UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Bruno de Melo Ghisi

Análise De Ferramentas De Aprendizado De Máquina Voltadas Para A Melhoria De Desempenho Dos Estudantes De Programação

Trabalho de Graduação

Recife

2018

Bruno de Melo Ghisi

ANÁLISE DE FERRAMENTAS DE APRENDIZADO DE MÁQUINA VOLTADAS PARA A MELHORIA DE DESEMPENHO DOS ESTUDANTES DE PROGRAMAÇÃO

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Patrícia Cabral de Azevedo Restelli Tedesco

Recife

2018

Bruno de Melo Ghisi

ANÁLISE DE FERRAMENTAS DE APRENDIZADO DE MÁQUINA VOLTADAS PARA A MELHORIA DE DESEMPENHO DOS ESTUDANTES DE PROGRAMAÇÃO

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: _____/_____/_____

_____

Orientador: Patrícia Cabral de Azevedo Restelli Tedesco

_____

Examinador: Ricardo Bastos C. Prudencio

PARECER

AGRADECIMENTOS

Gostaria de primeiramente agradecer aos meu pais por sempre acreditarem nas minhas escolhas e por estarem sempre me apoiando.

Aos meus irmãos Rodrigo, Victor e Rayssa pelos conselhos, pelas conversas, pela ajuda mútua e pelo incentivo dado para que o melhor que aconteça ao próximo.

À amiga Maria Fernanda, por toda a ajuda, dúvidas sanadas, tempo e esforço disponibilizados e paciência para que fosse possível a conclusão deste trabalho.

À Professora Patrícia Tedesco, por todas as chances e oportunidades cedidas, toda a calma e paciência, além da confiança ao se tornar a minha orientadora.

Enfim, a todos os amigos, colegas, professores, que cobraram, perguntaram, me deram suporte nos momentos de frustrações ou de qualquer forma contribuíram na realização deste trabalho.

RESUMO

São poucos os estudantes que acham que aprender a programar na primeira linguagem de programação é uma tarefa fácil, pois nota-se que nas turmas iniciantes do curso de computação, a dificuldade é aparente de forma generalizada (Jenkis, 2002).

Existem muitos fatores profundamente interligados com as expectativas (Jenkis, 2002), atitudes e experiências anteriores do corpo docente e seus alunos. Por essa razão, muitos professores de programação procuram desenvolver um ambiente de aprendizado onde todos os alunos aprendam a programar rapidamente e bem, e para isso, adotam ferramentas de aprendizagem de máquina, que analisam os padrões das maiores dificuldades enfrentadas pelos alunos, e desenvolvem um método para sanar ou aliviar este problema.

Neste sentido, este trabalho tem como objetivo fornecer uma análise de ferramentas de aprendizagem de máquina existentes, consideradas relevantes, e assim, destacar os pontos positivos e negativos de cada ferramenta. Desta forma, será possível propor as melhores estratégias e soluções para que futuramente sejam implementadas melhores ferramentas de aprendizagem voltadas a estudantes de programação.

Palavras-chave: Aprendizagem de Máquina, Ensino de Programação, Tecnologia Assistiva

ABSTRACT

There are few students who find that learning to program in the first programming language is an easy task, as it is noticed that in the beginner classes of the computer course, the difficulty is apparent in a generalized way (Jenkis, 2002).

There are many factors deeply intertwined with the expectations, attitudes, and past experiences of faculty and their students (Jenkis, 2002). For this reason, many programming teachers seek to develop a learning environment where all students learn to program quickly and well, adopting machine learning tools that analyze the patterns of difficulties and complexities that students are faced with.

In this sense, this work aims to provide an analysis of existing machine learning tools, considered relevant, and thus highlight the positive and negative points of each tool. In this way, it will be possible to propose the best strategies and solutions so that in the future better learning tools for programming students are implemented.

Keywords: Machine Learning, Programming Teaching, Assistive Technology

Sumário

1. INTRODUÇÃO

O objetivo deste capítulo é apresentar os fatores que ajudaram na concepção deste trabalho a partir da seção 1.1, quais são os objetivos propostos na seção 1.2, e apresentar como o mesmo encontra-se estruturado, na seção 1.3.

1.1. Motivação

Um dos maiores desafios para os programadores iniciantes é detectar erros e interpretá-los (Hartmann, MacDougall, Brandt, Klemmer, 2010). Isso ocorre pelo fato de que os erros contidos no trabalho desenvolvido ou são muito sutis ou muito complexos para serem detectados pelo olho humano (Blikstein, 2011). Além disso, esses erros muitas vezes acabam desestimulando os alunos a continuarem aprendendo.

