Modelo de Recomendação de Dicas para Resolução de Problemas de Programação, Um Estudo Preliminar

Fabrício Gustavo de Paiva Vicente (IFPB, Campus João Pessoa), Thiago Gouveia da Silva (IFPB, Campus João Pessoa)

E-mails: paiva.fabricio@academico.ifppb.edu.br, thiago.gouveia@ifpb.edu.br.

Área de conhecimento:(Tabela CNPq): 1.03.03.04-9 Sistemas de Informação.

Palavras-Chave: knn; sistema de recomendação; aprendizagem de máquina; ensino de programação.

1 Introdução

O estudo de programação é essencial para cursos de nível superior em Informática, tais como Sistemas para Internet, Redes de Computadores, Engenharia de Software, Computação e afins, como também para diversos outros cursos das áreas de Exatas, Humanas e Ciências Biológicas [Bosse e Gerosa 2015]. Não obstante a grande importância, as disciplinas iniciais de programação comumente apresentam altos índices de reprovação e desistência [Chaves et al. 2013]. Ter conhecimento de programação é uma tarefa difícil, principalmente para alunos em etapas iniciais, ainda se adaptando à universidade. Se faz necessário ter uma base em matemática e lógica e dedicação para resolver diversos problemas e exercícios.

Há várias causas identificadas para o grande número de reprovações em disciplinas de programação, dentre estas destacam-se: (i) os alunos apresentam dificuldades em aprender conceitos de programação; (ii) os alunos apresentam dificuldades em aplicar conceitos de programação; (iii) os alunos apresentam dificuldades em compreender códigos de programas; (iv) os alunos apresentam dificuldade em dividir problemas em módulos; (v) os alunos apresentam falta de interesse ou desânimo para resolver problemas de programação; (vi) os professores apresentam dificuldades em ensinar os conceitos de programação, desenvolver materiais e exercícios de apoio, bem como avaliar os trabalhos de programação [Souza, Batista e Barbosa 2016].

Dado as possibilidades de soluções para o problema do alto índice de reprovação e desistência, está a constante resolução de exercícios diários, sendo que estes são pequenos e ensinam algo pontual [Gomes, Henriques e Mendes 2008]. Para se alcançar a eficácia na aprendizagem, faz-se necessário que os alunos estejam motivados a aprender. Esta motivação dos alunos pode vir através de ferramentas de gamificação, dada a inclinação dos estudantes pelos jogos [Campos, Gardiman e Madeira 2015]. Contudo, muitas vezes, pequenas dificuldades para conseguir resolver um problema podem fazer com que o estudante fique desestimulado para continuar a jornada de estudos. Desta forma, estimular o aluno com questões pequenas e pontuais e apoiá-lo em suas dúvidas agrega valor no aprendizado de programação.

Este trabalho busca propor um modelo de recomendação de dicas para resolução de problemas que seja eficaz e vá direto ao ponto para que o estudante consiga destravar a resolução de um problema. Para construir este modelo, utilizaremos o conhecido algoritmo KNN (K-Nearest Neighbors) [Guo et al. 2003] para classificação dos problemas e recomendação da dica. Baseado no código do aluno que não acertou a questão e em informações específicas do problema, o modelo sugere dicas onde o aluno poderá ter *insights* para conseguir resolver os exercícios propostos.

Serão discutidos o algoritmo KNN e o procedimento de geração de dados sintéticos, conceitos foram utilizados para a criação do modelo. O objetivo do KNN é determinar a qual grupo uma determinada amostra vai pertencer com base nas amostras vizinhas. O princípio por trás do método dos *k* vizinhos mais próximos é encontrar um número de amostras (*k*) de treinamento mais próximas (dado algum critério pré-definido de distância) do novo ponto e prever o rótulo a partir deles. O número de amostras pode ser uma constante definida pelo usuário ou pode variar com base na densidade local de pontos (aprendizado de vizinho baseado em raio). A distância pode, em geral, ser qualquer medida métrica, sendo a distância euclidiana padrão a escolha mais comum[1].

