisio 2009]. Esse índice possui valores que variam entre 0 e 100, com valores menores representando textos considerados mais complexos quanto à inteligibilidade. Assim, fórmula para o índice Flesch adaptado é:

Índice Flesch = 248,835 - (1,015 * ASL) - (84,6 * ASW)

sendo ASL o número de palavras dividido pelo número de sentenças, e ASW o número de sílabas dividido pelo número de palavras em um dado trecho de texto.

3.2. Métricas do Coh-Metrix-Port

A ferramenta Coh-Metrix-Port é resultado do trabalho de [Scarton and Aluisio 2010], em que uma outra biblioteca de nome Coh-Metrix, originalmente criada para computar es-

tatísticas potencialmente relevantes para a compreensão de textos em inglês é adaptada para a língua portuguesa. A adaptação para o português do Coh-Metrix-Port utiliza de técnicas de processamento de linguagem natural e aprendizagem de máquina para a extração de atributos a partir de textos escritos na língua portuguesa. A ferramenta em questão possui 41 métricas, que capturam desde elementos simples como comprimento do texto até mais complexos, como incidências de advjetivos, advérbios e etc.

4. Contexto da pesquisa

A presente pesquisa foi realizada a partir dos dados do ambiente de correção automática de código (ACAC) CodeBench[1] da Universidade Federal do Amazonas (UFAM). Porém, para garantir um conjunto homogêneo de dados, foram consideradas apenas questões resolvidas por alunos da disciplina de introdução à programação (IPC) . Ela é ministrada para dezessete cursos de engenharia e ciências exatas na UFAM, utilizando uma metodologia híbrida de ensino-aprendizagem adotada por todos os professores.

A nota da disciplina resulta de dois tipos de atividade:

1. **Exercícios práticos (avaliação formativa)** – os exercícios práticos são um conjuntos de questões disponibilizados aos alunos para que esses resolvam os problemas sem restrições. Ou seja, os estudantes podem realizar as tarefas em grupos, compartilhando respostas. Ao fim, os exercícios práticos somam 7 conjuntos distintos de questões, e eles correspondem a 0,9% da nota final dos estudantes.
2. **Avaliações presenciais (avaliação somativa)** – as avaliações presenciais correspondem aos conjuntos de exercícios com o objetivo de avaliar o conhecimento dos estudantes em um ambiente controlado. Essas avaliações presenciais não permitem que os estudantes compartilhem respostas entre si, logo o conhecimento de cada aluno é testado de maneira individual. Essas avaliações somam 7 conjuntos de exercícios e cada uma corresponde a aproximadamente 4% a 12% da nota final.

O ACAC utilizado nesta pesquisa disponibiliza uma IDE (*Integrated development environment*) para resolução dos exercícios. Ele coleta dados de uso dos estudantes, como o tempo de solução de cada questão, os caracteres pressionados, se um dado aluno acertou ou não uma questão, etc. Esses dados são armazenados de forma anônima em arquivos de log. Neste trabalho, foram utilizados dois conjuntos de dados. O primeiro corresponde aos logs de todas as ações de interação dos estudantes com a IDE. O outro contém as informações das questões realizadas pelos mesmos estudantes.

Foram utilizadas apenas questões aplicadas em exames presenciais porque havia controle para minimizar comportamento desonesto. Nos exercícios práticos, era possível que os estudantes compartilhassem entre si as respostas, de modo que sua inclusão neste estudo poderia comprometer a validade.

5. Metodologia

Esta seção descreve como foram coletadas as variáveis independentes (métricas de inteligibilidade textual) e as variáveis dependentes (métricas de dificuldade de resolução de exercícios), de modo a se buscar uma relação entre elas.

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

5.1. Variáveis dependentes (de resposta)

Na falta de uma definição formal e absoluta de "dificuldade de uma questão", optou-se por extrair métricas a partir da experiência dos estudantes durante o processo de resolução das questões, que guardam relação com o conceito de dificuldade. Por exemplo, podemos não saber como classificar a dificuldade uma questão que leva 10 min em média para ser resolvida. Mas podemos afirmar que ela tem menor dificuldade que outra em que os alunos levaram 30 min em média para resolver.