A aprendizagem é um processo lento que ocorre de forma incremental, ocorre com um processo de assimilação de determinados conhecimentos e modos de ação física e mental, organizados e orientados no processo de ensino e aprendizagem (Mota, Pereira, 2013), desta forma, sendo requisitado dos alunos paciência e persistência. Para auxiliar isso, hoje existem muitas ferramentas de aprendizagem de máquina que contém técnicas automatizadas para avaliar, analisar e visualizar o desempenho de alunos que estão aprendendo programação. Essas ferramentas possuem diferentes métodos, técnicas e abordagens, mas que apesar de seguirem caminhos diferentes, elas visam um objetivo em comum: manter o interesse e foco dos alunos sobre o aprendizado e aumentar a curva de aprendizagem.

1.2. Objetivos

O objetivo principal é realizar uma revisão sistemática da literatura para identificar o estado da arte com relação às ferramentas de aprendizagem de

máquina que já existem, cujo viés é de auxiliar alunos que estão aprendendo programação, e assim, destacar os pontos positivos e negativos das ferramentas que mais se mostraram relevantes. Por fim, também serão propostas melhores estratégias e soluções para que futuramente sejam implementadas melhores ferramentas de aprendizagem voltadas a estudantes de programação.

1.3. Estrutura do Trabalho

Este trabalho está dividido em 6 capítulos. No capítulo 1, será apresentado uma introdução, representando as primeiras impressões do trabalho. A seguir, no segundo capítulo, são apresentados os conceitos básicos necessários para o entendimento deste trabalho, como Aprendizagem de Máquina, como é realizada a automação das ferramentas analisadas e mostrar o cenário atual do aprendizado dos estudantes de programação, além de discutir onde e como esse aprendizado pode ser melhorado.

O capítulo 3 apresenta uma revisão sistemática sobre o tema, com o objetivo de ajudar a compreender o estado da arte da literatura relacionada às ferramentas de aprendizagem de máquina que auxiliam estudantes de programação. O protocolo criado para a condução da revisão é descrito no capítulo e posteriormente são apresentados os principais resultados obtidos.

A partir desses resultados adquiridos pela revisão, que são descritos no capitulo 4. Os mesmos são utilizados para propor, no capítulo 5, melhorias na estratégia de criação de novas ferramentas que visam ajudar os estudantes de programação mostrando todos os desafios encontrados, além de dar sugestões para trabalhos futuros. E por fim, no capitulo 6, é dada uma breve conclusão.

2. CONCEITOS FUNDAMENTAIS

Esta seção pretende proporcionar uma breve introdução dos assuntos técnicos abordados nesta revisão, para proporcionar ao leitor um melhor entendimento dos assuntos que compõem esta pesquisa. Para isso, primeiro será iniciado com a seção 2.1. que contém informações sobre Aprendizagem de Máquina, em seguida, na seção 2.3. Cenário do Aprendizado dos Estudantes de Programação, são descritas as maiores dificuldades que os alunos encontram hoje em dia, pois é justamente sobre os problemas identificados que surge uma oportunidade para ocasionar a melhoria de desempenho.

2.1. Aprendizagem de Máquina

A inteligência artificial é uma parte da ciência da computação que tenta tornar os computadores mais inteligentes. Um dos requisitos básicos para qualquer comportamento inteligente é aprender. A maioria dos pesquisadores hoje concorda que não há inteligência sem aprendizado. Portanto, o aprendizado de máquina (Shavlik e Dietterich, 1990; Michie, Spiegelhalter e Taylor 1994; Mitchell, 1997; Michalski e Chilausky, 1998) é um dos principais ramos da inteligência artificial e, de fato, é um dos subcampos de desenvolvimento mais rápido de pesquisa de IA (Jones, 2017).

O nome aprendizagem de máquina foi concebido por Arthur Samuel em 1959. Segundo Samuel a aprendizagem de máquina é o campo de estudo que concede aos computadores a habilidade de aprender sem ser programado (Samuel, A. L., 1959). Sendo considerada uma área que explora o estudo e a construção de algoritmos que aprendem sobre dados e fazem predições sobre os mesmos (Kohavi e Provost, 1998).

Hoje, ainda não há um consenso entre os pesquisadores do que seria a definição exata de aprendizagem de máquina, mas Tom Michell, um pesquisador que contribuiu no avanço de machine learning, criou uma definição mais usual para explicar o seu conceito. Segundo ele, “um programa de computador é dito para aprender com a experiência E com relação a alguma classe de tarefas T e medida de desempenho P, se o seu desempenho em tarefas em T, medida pelo P, melhora com a experiência E” (Mitchell, 1997).

Ao pensar em uma máquina programada para aprender a jogar xadrez, ela será composta por um conjunto de dados de jogos e jogadas variadas, em que a cada jogada é medida a probabilidade de se ganhar. Quanto maior a experiência da máquina, ou seja, mais ela é treinada, melhor será seu desempenho e maior será a probabilidade da máquina correspondente terminar vencedor.

