Uso de Mineração de Dados para o Auxílio de Produção de Material Didático em Disciplinas de Algoritmos

Rafael A. S. Andrade[1,2], **Franciny M. Barreto**[1], **Esdras L. Bispo Jr.**[1,2]

[1] Instituto de Ciências Exatas e Tecnológicas (ICET)
Universidade Federal de Jataí (UFJ)
Jataí – GO – Brasil

[2]Jataí ACM SIGCSE *Chapter*
Grupo de Interesse Especial em Educação de Computação

rafasouza@protonmail.com, {franciny, bispojr}@ufg.br

Abstract. *Algorithms can be defined as a sequence of executable actions to obtain a solution for many kind of problems. The discipline of algorithms, which is the first contact the students with Computer Programming, and beyond all is the basis of any course of Computing. However, for many students, there is a difficulty in the course, in which the problem is often due to technical impasses, which hinder the development of the algorithm, many technologies are used for the elaboration of algorithm and many of them have peculiarities that make the student have obstacles. Faced with these difficulties, there may be cases in which the teacher can not identify the frequency of these errors, which are motivated by technical order. This work aims to proposes the creation of programmatic technical content, and it is suggested from the data mining process of the StackOverflow data base that contains specific aspects of the discipline of programming. Some preliminary results of this model have already been obtained with the accuracy level of data greater than 90%.*

Resumo. *Algoritmo pode ser definido como uma sequência de ações executáveis para a obtenção de uma determinada solução buscando a obtenção da solução de um problema. A disciplina de Algoritmos, é o primeiro contato dos alunos com a área de Programação de Computadores e, além disso, a base para a Computação. Pórem, para muitos alunos a implementação de algoritmos é um desafio devido a dificuldade de ordem técnica, detalhes de linguagem de programação ou ainda configuração dos ambientes de desenvolvimendo. Diante dessas dificuldades, podem ocorrer casos nos quais o professor não consegue identificar a frequência dessas dificuldades, levando os alunos a uma maior frustração na aprendizagem dos conceitos subjacentes da disciplina. Este trabalho propõe a criação de um sistema de recomendação de conteúdo técnico programático. Este sistema tem como parte essencial um modelo gerado a partir da mineração da base de dados do StackOverflow. Alguns resultados preliminares deste modelo já foram obtidos com o nível de acurácia de treinamento maior que 90%.*

1. Introdução

A quantidade de dados produzidos na Internet vem crescendo gradualmente todos os dias [Wu et al. 2014]. Perante este fato, o tratamento dos dados pode auxiliar no processo

de tomada de decisão. No cenário educacional, este mecanismo de utilização dos dados vem sendo muito utilizado. As contribuições vão desde o auxílio de estudantes com necessidades de aprendizagem até à prevenção de certas situações, como a evasão escolar [Balaniuk et al. 2011].

Diante deste cenário, mais especificamente na Educação de Computação, há questões em relação ao ensino e a aprendizagem em disciplinas introdutórias de programação que precisam de serem atendidas [Francisco et al. 2017]. Diversos alunos deparam-se com diversas dificuldades nestas disciplinas. Dentre estas dificuldades, podemos citar o pouco desenvolvimento de lógica matemática, dificuldade de concentração e o ritmo de aprendizagem específico de cada aluno. Estas dificuldades costumam comprometer a compreensão de conteúdos futuros, levando os estudantes a um sentimento de frustração durante o desenvolver da matéria de algoritmos de programação.

Uma das formas de abordar esses problemas de aprendizagem em programação é através da criação de uma material didático personalizado para um dado contexto, conforme apresentado em [Berssanette et al. 2016]. Em [Rocha et al. 2010] é possível ver a personalização de material didático em programação. Os autores implementaram um método de ensino baseado no Sistema Personalizado de Ensino, e de uma ferramenta integrada ao AVA Moodle. Em [Cardoso et al. 2009], também é possível encontrar o desenvolvimento de material didático para o auxílio no processo de ensino e aprendizagem dentro de Ciência e Biologia. Entretanto, há uma carência de trabalhos nesta direção utilizando o conhecimento implícito em redes sociais, por exemplo.