Os dados sintéticos podem ser definidos como dados que não foram obtidos do mundo real. Desta forma, são dados gerados por um sistema com o objetivo de imitar dados reais em termos de características essenciais. Os

[1] Nearest Neighbors. Scitkit Learn (2020), Disponível em: <https://scikit-learn.org/stable/modules/neighbors.html> Acesso em 11 de agosto de 2021.

dados sintéticos oferecem vários benefícios: privacidade, pois todas as informações pessoais foram removidas e os dados não podem ser rastreados até serem menos caros e mais rápidos de coletar quando comparados à coleta de dados do mundo real. Além disso, é possível antecipar e simular dados para tentar prever situações reais futuras. O restante deste trabalho está organizado como segue: a Seção 2 discute o processo de validação do modelo, a Seção 3 discute os resultados obtidos e o nível de acerto obtido, a Seção 4 debate sobre as considerações finais e as contribuições deste artigo.

2 Materiais e Métodos

Entre as dificuldades encontradas no processo de ensino-aprendizagem de disciplinas de programação, um dos principais pontos é a ausência de *feedback* para os alunos [Moreno e Pineda, 2018]. Estudantes, ao tentar resolver questões, muitas vezes ficam a poucos passos de conseguir concluir. Entretanto, esses impedimentos podem gerar desmotivação e, por consequência, desistência de concluir a resolução. O modelo de recomendação proposto vai analisar características do código submetido que não conseguiu resolver a questão e outras características do problema. Posteriormente, a partir destas informações, são definidas características (*features*) que são analisadas via KNN, através da biblioteca de aprendizagem de máquina Scikit-learn.

Para validar o modelo, foram selecionados 10 problemas da plataforma URI Online Judge [URI, 2021]. Como exemplo de questão, tomamos o problema 1006. Este problema requer a leitura de três valores (A, B e C), que representam as três notas de um aluno fictício, e o cálculo da média, considerando que a nota A tem peso 2, a nota B tem peso 3 e a nota C tem peso 5. Cada nota é um valor de ponto flutuante com um dígito após a casa decimal. A saída é `"MEDIA = VALOR"`, onde `VALOR` representa o cálculo da média, como indicado. Assim sendo, foram geradas três versões de código. Uma correta (Figura 1) e duas versões erradas (dados sintéticos) (Figura 2 (a) e (b)). O problema é como escolher a dica apropriada para aumentar o nível de aprendizado e acerto nas questões.

```
a = float(input())
b = float(input())
c = float(input())

media = ((a * 2) + (b * 3) + (c * 5)) / 10

print("MEDIA = %0.1f" %media)
```

Figura 1: Código correto do Problema 1006 do URI

a)
```
a = float(input())
b = float(input())
c = float(input())

media = ((a x 2) + (b x 3) + (c x 5)) / 10

print("MEDIA = %0.1f" %media)
```

b)
```
a = float(input())
b = float(input())
c = float(input())

media = a * 2 + b * 3 + c * 5 / 10

print("MEDIA = %0.1f" %media)
```

Figura 2: Código com erro para o Problema 1006 do URI; (a) Com erro de sintaxe; (b) Com erro de lógica

3 Resultados e Discussão

O primeiro passo para implementar o sistema de recomendações proposto foi a definição do conjunto de características extraídas do programa escrito pelo estudante, uma fase conhecida como engenharia de características. Foram selecionadas características binárias (valor *1* para presente, *0* para ausente) do código-fonte (A), dos arquivos de entrada (B) e dos arquivos de saída corretos (C) para o problema, como apresentado na Tabela 1. Foram escolhidos 10 assuntos abordados em disciplinas iniciais de introdução à programação e foram gerados (como dados sintéticos) 10 exemplos de códigos corretos e 20 exemplos de códigos errados. Para cada exemplo de código errado, foi escrita uma dica, conforme exemplificado na Figura 3. Por fim, foi escrito o código para extração das características e os

dados sintéticos foram usados para o treinamento do algoritmo KNN. Os resultados preliminares foram satisfatórios. Vale notar que o conjunto de dados é muito pequeno e que serão gerados novos dados na continuidade da pesquisa.