E o objetivo de treinar uma máquina a ponto de vencer um humano foi atingido no ano de 1997, quando o russo Garry Kasparov, que era o Mestre Enxadrista número 1 do mundo, foi derrotado pelo computador *Deep Blue*[1]. Este fato se tornou a primeira vez que uma máquina a venceu o número 1 de xadrez. Isso se deve ao fato que a máquina conseguiu aumentar seu desempenho e melhorar sua precisão, apenas tendo um aumento na própria experiência.

2.2. Cenário do Aprendizado dos Estudantes de Programação

Muitos professores de computação sabem que a maioria dos alunos iniciantes do curso acham difícil aprender a programar, e essa característica acaba diminuindo o interesse desses alunos por programação. Isso acaba aumentando a evasão de alunos na área de computação, e hoje, o que vemos, é uma escassez de desenvolvedores capacitados no mercado de trabalho[2][3].

Hoje, grande parte da pesquisa existente, pelo menos na literatura sobre educação em computação, concentra-se em maneiras novas e interessantes de ensinar programação. Como usar alguns adereços visuais com o objetivo de engajar o público, ou focar o aprendizado aplicando à construção de algum jogo. Estas podem parecer ser boas estratégias, mas dificilmente apresentará alguma evidência de que esses métodos de ensino têm algum impacto relevante sobre a aprendizagem. É possível que essas formas de ensinar engaje os alunos, mas há poucas evidências mostrando que o conhecimento adquirido por eles não seria o equivalente ao de uma aula tradicional.

[1] https://www.scientificamerican.com/article/20-years-after-deep-blue-how-ai-has-advanced-since-conquering-chess/
[2] https://www.otempo.com.br/capa/economia/tecnologia-gera-vagas-mas-falta-profissional-qualificado-1.1594313
[3] https://exame.abril.com.br/carreira/brasileiros-qualificados-nestas-areas-estao-em-falta-veja-o-que-estudar/

É sugerido, que uma das melhores estratégias seria explorar as literaturas da psicologia e ciência cognitiva aplicadas ao aprendizado de programação, entender qual seria o motivo da dificuldade no aprendizado para programar encontrado em tantos estudantes, e como os educadores podem aplicar isso para que o ensino ocorra de forma efetiva.

Acredita-se que a aplicação de aprendizagem de máquina no ensino de programação ajudaria muito os alunos que estão tendo o primeiro contato com programação, pois a partir de padrões, a máquina pode identificar o perfil cognitivo do aluno e focar mais nas áreas que o mesmo apresentaria maior dificuldade, retornando consequentemente um resultado positivo.

3. REVISÃO SISTEMÁTICA

Uma revisão sistemática é uma forma de avaliar e interpretar todos os estudos relevantes para uma determinada pesquisa, tópico ou área de interesse (Kitchenham, 2004). Desta forma, será apresentado neste capítulo, uma revisão sistemática da literatura baseadas em recomendações propostas por Kitchenham & Charters (2007). Este trabalho se trata sobre as ferramentas de aprendizado e máquina voltadas para a melhoria de desempenho dos estudantes de programação, e foi conduzida para identificar o estado da arte a respeito do tema. A vantagem do uso deste tipo de revisão, é que ela tem como objetivo apresentar uma avaliação justa sobre um tópico pesquisado, usando uma metodologia confiável, rigorosa e auditável (Kitchenham, 2004), que pode ser reproduzida, caso queira ser analisada a veracidade.

Neste terceiro capítulo, são citados todos os processos e definidos os parâmetros da revisão sistemática que a ser realizada.

3.1. Metodologia de Pesquisa

Nesta seção será abordado todos os critérios utilizados ao longo de toda a revisão. Ela pode ser considerada o coração da pesquisa, pois nela foram explicitados os objetivos finais, além de outros parâmetros, como estratégias a serem abordadas, critérios de inclusão, exclusão, para selecionar as fontes literárias a serem abordadas, além de critérios de qualidade, para classificá-las de acordo o que está sendo proposto neste projeto.

3.1.1. Objetivos e Perguntas da Pesquisa

O objetivo desta revisão sistemática foi adquirir conhecimento a respeito do estado da arte, a partir de análises das ferramentas que auxiliam o desenvolvimento dos estudantes nos cursos de programação, conduzida de forma imparcial para que os melhores resultados sejam adquiridos.

Para chegar até este objetivo e auxiliar na estruturação das questões de pesquisa, foram seguidas as estratégias citadas por Kitchenham (2004), em que primeiramente é traçado o público-alvo, em seguida, observa-se qual foi a intervenção utilizada, e por fim, são traçados os resultados dos estudos. O

público alvo é composto por estudantes que estão cursando disciplinas de programação. A intervenção definida é o uso de ferramentas de Machine Learning ou automação com o foco na melhoria de desempenho do público-alvo. E para finalizar com os resultados, procuram-se estudos que propõem como resultados a comparação do desempenho de alunos antes e depois da utilização das ferramentas, ou comparação entre o desempenho das turmas que utilizaram a ferramenta ou não.