O presente trabalho tem como propósito a criação de um sistema de recomendação de conteúdo técnico programáticoa partir do uso de uma base de dados de perguntas e respostas. Uma das etapas deste método consiste na criação de um sistema de recomendação de conteúdo técnico programático a partir das ocorrências recorrentes de perguntas presentes na base de dados. Este sistema tem como parte essencial um modelo gerado a partir da mineração da base de dados do *StackOverflow*. Alguns resultados preliminares deste modelo já foram obtidos atráves de testes realizados partir da base de dados, obtendo um nível de acurácia de treinamento maior que 90%. O conteúdo técnico programático recomendado será utilizado como referência para a redação do plano de ensino do professor para uma dada disciplina da Computação que exija a aprendizagem de competências que envolvem a implementação concreta de algoritmos.

O restante do trabalho é dividido como se segue. Na Seção 2, são apresentados os trabalhos relacionados. Na Seção 3, são pontuados conceitos básicos e fundamentação teórica. Na Seção 4, é apresentada a proposta do trabalho. Na Seção 5, é descrita a metodologia aplicada. Na Seção 6, são apresentados os resultados preliminares obtidos. E, por fim, na Seção 7, são delineadas as considerações finais e perspectivas dos próximos passos do projeto.

2. Trabalhos Relacionados

Em [Bryce 2011], foi observado que erros em programas era problemas comuns encontrados em qualquer programa desenvolvidos em Ciências da Computação. Portando, de uma maneira criativa e divertida foi desenvolvido uma competição que busca dividir em times que tem como intuito identificar erros nos códigos. E com isso, muitos alunos tinham maior interesse em testes e, também evitou que muitos deles evitassem esses erros

no futuro. Em [Gaber and Kirsh 2018], foi feito o estudo no qual estudantes reportavam sobre os defeitos encontrados em seus códigos, sendo eles divido em três categorias: erros no programa, recurso ausente ou código mal escrito. E com isso foi identificado que a maioria era em erros de programa e recurso ausente. E muitos estudantes afirmaram que devido a eles reportarem conseguiram melhorar e evitar erros em seus códigos.

Em [Delavari et al. 2008], são abordadas as capacidades da MD nas instituições de ensino superior, de modo a propor uma nova orientação envolvendo uma aplicação de Mineração de Dados na Educação (MDE). Esta aplicação trata de como a MD pode auxiliar no processo de tomada de decisão em universidades. Em [Tang et al. 2000], é exposto a motivação da aprendizagem dos alunos, utilizando uma árvore de decisão como tutor, sendo eficiente para o ensino à distância.

Dentre todos estes artigos mostrado, todos diferem em pelo menos um aspecto da proposta deste trabalho. Um destes aspectos, por exemplo, é que nenhum deles admitem a necessidade da criação de um sistema de recomendação de conteúdo técnico programático com mediação do professor no ensino de algoritmos de programação.

3. Fundamentação Teórica

O objetivo desta Seção é apresentar conceitos importantes para a compreensão da proposta deste trabalho. Apresenta-se a seguir uma breve descrição da Mineração de Dados (Seção 3.1), da Mineração de Dados na Educação (Seção 3.2), e, também, sobre a biblioteca *TensorFlow* que foi utilizada no desenvolvimento do projeto (Seção 3.3), e por fim, Categorização de Bugs (Seção 3.4).

3.1. Mineração de Dados

A partir de uma grande variedade de contextos, a produção e a coleta de dados vêm ocupando proporções cada vez maiores. Como consequência desta realidade, surge a necessidade da proposição de novas teorias computacionais e da criação de ferramentas para auxiliar os usuários na extração de informações úteis, a partir de um crescente volume de dados digitais [Fayyad et al. 1996]. Um conjunto de técnicas é proposto com o objetivo de atender a este fim, permitindo o início ao surgimento da Mineração de Dados.

A MD é uma importante área da Inteligência Artificial (IA) que se preocupa em descobrir informações decisivas em determinados contextos. A partir dos dados coletados, busca-se extrair informações implícitas de forma que, subsequentemente, estas possam ser categorizadas e auxiliem no processo de tomada de decisões. Na MD, as soluções podem ser realizadas por meio de algoritmos independentes, ou por meio da união (ou mesclagem) de algoritmos para a obteção de resultado específico. Um exemplo é a descoberta de conhecimento (*Knowledge Discovery* - KD), sendo um processo de extração de conhecimento desconhecido em uma grande base de dados [Wu 2007].

A KD resulta em um processo de extração não trivial de informação implícita vinda de uma grande base de dados. Esta informação pode ser desconhecida ou potencialmente útil. A KD identifica padrões normalmente relacionados aos usuários da base em questão [Usai et al. 2018].