Dessa forma, foram utilizados as seguintes variáveis para estimar a dificuldade de uma determinada questão:

- **número médio de tentativas**: quantidade média de testes que os estudantes fizeram do código de solução com o console da IDE, sem submeter à correção.
- **número médio de submissões**: quantidade média de tentativas dos estudantes em submeter seu código para a correção automática.
- **taxa de acerto**: razão entre o número de estudantes que acertaram a questão e o número de estudantes que receberam a questão em seu teste.
- **tempo de solução**: para cada estudante, corresponde à soma dos tempos de interação com a IDE durante a solução de um determinado problema. Ou seja, não são contados o tempo dispendido em outras questões.

5.2. Variáveis independentes (preditoras)

Como os índices de inteligibilidade são as informações existentes antes de os alunos realizarem as questões, eles são as variáveis preditoras da pesquisa. Com exceção do índice Flesch, todas as outras métricas foram extraídas utilizando a ferramenta Coh-Metrix-Port.

Como a biblioteca Coh-Metrix-Port disponibiliza um grande número de métricas, como detalhado na Seção 3.2, foram escolhidas apenas as seguintes: incidência de adjetivos, de advérbios, de pronomes, de verbos e de operadores lógicos.

O termo incidência representa a proporção do item em relação ao texto original. A incidência de operadores lógicos representa a proporção de operadores como "e", "se" e "ou" no texto analisado. O uso dessas métricas é discutido e validado no trabalho de [Scarton and Aluisio 2009].

5.3. Definição dos métodos utilizados

A dificuldade dos exercícios foi classificada adotando-se as seguintes etapas:

1. Extração dos índices de inteligibilidade a partir dos textos.
2. Extração de indicadores de dificuldade a partir dos logs dos alunos.
3. Discretização dos indicadores de dificuldade.
4. Balanceamento do conjunto de dados de treino.
5. Indução dos modelos de classificação.
6. Avaliação dos modelos gerados

A extração dos índices de inteligibilidade é o processo de calcular o índice de Flesch para os enunciados de cada questão e de utilizar as ferramentas do Coh-Metrix-Port para calcular os outros índices mencionados na Seção 3. Esses índices de inteligibilidade serão utilizados para gerar o modelo de classificação explicado adiante.

A extração dos indicadores de dificuldade é o processo de extrair os dados de uso gerados pelos alunos durante a utilização do juiz online. Esses dados foram extraídos anonimamente.

O processo de balanceamento do conjunto de dados das questões foi realizado utilizando a técnica SMOTE (*Synthetic Minority Over-sampling Technique*) [Chawla et al. 2002]. A técnica citada permite o balanceamento de um conjunto de dados em que uma classe ocorre com muito mais ocorrência do que outras. O processo é baseado em uma sobreamostragem controlada das instâncias de classes minoritárias.

A classificação quanto à dificuldade dos exercícios foi realizada fazendo-se uso dos modelos de Máquinas de Vetores de Suporte (*Support Vector Machines – SVM*) e do modelo ensemble de Floresta Aleatória (*Random Forest*). Os hiperparâmetros dos dois modelos (o valor "C" e o número de estimadores, respectivamente) foram otimizados segundo o algoritmo Grid search, ou seja, uma otimização exaustiva dentro de um intervalo $\{x_0, x_1, ..., x_n\}$. No caso, o intervalo escolhido foi de $\{0,001; 0,01; ...; 1000\}$.

Finalmente, foi utilizada a técnica de validação cruzada (*cross-validation*) para testar a robustez do modelo induzido. Essa técnica foi aplicada realizando validação cruzada com 5-subconjuntos. Além disso, um modelo com as questões aplicadas no primeiro semestre de 2017 foi induzido e comparado com as categorias das questões do primeiro semestre de 2018, para testar como as variações entre o desempenho dos estudantes de cada ano pode influenciar na classificação final.

5.4. Discretização dos indicadores de dificuldade para modelos de classificação

Para poder realizar um trabalho de classificação em cima dos dados mencionados anteriormente, a variável dependente "Porcentagem de acerto" foi discretizada em um conjunto de categorias, já que os valores originais eram todos numéricos. A discretização segue a classificação de "índice de facilidade", adotado pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) no Enade [Inep: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira 2017].