Baseado nos levantamentos traçados, foram buscadas informações relacionadas a ferramentas e análises de ferramentas que tem como objetivos, avaliar e/ou melhorar o desempenho de alunos nos cursos de programação, com a condição que a ferramenta, de alguma forma, auxilie o processo de aprendizagem dos estudantes. Após ter os objetivos bem definidos na seção 1.2, quatro questões de pesquisa, Q1, Q2, Q3 e Q4, foram formuladas para auxiliar na extração dos dados e na seleção de pesquisas.

Q1: Quais são as maiores dificuldades encontradas pelos estudantes de programação?
Q2: Quais são as estratégias usadas para contornar essas falhas e melhorar o desempenho dos alunos de programação?
Q3: Como essas técnicas e estratégias são aplicadas com a utilização da Aprendizagem de Máquina?
Q4: Como a utilização desses sistemas melhora a performance dos alunos e diminui a desistência dos mesmos?

3.1.2. Estratégias de Busca e Fontes de Dados

A busca de dados se restringiu a pesquisas mais recentes, englobando apenas as pesquisas publicadas entre o período de janeiro de 2008 até outubro de 2018. Os estudos que estão incluídos nesta revisão, foram localizados a partir de bases de busca automáticas (Tabela 1), levando em consideração pesquisas relevantes nos meios da indústria e acadêmico, além disso, foram levados em consideração apenas estudos completos e que são disponibilizados para acesso a partir do domínio da Universidade Federal de Pernambuco.

Tabela 1: Bases de busca automáticas utilizadas no processo da revisão sistemática

Base de Dados	Endereço Eletrônico
ACM Digital Library IEEE Xplore	https://dl.acm.org/ https://ieeexplore.ieee.org/

Para a localização das pesquisas, foram utilizadas strings de busca sofisticadas (Quadro 1), para gerar melhores resultados retornados pelas bases de dados utilizadas (Tabela 1). Essas strings de busca foram definidas a partir de termos de busca relacionados às questões de pesquisa (QP), e sofreu algumas alterações após uma fase de testes visando aperfeiçoar as strings cadastradas e retornar artigos mais relevantes ao nosso estudo. As buscas de teste foram realizadas entre no mês de setembro de 2018, e após definida a versão final das strings de busca, foram realizadas as buscas nos meses de outubro e novembro de 2018.

((Machine Learning) AND ((Software) OR (System) OR (Program)) AND (Assessment) AND (((Programming) AND (Students)) OR ((Programming) AND (Course))))

Quadro 1: *Strings* de busca utilizadas na revisão sistemática da literatura

3.1.3. Procedimentos de Seleção dos Estudos

Para a seleção dos artigos, foi desenvolvido um processo separado em três etapas. A primeira etapa foi a análise dos títulos dos artigos retornados pela busca, a segunda etapa foi a avaliação dos resumos, e por último, a avaliação dos artigos completos. Durante todas as etapas, foram avaliados se os artigos selecionados faziam parte dos critérios de inclusão (CI) ou de exclusão (CE).

Os artigos que satisfazem a todos os critérios de inclusão listados abaixo, foram incluídos no estudo:

- CI1: Estudos que respondam a pelo menos uma das questões de pesquisa;
- CI2: Estudos da indústria ou acadêmicos;
- CI3: Estudos primários usados como base para outros estudos;
- CI4: Estudos com delineamento experimental ou observacional;
- CI5: Estudos caracterizados como completos.

Os artigos que satisfazem a pelo menos um dos critérios de exclusão listados abaixo, foram desconsiderados dos estudos:

- CE1: Estudos que claramente tratam de outro assunto não relacionado às questões de Pesquisa;
- CE2: Estudos que não apresentam sua configuração completa;
- CE3: Estudos que não foram disponibilizados gratuitamente em sua versão completo, mesmo que acessados dentro do domínio da UFPE;
- CE4: Estudos escritos em um idioma que não seja o Português ou o Inglês;
- CE5: Estudos duplicados;

- CE6: Estudo cujo foco principal não seja relacionado ao uso de ferramentas no aprendizado e/ou ensino de programação;

3.1.4. Critérios de Qualidade

Depois de selecionados, os estudos foram avaliados por critérios de qualidade, com o objetivo de avaliar quais dos artigos filtrados, mais se adequam aos objetivos desta revisão, e desta forma, contribuir com a identificação de quais são os artigos mais relevante para esta pesquisa.

Os critérios foram inspirados e adaptados de MOREIRA (2015). A partir da aplicação dos 10 critérios de qualidade (CQ) listados abaixo. O que está sendo analisado nestes critérios de qualidade, é se por acaso, atendem às questões de pesquisa e a qualidade do trabalho em relação à estruturação, metodologia, objetivos e resultados:

• CQ1: Descreve as abordagens utilizadas que visam a melhoria do processo de aprendizado do aluno?

• CQ2: Aborda o contexto de aprendizagem de máquina?