3.2. Mineração de Dados Educacionais

A partir do aspecto acadêmico, há a necessidade de garantir que a maior quantidade de informações relevantes estejam disponíveis para a tomada de decisão. Gestores, educado-

res e educandos podem utilizar informações implícitas de bases de dados educacionais para o aperfeiçoamento de suas práticas acadêmicas. Com o propósito de responder a demandas deste contexto, e da Educação de uma forma geral, várias abordagens vêm sendo propostas para esta necessidade. Um conjunto destas abordagens estão vinculadas à Mineração de Dados na Educação.

O objetivo da MD costuma ser distinto em cada contexto de aplicação. Isto também ocorre na MDE de forma que existem dois cenários bastante comuns de aplicação, como a melhoria do processo de ensino e o fornecimento de retorno adequado para a própria evolução do aprendizado dos alunos. Estes objetivos em sua maioria são difíceis de serem quantificados e desta forma são necessárias técnicas especiais para tal medição [Romero and Ventura 2010].

Um dos espaços bastante promissores da MDE é a exploração de tipos únicos de dados que surgem dentro de instituições de Educação. A partir destes, métodos são utilizados para entender os estudantes e o seu ambiente de aprendizado [Fatima et al. 2015]. Existem métodos utilizados em MDE que são aprimoramentos de técnicas de MD. Algumas destas técnicas são: (i) o agrupamento (*clustering*), (ii) a predição (*prediction*), (iii) a mineração de relacionamento, (iv) descoberta com modelo e (v) "destilação"de dados [Tan et al. 2005].

A partir destas técnicas, algumas pesquisas interessantes foram realizadas. Em [Nandeshwar et al. 2011], foi utilizada a MD para encontrar padrões de evasão na universidade, de forma a descobrir se os estudantes continuariam pelos primeiros três anos de graduação. E em [Winne and Baker 2013], foram apresentadas as contribuições que a MDE proporciona para pesquisas na área de metacognição, motivação e aprendizado.

3.3. *TensorFlow*

O *TensorFlow*[1] é um sistema focado em aprendizado de máquina, possibilitando operações em grande escala. Esta ferramenta pode ser utilizada para diversas tarefas, sendo uma delas a criação e treinamento de redes neurais. Com o *TensorFlow*, é possível realizar tarefas de MD, como buscar, detectar e decifrar padrões e correlações.

O *TensorFlow* é uma ferramenta amplamente utilizada pela comunidade de pesquisadores que atuam na área de aprendizado de máquina. Em [Abadi et al. 2016], por exemplo, os autores apresentam o poder de extensibilidade da ferramenta. Alguns casos de usos são descritos, utilizando o *TensorFlow* em problemas de diferenciação, otimização e tolerância a falhas. Devido à possibilidade da execução do TensorFlow em ambientes distribuídos heterogêneos, já existem alguns projetos focados em classificação de imagens e detecção de objetos [Frome et al. 2013], atividades estas que exigem mais dos recursos computacionais. É possível executar rotinas com o *TensorFlow* até dentro de um navegador ou em um terminal sem muitos recursos computacionais.

O *TensorFlow* possibilitou a aprendizagem de máquina tornar-se uma atividade em nível declarativo. Cabe aos usuários apresentar os dados, descrever a representação de seu modelo, e deixar que a própria ferramenta realize todas as etapas posteriores de processamento. Isto garante, além de uma programação ágil, uma possibilidade (antes improvável) de profissionais não-especializados na área poderem criar os seus próprios

[1] <www.tensorflow.org>

modelos de aprendizagem de máquina [Fiebrink 2019].

3.4. Categorização de Bugs

Muitos estudantes acreditam que linguagem de programação é uma tarefa complicada [Spohrer et al. 1985]. Mas em sua maioria, linguagem de programação pode se tornar algo complexo a devido erros que surgem no decorrer da implementação. E, diante desses erros foi analisado que em sua maioria nem todos eles eram randômicos, eles poderiam ser caracterizados em certas categorias, podendo perceber a ocorrência deles em grande parte das implementações de códigos feitos pelos alunos.

Tendo em vista a dúvida diante dos erros que acontecem em muitos códigos, uma parte significante de pesquisadores focaram em identificar os erros específicos e por fim categorizá-los [McCauley et al. 2008]. Pois, devido a existência de uma lista de erros comuns acaba facilitando como forma de identificação. Existe estudos que buscam utilizar dessas categorias como forma de ferramenta para auxiliar novos estudantes [Hristova et al. 2003]. E outros estudos buscam utilizar da categorização para entender os problemas que os estudantes sofrem [Ahmadzadeh et al. 2005, Robins et al. 2006].