Tabela 1. Classificação de questões por índice de facilidade. Fonte: [Inep: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira 2017]

Porcentagem de acerto	Classificação
$\geq$ 0,86	Muito fácil
0,61 a 0,85	Fácil
0,41 a 0,60	Médio
0,16 a 0,40	Difícil
$\leq$ 0,15	Muito difícil

Além disso, para amenizar problemas com o desbalanceamento do conjunto de dados, as classes "Muito difíceis" e "Difíceis" foram unidas. Esse problema ocorreu devido ao fato de poucas questões possuírem taxa de acerto menor que 15%, assim essa classe possui poucas instâncias.

6. Resultados e Análise

6.1. Caracterização da amostra

As questões utilizadas neste estudo foram aplicadas durante as avaliações presenciais do primeiro período letivo de 2017 e de 2018 na disciplina IPC, ministrada para 11 dos 17 cursos de graduação da área de engenharia e ciências exatas (*non-majors*). Essas questões foram resolvidas por 800 estudantes (403 em 2017/1 e 397 em 2018/1) ativos no juiz online CodeBench, como mostra a Tabela 2. O conteúdo das questões abrangiam estruturas sequenciais, estruturas condicionais, laços de repetição, vetores e matrizes.

Os segundos semestres de 2017 e de 2018 foram desconsiderados devido ao baixo volume de estudantes matriculados (cerca de 400 no total) e também devido ao fato de o nível dos exercícios aplicados durante esses períodos de tempo serem diferente dos aplicados nos primeiro semestres, assim a análise das questões não ficaria uniforme.

Além disso, foi aplicado um filtro nas 450 questões, a fim de se extrair apenas aquelas com mais de 30 respostas (ou seja, mais de 30 estudantes realizando a mesma questão). Esse filtro retornou 187 exercícios, com uma média de 39 respostas por questão.

Tabela 2. Distribuição dos dados extraídos

Indicador	Período letivo		Total
	2017/1	2018/1	
Número de questões (inicial)	235	215	450
Número de questões (mais de 30 respostas)	90	97	187
Número de estudantes	403	397	800

A técnica SMOTE foi aplicada no conjunto de dados das questões devido ao desbalanceamento da base de informações. A classe “Fácil” possuia muitas instâncias em comparação com as outras, então classificadores gerados a partir desse conjunto de dados são prejudicados. O SMOTE realizou a transformação exibida na Figura 1.

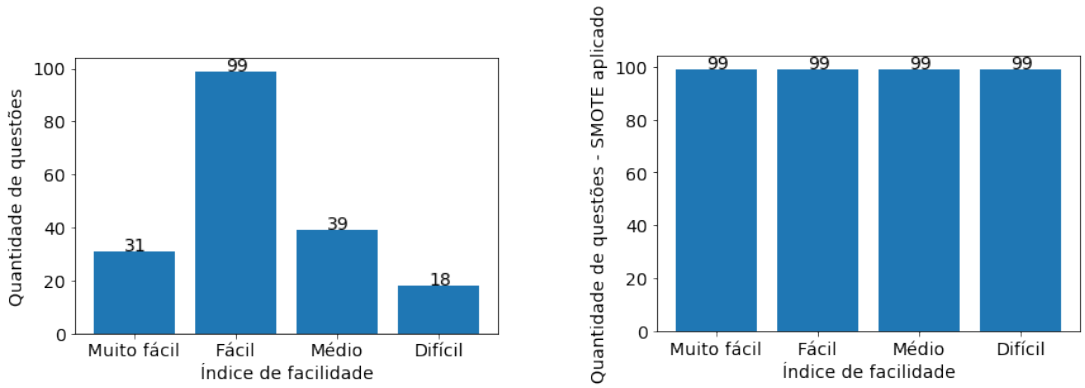

Figura 1. Distribuição das questões quanto aos índices de facilidade

6.2. Resultados da classificação e discussão

A abordagem do presente trabalho para classificar questões baseia-se nos dados dos enunciados dessas. Assim, foram testadas as correlações entre as variáveis independentes com

as dependentes da pesquisa.

O cálculo foi feito segundo a correlação de Pearson e gerou o mapa de calor da Figura 3. É possível observar que os indicadores de dificuldade possuem baixa correlação estatística com os índices de inteligibilidade. As únicas exceções são da incidência de operadores lógicos com o número médio de submissões, com uma correlação de 0,27 e o índice Flesch com o tempo de solução, com uma correlação negativa de -0,27. Ainda assim, essas correlações são baixas.