• CQ3: As linguagens de programação abordadas são descritas?

• CQ4: Aponta problemas e dificuldades dos alunos que estão aprendendo a programar?

• CQ5: Descreve o objetivo da pesquisa de forma clara?

• CQ6: Existe uma descrição adequada do contexto?

• CQ7: A metodologia utilizada está descrita de forma objetiva?

• CQ8: Os dados coletados foram exibidos de forma clara e objetiva?

• CQ9: Os resultados condizem com os objetivos?

• CQ10: Os resultados contribuem para o estado da arte e estado da prática?

Após aplicar critérios de inclusão e exclusão, utilizamos os critérios de qualidade para poder ordenar os artigos conforme sua relevância, desta forma, este ranqueamento irá ajudar a decidir quais artigos foram mais importantes para o resultado para esta pesquisa, no caso, aqueles que tiverem maior peso, são os que mais condizem com os critérios de qualidade abordados. Cada artigo foi avaliado, sendo analisado sua correspondência a cada um dos critérios de qualidade listados acima. Dentro desses critérios, pontuamos com 1 aos artigos que corresponde amplamente ao critério de qualidade avaliado no momento, 0.5 quando corresponde parcialmente aos critérios e 0 não há correspondência alguma. As tabelas com os resultados dessa avaliação podem ser encontradas no apêndice, e a pontuação final de cada artigo, foi considerada no momento de ponderar os resultados e realizar as conclusões desta pesquisa.

4. APRESENTAÇÃO DOS RESULTADOS

Este capítulo tem o objetivo de apresentar os resultados obtidos, a partir da aplicação prática de todo o processo descrito ao longo do Capítulo 3. E em seguida, fazer uma análise sobre estes resultados.

. Na seção 4.1, são descritas as principais informações colhidas dos estudos filtrados pelo processo de revisão sistemática. A seguir, na seção 4.2, são comentadas as ameaças que poderiam interferir ou influência na validade e relevância dos dados apresentados.

4.1. Resultado do Procedimento de Busca

A princípio, foram escolhidos diversos engenhos de busca para realizar a pesquisa sobre a *string* de busca definida na seção 3.1.2. Porém, foi constatado que os engenhos de busca da ACM Digital Library[4] e IEEE Xplore[5] apresentavam resultados mais concretos, composto por estudos mais completos, além do fato de possuírem um uma grande credibilidade na comunidade acadêmica.

Ao se realizar consultas em outras bases de dados como CEIE-BR[6], SpringerLink[7], Semantic Scholar[8] e ResearchGate[9], notava-se que a quantidade de arquivos retornados eram muito além ou aquém do esperado, alguns apresentando retorno de dezenas de milhares de artigos, onde a maioria era considerado irrelevante apenas pela análise de títulos. Por isso, foi julgado que a adição das outras bases de dados para este estudo, tornaria a revisão sistemática muito mais trabalhosa, com pouco retorno relevante.

Além disso, ao adaptar a *string* de busca para que fosse retornado um resultado mais robusto nas outras bases de dados, como consequência, prejudicava os resultados retornados nos engenhos de busca que estão sendo usados nesta revisão, que são o ACM e IEEE. Outros motivos que levaram à escolha dessas bases, foi a disponibilidade de quase todos os estudos, de forma gratuita e com sua configuração completa, ao serem acessados por meio da rede da UFPE. Comparando as duas bases de dados usadas neste estudo, acredito

[4] ACM Digital Library - https://dl.acm.org/
[5] IEEE Xplore - https://ieeexplore.ieee.org/
[6] CEIE-BR - http://www.br-ie.org/pub/index.php/index
[7] SpringerLink - https://link.springer.com/
[8] Semantic Scholar - https://www.semanticscholar.org/
[9] ResearchGate - https://www.researchgate.net/

que o ACM retornou mais resultados relevantes, pois os estudos disponíveis continham um enfoque maior em aprendizagem de máquina.