4. Proposta de Trabalho e Metodologia

Este trabalho tem como propósito a criação de um método para a produção de material didático para disciplinas de programação a partir do uso de uma base de dados de perguntas e respostas. Uma das etapas deste método consiste na criação de um sistema de recomendação de conteúdo técnico programático a partir das ocorrências recorrentes de perguntas presentes na base de dados. Este sistema tem como parte essencial um modelo gerado a partir da mineração da base de dados do StackOverflow.

O processo subjacente ao sistema é esquematizado na Figura 1. Primeiramente, recebe-se como entrada a ementa da disciplina de programação. Após processada, será utilizado um modelo de aprendizagem de máquina criado a partir do TensorFlow. Este modelo, o *BugSniffer*, será composto por redes neurais artificiais em que, além da ementa, receberá como entrada um fragmento da base de dados do *StackOverflow*[2]. O *BugSniffer* retornará uma lista das falhas mais frequentes ocorridos nos últimos anos a partir dos tópicos presentes na ementa. Esta lista irá ser apresentada para o docente como uma recomendação para que ele possa incluir, em seu plano de ensino, um conteúdo programático de ordem técnica, com o propósito de evitar que os alunos se envolvam com menor frequência em dificuldades não-conceituais durante a disciplina.

Com o intuito de realizar a validação da proposta apresentada, será utilizado o paradigma GQM. Descreveremos em mais detalhes este paradigma (Seção 4.1) e a sua utilização no trabalho, através de suas especificações (Seção 4.2).

4.1. Paradigma GQM

O paradigma *Goal-Question-Metric* (GQM) [Van Solingen et al. 2002] é utilizado na Engenharia de Software para auxiliar na medição de processos e softwares, sendo uma abordagem orientada a métricas. A vantagem da sua utilização é que ele separa as preocupações organizacionais (objetivos) das específicas do processo (questões). Isto fornece

[2]<www.stackoverflow.com>

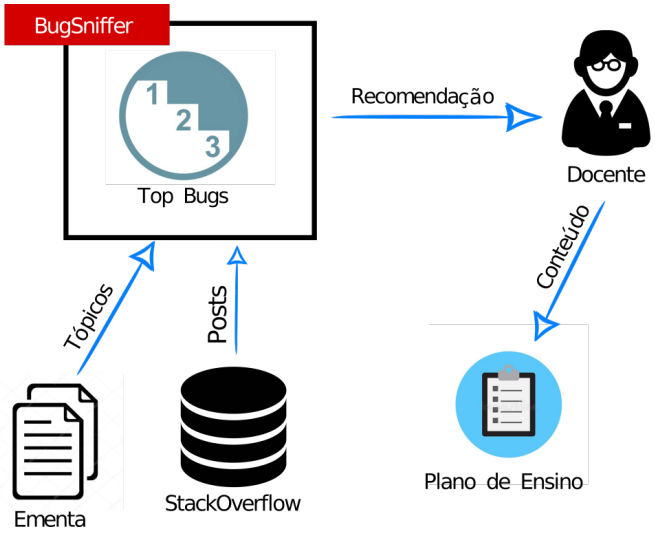

Figura 1. Esquema ilustrando o processo de utilização do *BugSniffer* pelo docente.

uma base para decidir quais dados deveriam ser coletados e como eles deveriam ser analisados de forma a responder ao conjunto de questões que de fato pretende-se resolver [Sommerville 2010].

Os conceitos utilizados dentro do paradigma GQM podem ser definidos sucintamente como: (i) objetivo - meta a qual a pesquisa almeja alcançar; (ii) questões - refinamentos dos objetivos em que áreas específicas de incertezas relacionadas são identificadas; (iii) métricas - medições que precisam ser feitas para ajudar a responder às questões e confirmar, quantitativamente, se a pesquisa alcançou ou não o objetivo desejado.

Embora o GQM seja largamente utilizado na Engenharia de Software, este paradigma vem sendo utilizado e adaptado em pesquisas na Educação [Gladcheff et al. 2002, Machado 2016].

4.2. Especificações GQM

As especificações concretas para os três conceitos principais do GQM foram estabelecidas. O objetivo do projeto consiste em avaliar o uso do sistema de recomendação *BugSniffer* no auxílio ao docente na construção do conteúdo técnico programático de disciplinas que envolvam implementação de algoritmos. A ferramenta pode ser devidamente adaptada para qualquer base de dados de perguntas e respostas. Entretanto, neste projeto, foi escolhida a base do *StackOverflow*.