Os resultados obtidos foram gerados a partir do uso dos modelos de classificação mencionados na Seção 5.3. Para o modelo de Máquinas de Vetores de Suporte, o hiperparâmetro "C" que gerou os melhores resultados foi de 0.001. Esse valor gerou acurácia de 73%. Por outro lado, o melhor número de estimadores para o modelo de Floresta Aleatória foi de 60, que atingiu uma acurácia de 75%.

Os modelos utilizados obtiveram resultados consideráveis, já que os dois classificadores foram capazes de acertar a classe de mais de metade do conjunto de teste com precisão razoável. Assim, conclui-se que os índices de inteligibilidade podem explicar parte da dificuldade das questões. Porém, essa acurácia também não representa integralmente como os modelos gerados estão generalizando o conjunto de dados de entrada. Para interpretar melhor os resultados, foram geradas as matrizes de confusão da Figura 2. Cada matriz foi gerada utilizando modelos com 25% das questões do conjunto de dados como dados de teste, e os outros 75% como de treino, escolhidas de maneira aleatória.

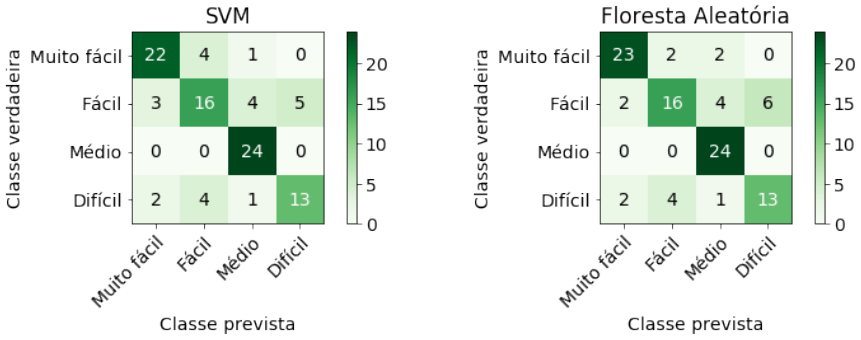

Figura 2. Matrizes de confusão para os classificadores SVM e Floresta Aleatória

A diagonal principal representa os casos em que os classificadores escolheram a classe certa para um dado exemplo. Qualquer outro valor que não esteja na diagonal representa o número de erros cometidos pelo classificador. As matrizes de confusão permitem observar que os classificadores não sofreram problemas com o balanceamento da base, já que cada classe foi classificada de maneira correta com boa precisão.

6.3. Ameaças à validade

A abordagem escolhida na presente pesquisa para a classificação de questões de acordo com a dificuldade é baseada apenas nos índices de inteligibilidade. Porém, a definição de dificuldade possui alto caráter subjetivo. Assim, não existe maneira definitiva de determinar o que deve ser considerado para classificar questões. O foco do presente trabalho foi utilizar os indicadores de dificuldade para representar a dificuldade, porém outras abordagens de classificação podem utilizar outros indicadores.

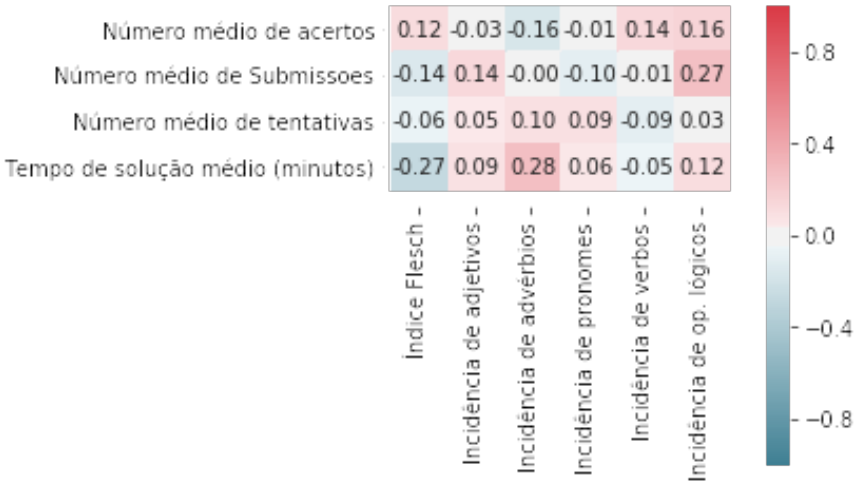

Figura 3. Correlações entre indicadores de dificuldade de questões e indicadores de inteligibilidade textual

Por outro lado, os índices de inteligibilidade considerados são formados a partir da estrutura sintática dos textos. Isso é uma limitação da abordagem do presente trabalho, devido ao fato de métricas baseadas na forma do texto não captarem informações que métricas baseadas em semântica poderiam captar [Scarton and Aluisio 2009].