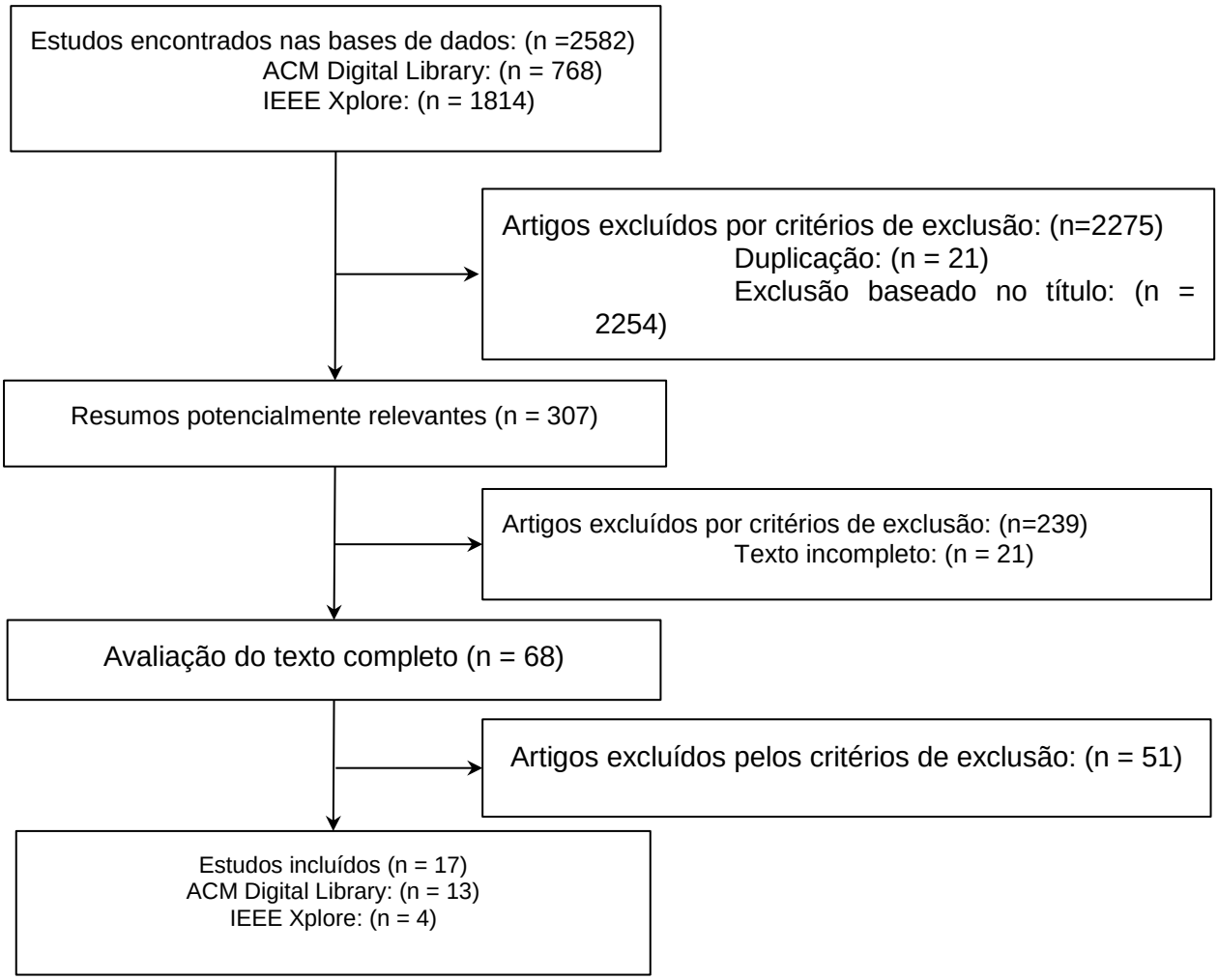

Figura 1: Fluxograma do processo de seleção dos estudos

Após realizar a pesquisa nas bases de dados, foi efetuada a avaliação dos artigos em três etapas, conforme descrito na Seção 4.1.3. A Figura 1 apresenta as etapas do processo de avaliação de artigos, podendo ser divido em três fases. Primeiramente foram identificados 2582 artigos, sendo 768 artigos pertencentes à ACM e 1814 artigos pertencentes ao IEEE Xplore.

A partir deste resultado, foi constatado que 21 encontravam-se em ambas as bases de dados, sendo excluídos por duplicação. Ao final da análise dos títulos dos artigos, 307 estudos foram considerados potencialmente relevantes. Após realizar a análise dos resumos, 68 foram selecionados para a avaliação do texto completo. No final do processo de seleção, 17 estudos preencheram os critérios preestabelecidos.

De acordo com a Tabela 1 abaixo, é possível notar que a maioria dos estudos analisados foram extraídos da Base de Dados ACM Digital Library (76,47%),

Tabela 1: Proporção de artigos selecionados por base de dados.

Base de Dados	Número de artigos n (%)	ID
ACM Digital Library	13 (76,47%)	E01, E02, E03, E04, E05, E06, E07, E08, E09, E10, E11, E12, E13
IEEE Xplore	4 (23,53%)	E14, E15, E16, E17
Total	17 (100%)	--

ID: Identificação do estudo.

Ao se analisar o ano de publicação, pode-se notar que há um ganho muito grande de interesse sobre a utilização de ferramentas para ajudar no ensino de programação. Mais da metade dos estudos selecionados, foram publicados nos últimos 3 anos, correspondendo a 10 artigos ou 58,82%, mesmo sendo considerado como um dos critérios de exclusão, inclusão de artigos publicados nos últimos 15 anos. Isso mostra que o assunto está em grande ascendência, e assim, nota-se uma tendência que essa área ainda irá crescer muito num futuro próximo.

Tabela 2: Proporção de artigos selecionados por ano de publicação.