Três questões foram elencadas como refinamentos do objetivo proposto. A primeira é (i) "o *BugSniffer* contribui positivamente no processo de tomada de decisão dos professores em relação à escolha e construção de conteúdos técnico programático?"; a segunda é (ii) "o *BugSniffer* contribuiu para o êxito dos alunos na disciplina em questão?" e a terceira e (iii) "o *BugSniffer* é um sistema fácil de ser utilizado?".

Por fim, métricas são construídas para as três questões elencadas. Para as especificações das métricas, serão criados três questionários. Estes questionários serão submetidos aos participantes dos experimentos. As respostas concedidas aos questionários servirão de entrada para o cálculo das métricas.

O Questionário 1 servirá para identificar aspectos sobre a utilidade do *BugSniffer* para a tomada de decisão dos professores.

O Questionário 2 servirá para identificar aspectos sobre o sucesso dos alunos expostos ao método de ensino afetado pelo uso do *BugSniffer*. Variáveis como número de aprovados (em comparação ao semestre anterior) e percepção sobre o método são exemplos dos dados requisitados. O propósito deste questionário é de correlacionar, se possível, de maneira exploratória, os resultados que serão obtidos da aplicação do Questionário 1 com os resultados deste questionário.

Por fim, o Questionário 3 servirá para identificar aspectos sobre a usabilidade do *BugSniffer*. Questões sobre facilidade na visualização dos dados, eficiência de uso, frequência de ocorrência e seriedade dos erros são exemplos dos dados requisitados.

Todos os questionários serão compostos por questões objetivas e subjetivas. As questões objetivas utilizarão a escala do tipo *Likert* [Nemoto and Beglar 2014] de todos os questionários. As pontuações para cada questão será associada de um a cinco pontos, respectivamente com rótulos: (i) Discordo Totalmente, (ii) Discordo Parcialmente; (iii) Indiferente, (iv) Concordo parcialmente, e (v) Concordo plenamente.

A partir das respostas das questões objetivas obtidas dos questionários, duas métricas foram propostas. A métrica M_1 é um indicador da utilidade do *BugSniffer* para a tomada de decisão dos professores. M_1 é definida a partir das respostas coletadas a partir do Questionário 1. M_1 é definida como $M_1 = \frac{PONT^{Q1}}{n}$, sendo $PONT^{Q1}$ a pontuação obtida no Questionário 1 com todos os participantes. A seguinte interpretação de M_1 será adotada: (i) se $M_1 \geq 3,0$, então *BugSniffer* contribuiu positivamente para a tomada de decisão dos professores; (ii) caso contrário, a ferramenta tem a sua utilidade comprometida.

A métrica M_2 é um indicador para a usabilidade do *BugSniffer*. M_2 é definida a partir das respostas coletadas a partir do Questionário 3 com todos os participantes. M_2 é definida como $M_2 = \frac{PONT^{Q3}}{n}$. A seguinte interpretação de M_2 será adotada: (i) se $M_2 \geq 3,0$, então o *BugSniffer* é fácil de usar; (ii) caso contrário, a ferramenta tem a sua usabilidade comprometida.

Os resultados do Questionário 2 (incluindo todas as questões subjetivas de outros questionários) serão utilizados primordialmente para a análise qualitativa dos dados que serão obtidos após a realização do experimento.

5. Resultados Preliminares

O desenvolvimento do sistema de recomendação *BugSniffer* para o processo de Mineração dos Dados de postagens da base de dados do *StackOverflow* foi parcialmente concluído. Utilizando o *TensorFlow*, este módulo foi dividido em três fases, sendo elas: (i) obtenção dos dados, (ii) mineração dos dados e (iii) avaliação do resultado.

Na primeira fase, o objetivo foi a obtenção dos dados da base do *StackOverflow*, e para isso foi acessado o *StackExchange*[3]. O *StackExchange* provê acesso a diversos sites que possuem base de dados aberta. Para a obtenção dos dados para este módulo, foram escolhidas quatro grande categorias de *bugs* a partir da realidade concreta no ensino

[3]<<https://data.stackexchange.com/>>

de programação documentada em [Bryce et al. 2010]. Das categorias apresentadas no trabalho, foram selecionadas quatro, a saber: (i) laços, (ii) ponteiros, (iii) vetores, e (iv) funções.