Além disso, considerar apenas os índices de inteligibilidade para classificar os textos limita a capacidade da classificação de dificuldade. Isso ocorre devido ao fato de a interpretação dos enunciados ser apenas parte da experiência dos estudantes do processo de resolução de uma questão. Efetivamente gerar uma solução para as questões representa um conjunto de dificuldades não captadas integralmente pelos índices de inteligibilidade.

Por fim, o fato do conjunto de dados ser desbalanceado de acordo com as classes dificulta a criação de um classificador capaz de generalizar a classificação de questões.

7. Conclusão e trabalhos futuros

No presente artigo foi verificada a viabilidade de se utilizar métricas de inteligibilidade para a classificação de dificuldade de questões de programação, a fim de facilitar o trabalho de instrutores na geração de avaliações equilibradas. Os resultados obtidos pelos classificadores utilizados não foram satisfatórios, devido ao fato de os índices de inteligibilidade utilizados não serem capazes de representar bem os indicadores de dificuldade escolhidos. Porém, ainda existem relações relevantes presentes entre os dois.

Trabalhos futuros são direcionados a outra abordagem para a classificação, que considera não os textos dos enunciados das questões, mas sim modelos de solução criados por instrutores. Nessa abordagem, cada questão tem uma solução padrão criada por instrutores e a complexidade dessa solução seria utilizada para medir a dificuldade da questão em si. Porém, apesar dessa abordagem aproveitar dados de fora do contexto da pesquisa atual, é possível ainda um trabalho em que as duas metodologias são utiliza-

das em conjunto para gerar um modelo com múltiplas informações sobre o problema de classificação de questões quanto à dificuldade.

Agradecimentos

Os autores agradecem ao apoio prestado pela Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM por meio do Edital N. 002/2018 - Universal Amazonas.

Referências

Barbosa, A., Costa, E., and Brito, P. (2017). Uma abordagem adaptativa para gerar agrupamento de códigos em disciplinas de programação introdutória. In *Simpósio Brasileiro de Informática na Educação*, volume 28, page 1427.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*.

Cherenkova, Y., Zingaro, D., and Petersen, A. (2014). Identifying challenging cs1 concepts in a large problem dataset. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 695–700.

Elnaffar, S. (2016). Using software metrics to predict the difficulty of code writing questions. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, pages 513–518. IEEE.

Inep: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (2017). Relatório Síntese de Área – Ciência da Computação.

Llana, L., E., M.-M., and Pareja-Flores (2012). Flop, a free laboratory of programming. *In Proceedings of the 12th Koli Calling International Conference on Computing Education Research*.

Martins, T. B., Ghiraldelo, C. M., Nunes, M. G. V., and Oliveira Junior, O. N. Readability formulas applied to textbooks in Brazilian Portuguese. *Notas do ICMC*, 28.

Meisalo, V., Sutinen, E., and Torvinen, S. (2004). Classification of exercises in a virtual programming course. In *Proceedings - Frontiers in Education Conference, FIE*, volume 3.

Scarton, C. and Aluisio, S. (2009). Análise da inteligibilidade de textos via ferramentas de processamento de língua natural: adaptando as métricas do coh-metrix para o português. *Linguamática*, 2.

Scarton, C. and Aluisio, S. (2010). Coh-metrix-port: a readability assessment tool for texts in Brazilian Portuguese. *9th International Conference on Computational Processing of the Portuguese Language*, 2.

Sheard, J., Simon, B., Carbone, A., Chinn, D., Laakso, M.-J., Clear, T., Raadt, M., D'Souza, D., Harland, J., Lister, R., Philpott, A., and Warburton, G. (2011). Exploring programming assessment instruments: A classification scheme for examination questions. In *Proceedings of the 7th International Workshop on Computing Education Research, ICER*, Providence, USA.

Whalley, J. and Kasto, N. (2014). How difficult are novice code writing tasks?: A software metrics approach. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*, pages 105–112. Australian Computer Society, Inc.