Ano de Publicação	Número de Artigos n (%)	ID
2004	1 (5,88%)	E14
2006	1 (5,88%)	E15
2011	1 (5,88%)	E05
2012	1 (5,88%)	E02
2013	2 (11,76%)	E12, E16
2015	1 (5,88%)	E17
2016	3 (17,65%)	E04, E06, E08
2017	1 (5,88%)	E07
2018	6 (35,29%)	E01, E03, E09, E10, E11, E13
Total	17 (100%)	--

ID: Identificação do estudo.

Como a maior parte dos artigos selecionados foram extraídos da base de dados da ACM Digital Library, isso acabou influenciando também no tipo de publicação dos artigos, pois pude notar que a maior parte dos artigos presentes na ACM foram publicados em conferências, enquanto a maior parte dos artigos presentes no IEEE Xplore, foram publicados em periódicos.

Tabela 3: Proporção de artigos selecionados por tipo de publicação.

Tipo de publicação	Número de Artigos n (%)	ID
Periódico	5 (29,41%)	E06, E10, E14, E15, E17
Conferência	12 (70,59%)	E01, E02, E03, E04, E05, E06, E07, E08, E09, E11, E12, E13, E16
Total	17 (100%)	--

ID: Identificação do estudo.

4.1.1. Principais Pontos Analisados de Cada Artigo

E01: Apresenta um sistema chamado Artemis, que procura facilitar o aprendizado do aluno usando uma ferramenta visual, que ajuda no entendimento do código, tudo isso de forma interativa. É multilinguagem, ou seja, não foi feito apenas para uma linguagem específica. Ele não procura trabalhar sobre a dificuldade do aluno, e sim facilitar o entendimento. Ou seja, funciona mais de forma preventiva, do que uma remediação.

E02: É apresentado o Aari, um algoritmo que utiliza de técnicas de machine learning para avaliar a solução dos alunos e encontrar padrões de ineficiência de implementação. O foco deste algoritmo é avaliar a implementação de alguns algoritmos de ordenação, tendo uma eficiência maior que 90% na avaliação das implementações realizadas pelos alunos. Esta aplicação seria muito útil para avaliar os alunos, diminuiria o esforço do professor e o deixaria mais livre para focar nos casos mais incomuns, ou até mesmo identificar padrões de problemas de implementação que afetam os alunos de forma geral, como a utilização desnecessária de swaps que afetam o desempenho das ordenações, e foi citado no artigo.

E03: Fala sobre o Auto Item Genereration (AIG), que é um gerador de questões a partir de um template pré-definido. Consegue gerar várias questões de exercício, apenas alterando as variáveis. Esse estudo realiza um teste sobre dois grupos, mas apenas abordando de formas distintas.

E04: O ViLLE é descrito como uma ferramenta que cria automaticamente exercícios de programação em java, de forma interativa. Como ele retorna feedback instantâneo do desempenho do aluno, permite que o aluno corrija imediatamente o próprio erro. Muitos dos erros de sintaxe cometidos pelos alunos e identificados no sistema estavam relacionados a objetos, que não estava no escopo do curso. Porém, ao ser identificado este padrão, deixa os professores mais alertas sobre este problema, que nas aulas podem enfatizar mais esse tema em aulas futuras, e assim, evitar que dificuldades semelhantes voltem a acontecer com os alunos.

E05: O sistema Netlogo mapeou as atividades dos alunos e fez uma análise sobre esses dados. Consegue fazer uma análise sobre os sentimentos dos alunos, identificar onde eles estão errando mais e conseguiu inferir algumas

outras características da forma de programar, caracterizou os tipos de alunos em três grupos, a partir de análises da linha do tempo que o aluno levava programando, além das tentativas de compilação, quantidade de erros e em que momento ocorriam esses acontecimentos. Como a base de alunos analisados foi muito pequena, os resultados retornados pelo artigo são mais qualitativos e pessoais do que quantitativos.

E06: IPRACTICE, uma ferramenta com o objetivo de auxiliar os alunos que estudam em campus urbanos e enfrentam longo tempo de deslocamento para a aula. Por isso, foi criado uma aplicação para smartphones, que além de conter trechos do livro que ensinam programação, os professores consegue propor exercícios da aplicação como uma forma de avaliação. Após a implementação do app, foi notado um aumento no número de alunos aprovados na disciplina e melhoria na qualidade dos projetos de finais de semestre.

E07: Tem o objetivo de identificar quais conceitos os pesquisadores e profissionais consideram importantes nos cursos de introdução a programação, e examinou o alinhamento entre esses conceitos, os objetivos de aprendizagem estabelecidos para os cursos típicos de programação introdutória e as avaliações usadas para avaliar a aprendizagem dos alunos.

E08: APA, ferramenta que auxilia o teste de códigos, desta forma, ajuda o aluno a encontra bugs no código e melhorar seu código. Com a rápida localização dos erros, diminui a frustração do aluno, pois ele pode ajudar a desenvolver habilidades de programação e *debug* de códigos. Porém, ele funciona apenas da forma “tentativa e erro”.