O recorte da base de dados escolhida, dentro do *StackOverflow*, foi a estaduniden-se. A principal justificativa deve-se à concentração maior de dados e a um alcance a diversos países e usuários de língua inglesa. Por este motivo, na construção da consulta à base, os rótulos das categorias foram devidamente traduzidos (*e.g.* *loops*, *pointers*, *array*, *function*). O código do modelo do *BugSniffer*, as consultas ao *StackExchange* e os dados utilizados como entrada para o modelo, suficientes para a reprodução dos resultados preliminares deste trabalho, estão disponíveis em repositório no GitHub[4].

Após obtidos os dados, a segunda fase consiste da mineração dos dados propriamente dita. Como mencionado na Seção 3.3, após informar os devidos parâmetros para o *TensorFlow*, a execução da MD é realizada internamente pela ferramenta. Para estes resultados preliminares, foram utilizados 500 posts, sendo cada um deles um fluxo de pergunta (com suas respostas encadeadas) do *StackOverflow*.

Foi estabelecido que, para a fase de treinamento e validação do modelo, a divisão dos dados foi de 80% e 20%, respectivamente. A quantidade de iterações de treinamento do modelo foi de 600 épocas, com tamanhos de lote de 512. As demais especificações podem ser obtidas em mais detalhes no repositório.

Uma captura da tela de resultado final do treinamento pode ser observada na Figura 2. Percebe-se, na figura, que a acurácia do treinamento atingiu o valor de 0,9875, sendo 1,0 o seu valor máximo.

Figura 2. Captura de tela que apresentada o resultado final do treinamento da rede neural artificial do modelo do *BugSniffer*.

[4]<<https://github.com/rafaelandrade/BugSniffer-WEI2019>>

Apesar de utilizar um número de 600 épocas, é possível observar, no gráfico apresentado na Figura 3, que a acurácia, tanto do treinamento quanto da validação, atingiu o equilíbrio a partir da Época 100, aproximadamente. Em passos futuros do projeto, em que a quantidade de dados para o treinamento será bem maior, será possível admitir a hipótese de refinar os parâmetros do modelo para que o seu tempo de processamento seja não mais do que o necessário. Por exemplo, é bem possível que a quantidade de épocas não necessite ser tão grande quanto a utilizada.

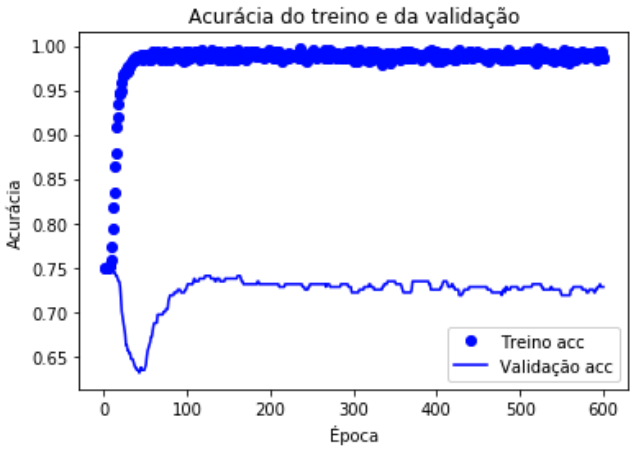

Figura 3. Gráfico apresentando a acurácia do treinamento e validação do modelo do *BugSniffer* ao decorrer das 600 épocas.

E, por fim, a última fase é a de avaliação dos resultados. Na Figura 4, temos uma "matriz de confusão", apresentando em quais categorias o modelo obteve êxito, e quais são os erros comumente feitos. Por exemplo, é possível observar que o modelo obteve um alta precisão em questões categorizadas como *array*, *function* e *pointers* (acurácia próxima a 100%); e ocasionalmente categorias "confusas" como em *loop* (60%). É interessante utilizar a "matriz de confusão" para descrever o modelo proposto, pois é importante saber em quais categorias deve-se ter uma atenção mais específica. A matriz informa quais categorias estão efetivamente não alcançando valores significativos, prejudicando mais efetivamente a acurácia total do modelo.

6. Considerações finais

Este trabalho teve como propósito a criação de um sistema de recomendação de conteúdo técnico programático para disciplinas de programação a partir do uso de uma base de dados de perguntas e respostas. Este sistema, o *BugSniffer*, tem como parte essencial um modelo gerado a partir da mineração da base de dados do *StackOverflow*. Alguns resultados preliminares deste modelo já foram obtidos com o nível de acurácia na fase de treinamento maior que 90%. O conteúdo técnico programático recomendado será utilizado como referência para a redação do plano de ensino do professor para uma dada disciplina da Computação que exija a aprendizagem de competências que envolvem a implementação concreta de algoritmos.