Tabela 1: Características extraídas da resposta enviada pelo estudante

(B) Múltiplas linhas	(B) *Split* Necessário	(B) Valores inteiros	(B) Valores decimais	(B) Valores textuais	(C) Múltiplas linhas
(C) Valores inteiros	(C) Valores decimais	(C) Possui espaço	(A) Possui int()	(A) Possui float()	(A) Espaço extra
(A) Op. Aritméticos	(A) Op. Lógicos	(A) Formatação (%)	(A) Faltando ':'	(A) For/While	(A) Range() / []

Quando mais de um operador aparece em uma expressão, a ordem em que são realizadas as operações dependem das regras de **precedência** (*rules of precedence*). Python segue as regras de precedência dos seus operadores matemáticos da mesma forma que a matemática. Parênteses têm a mais alta precedência e podem ser usados para forçar que uma expressão seja calculada na ordem que você deseja. Como expressões entre parênteses são calculadas primeiro, **2*(3-1)** é **4**, e **(1+1)**(5-2)** é **8**. Você pode usar parênteses para tornar uma expressão mais legível, como em **(minutos * 100) / 60**, mesmo que isto não mude o resultado. Exponenciação tem a segunda precedência mais alta, assim **2**1+1** é **3** e não **4**, e **3*1**3** é **3** e não **27**. Você pode explicar o porquê? Multiplicação e ambas as divisões têm a mesma precedência, que são mais altas que adição e subtração, que também têm a mesma precedência. Logo, **2*3-1** é **5** e não **4**, e **5-2*2** é **1** e não **6**. Operadores com a mesma precedência são executados da esquerda para a direita. Em álgebra dizemos que eles são **associativos à esquerda** (*left-associative*). Desta forma, na expressão **6-3+2** a subtração é realizada primeiro e tem como resultado **3**. Depois adicionamos **2** e obtemos o resultado **5**. Se os operadores tivessem sido executados da direita para a esquerda o resultado seria **6-(3+2)** que é **1**.

Figura 3: Dica para o erro de lógica relatado na Figura 2(b)

4. Considerações Finais

Este trabalho apresenta a proposta de um modelo de recomendação de dicas para resolução de problemas de programação utilizando a técnica KNN e dados sintéticos para sua validação com objetivo de oferecer um *feedback* de qualidade para minimizar os efeitos dos pequenos erros no aprendizado de programação. Dentre as contribuições deste artigo, destacamos o uso do KNN com técnica de Aprendizagem de Máquina, o uso de dados sintéticos para simular dados reais, e o uso de dicas e sugestões no apoio de resolução de questões com tentativas de respostas erradas. Contudo, é importante salientar que este modelo está em fase de testes, estando suscetível a problemas e erros, que serão corrigidos tão logo detectados. Este modelo ainda não foi aplicado em grande escala, podendo ter ajustes em seu algoritmo.

Referências

Bosse, Y.; Gerosa, M. A. (2015). **Reprovações e Trancamentos nas Disciplinas de Introdução à Programação da Universidade de São Paulo: Um Estudo Preliminar.** In XXIII Workshop sobre Educação em Computação. Recife.

Campos A.; Gardiman R.; Madeira C. (2015). **Aplicação da Gamificação na Disciplina de Empreendedorismo.** In XXIII Workshop sobre Educação em Computação. Recife.

Chaves, J. O. M.; Castro, A. F.; Lima, R. W.; Lima, M. V. A.; Ferreira, K. H. (2013). **Integrando Moodle e Juízes Online no Apoio a Atividades de Programação.** In Anais do Simpósio Brasileiro de Informática na Educação (Vol. 24, No. 1, p. 244).

Guo, Gongde, et al. **KNN model-based approach in classification.** OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, Berlin, Heidelberg, 2003.

SOUZA, D. M.; BATISTA, M. H. S.; BARBOSA, E. F. **Problemas e dificuldades no ensino de programação: Um mapeamento sistemático.** Revista Brasileira de Informática na Educação, v. 24, n. 1, p. 39, 2016.

URI. **URI Online Judge.** Disponível em: https://www.urionlinejudge.com.br/judge/pt. Acesso em: 12 ago. 2021.