Os resultados preliminares foram importantes para identificar de maneira concreta algumas dificuldades inerentes ao processo de refinamento do modelo do *BugSniffer*. O uso do *TensorFlow* é de bastante importância pois permite uma dedicação maior a este

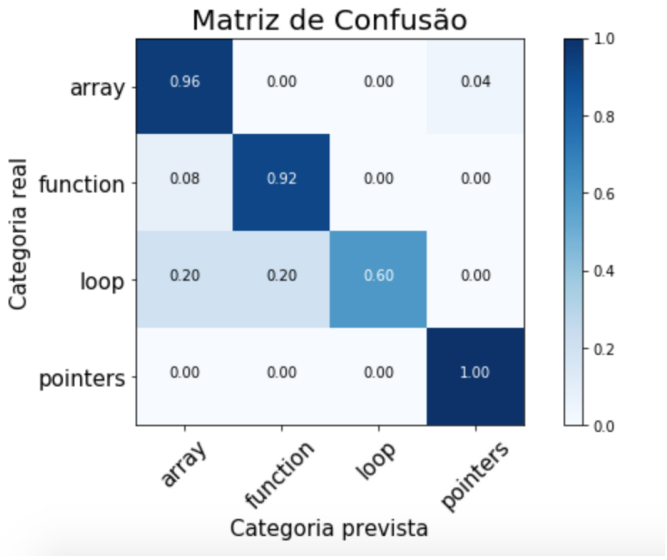

Figura 4. Gráfico apresentando a "matriz de confusão" indicando a acurácia do modelo do *BugSniffer* em cada uma de suas categorias na fase de avaliação.

processo de parametrização que a aprendizagem de máquina exige. O próximo passo natural da evolução do modelo é escolher especificamente uma disciplina para que este refinamento seja realizado, aumentando a quantidade de categorias e a quantidade dos dados de treinamento.

O passo posterior à finalização do modelo é aplicação do *BugSniffer* em um caso concreto. Uma turma do segundo semestre do Bacharelado de Ciência da Computação da Universidade Federal de Goiás (UFG), Regional Jataí, será utilizada para que um estudo de caso descritivo seja realizado. Com este último passo, deseja-se verificar se o *BugSniffer* contribui efetivamente com a melhoria da qualidade do processo de ensino e aprendizagem de disciplinas que envolve a implementação de algoritmos.

Referências

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A System for Large-Scale Machine Learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

Ahmadzadeh, M., Elliman, D., and Higgins, C. (2005). An Analysis of Patterns of Debugging among novice Computer Science Students. In *ACM SIGCSE Bulletin*, volume 37, pages 84–88. ACM.

Balaniuk, R., do Prado, H. A., da Veiga Guadagnin, R., Ferneda, E., and Cobbe, P. R. (2011). Predicting Evasion Candidates in Higher Education Institutions. In *International Conference on Model and Data Engineering*, pages 143–151. Springer.

Berssanette, J. H. et al. (2016). Ensino de Programação de Computadores: uma Proposta de Abordagem Prática Baseada em ausubel. Master's thesis, Universidade Tecnológica Federal do Paraná.

Bryce, R. (2011). Bug wars: a Competitive Exercise to Find Bugs in Code. *Journal of Computing Sciences in Colleges*, 27(2):43–50.

Bryce, R. C., Cooley, A., Hansen, A., and Hayrapetyan, N. (2010). A One Year Empirical Study of Student Programming Bugs. In *2010 IEEE Frontiers in Education Conference (FIE)*, pages F1G–1. IEEE.

Cardoso, D. C., Cristiano, M. P., and Arent, C. O. (2009). Development of New Didactic Materials for Teaching Science and Biology: the Importance of the New Education Practices. *OnLine Journal of Biological Sciences*, 9(1):1–5.

Delavari, N., Phon-Amnuaisuk, S., and Beikzadeh, M. R. (2008). Data Mining Application in Higher Learning Institutions. *Informatics in Education*, 7(1):31–54.

Fatima, D., Fatima, S., and Prasad, D. A. K. (2015). A Survey on Research Work in Educational Data Mining. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 17.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI magazine*, 17(3):37.

Fiebrink, R. (2019). Machine Learning Education for Artists, Musicians, and Other Creative Practitioners. *ACM Transactions on Computing Education*.

Francisco, R. E., Ambrósio, A. P., et al. (2017). Grau de Dificuldade de Problemas de Programação Introdutória: Uma Revisão Sistemática da Literatura. In *25º Workshop sobre Educação em Computação (WEI 2017)*, volume 25. SBC.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T., et al. (2013). Devise: A Deep Visual-Semantic Embedding Model. In *Advances in neural information processing systems*, pages 2121–2129.

Gaber, I. and Kirsh, A. (2018). The Effect of Reporting Known Issues on Students' Work. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 74–79. ACM.

Gladcheff, A. P., Sanches, R., and da Silva, D. M. (2002). Um Instrumento de Avaliação de Qualidade de Software Educacional: como elaborá-lo. *Pensamento & Realidade. Revista do Programa de Estudos Pós-Graduados em Administração-FEA. ISSN 2237-4418*, 11.

Guedes, L. R. and Paterno, A. S. (2018). BrC: Proposta de uma Biblioteca em Português para Ensino de Programação em Linguagem C. In *26º Workshop sobre Educação em Computação (WEI 2018)*, volume 26, Porto Alegre, RS, Brasil. SBC.

Hristova, M., Misra, A., Rutter, M., and Mercuri, R. (2003). Identifying and Correcting java Programming Errors for Introductory Computer Science Students. *ACM SIGCSE Bulletin*, 35(1):153–156.

Machado, A. S. (2016). Uso de Softwares Educacionais, Objetos de Aprendizagem e Simulações no Ensino de Química. *Revista Química Nova na Escola*, 38(2):104–111.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., and Zander, C. (2008). Debugging: a Review of the Literature from an Educational Perspective. *Computer Science Education*, 18(2):67–92.

Melo, L. B., Costa, P. R. S., de Paiva Onofre Filho, M., Lordão, F. A. F., and de Almeida, L. C. (2018). Uma Metodologia para Implementação da Disciplina Informática Básica em Cursos Técnicos Integrados ao Ensino Médio. In *26º Workshop sobre Educação em Computação (WEI 2018)*, volume 26, Porto Alegre, RS, Brasil. SBC.

Nandeshwar, A., Menzies, T., and Nelson, A. (2011). Learning Patterns of University Student Retention. *Expert Systems with Applications*, 38(12):14984–14996.

Nemoto, T. and Beglar, D. (2014). Likert-scale Questionnaires. In *Japan Association for Language Teaching (JALT) 2013 Conference Proceedings*, pages 1 – 8.

Robins, A., Haden, P., and Garner, S. (2006). Problem Distributions in a cs1 Course. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, pages 165–173. Australian Computer Society, Inc.

Rocha, P. S., Ferreira, B., Monteiro, D., Nunes, D. d. S. C., and do Nascimento Góes, H. C. (2010). Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino. *RENOTE*, 8(3).

Romero, C. and Ventura, S. (2010). Educational Data Mining: a Review of the State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(6):601–618.

Sommerville, I. (2010). *Software Engineering*. Addison-Wesley, Harlow, England, 9 edition.

Spohrer, J. C., Soloway, E., and Pope, E. (1985). A Goal/Plan Analysis of Buggy pascal Programs. *Human–Computer Interaction*, 1(2):163–207.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). Introduction to Data Mining. ed. *Addison-Wesley Longman Publishing Co., Inc.*

Tang, C., Lau, R. W., Li, Q., Yin, H., Li, T., and Kilis, D. (2000). Personalized Courseware Construction Based on Web Data Mining. In *Web Information Systems Engineering, 2000. Proceedings of the First International Conference on*, volume 2, pages 204–211. IEEE.

Usai, A., Pironti, M., Mital, M., and Aouina Mejri, C. (2018). Knowledge Discovery out of Text Data: a Systematic Review via Text Mining. *Journal of Knowledge Management*, 22(7):1471–1488.

Van Solingen, R., Basili, V., Caldiera, G., and Rombach, H. D. (2002). Goal Question Metric (GQM) approach. *Encyclopedia of software engineering*.

Winne, P. H. and Baker, R. S. (2013). The Potentials of Educational Data Mining for Researching Metacognition, Motivation and Self-Regulated Learning. *JEDM| Journal of Educational Data Mining*, 5(1):1–8.

Wu, S.-T. (2007). *Knowledge Discovery Using Pattern Taxonomy Model in Text Mining*. PhD thesis, Queensland University of Technology.

Wu, X., Zhu, X., Wu, G.-Q., and Ding, W. (2014). Data Mining with Big Data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107.