E09: SmartAPE, é um sistema de gerenciamento sobre aprendizado, analisa automaticamente o código enviado pelo aluno a ponto de identação do código, acerto do algoritmo e avisos. Realiza a comparação entre duas turmas consecutivas. Na primeira turma, o uso da ferramenta pelos alunos foi aplicado de forma voluntária e na turma seguinte, de forma obrigatória. Após a realização de um questionário e comparação das respostas, notou-se que mais usuários se sentiam desmotivados quanto ao uso do software, que na primeira turma. Acredita-se que isso se deve ao fato de ter tornado o uso do sistema obrigatório, como forma de avaliação. Porém a turma que utilizou este sistema de forma obrigatória, teve um desempenho maior.

E10: HAAP, uma plataforma que retorna continuamente feedback aos alunos, também auxiliam os professores a fazer avaliação de desempenho. Com isso, os professores conseguem tomar rápidas decisões, caso desejem dar um suporte maior aos alunos com desempenhos inferiores.

E11: O artigo contém uma abordagem que avalia as etapas de aprendizagem individuais, além de fornecer várias avaliações formativas e somativas do desempenho de aprendizado dos alunos. Com o mapeamento adequado entre os resultados, foi percebida uma melhoria no desempenho e envolvimento dos alunos, além do impacto significativo na avaliação dos próprios alunos.

E12: Este artigo primeiramente realiza um teste adaptativo de computador, a partir de várias simulações numéricas, e depois aplica em uma turma composta por cem alunos. Porém o sistema se encontra muito defasado e necessita de muitos ajustes para notar uma melhoria eficaz no desempenho dos alunos. Os testes realizados numericamente não tiveram resultados semelhantes aos aplicados a uma turma real.

E13: Este artigo desenvolveu um modelo para projetar um sistema de aprendizagem inteligente para qualquer linguagem de programação usando redes Bayesianas. O modelo de design do sistema de tutoria considera um modelo de usuário usando o modelo de estudante. A rede bayesiana foi utilizada para avaliar o estado atual do conhecimento do aluno para que o modelo possa ajustar e apresentar novos conhecimentos para melhorar o desempenho do aluno como um resultado em um ambiente de *e-learning*.

E14: ACME, um Sistema online que ajuda o aprendizado do aluno. Teve bom resultado ao comparar resultados entre os alunos que utilizaram o sistema e não utilizaram, porém, apresenta o conteúdo de forma muito dispersa, dificultando a inteligibilidade do artigo.

E15: GAME, ele analisa comentários, identação e estilo de programação. Sua intenção é estimular o aluno a manter boas práticas de programação. C, C++ e Java.

E16: NoobLab, oferece exercícios para os estudantes a partir da aplicação de gamefication. Ele gera um log de todas as atividades geradas pelos alunos, destacando os erros cometidos e ajudando na análise para se entender o que o levou a cometer tal erro. Ao longo deste artigo, são abordados resultados de

estudantes que acompanhavam o conteúdo oferecido pelo sistema, e também daqueles alunos que pulavam o conteúdo por ter a impressão de já conhecê-lo. Como resultado, notou-se que os aluno que normalmente pulavam conteúdo, tinham maiores dificuldades em avançar e concluir todos os exercícios.

E17: Sistema de avaliação do aluno automatizado, do qual consegue corrigir os exercícios propostos e dar a nota ao aluno pelo exercício feito. O programa é aplicado para alunos que tem o primeiro contato com programação, e os exercícios propostos tem estruturas simples. Há a comparação no desempenho dos alunos que utilizaram e não utilizaram o sistema e retorna um resultado positivo.

4.1.1. Respondendo às Questões de Pesquisa

Na seção 3.1.1. foram propostas questões de pesquisa que têm o objetivo de guiar o desenvolvimento deste trabalho. Para a resposta destas questões estão sendo utilizados como base, os trechos considerados mais relevantes dentre os artigos selecionados.

Q1: Quais são as maiores dificuldades encontradas pelos estudantes de programação?

Há várias razões para a maioria dos estudantes sentirem dificuldade ao aprender a programar. De acordo com E04, aprender os conceitos básicos de programação é uma tarefa complicada, pois geralmente são bem diferentes do que os alunos se acostumaram. Atingir o ponto de compreender os conceitos elementares da programação (orientada a objetos) leva tempo e a natureza única da mentalidade necessária faz com que a maioria dos alunos tropecem nos mesmos equívocos sobre os conceitos básicos (Kurvinen, Hellgren, Kaila, Laakso & Salakoski, 2016).

Já na visão do E07, os cursos de programação frequentemente exigem que os alunos concluam tarefas de avaliação que são complexas e envolvem muitos conceitos heterogêneos dentro de uma única pergunta (Luxton-Reilly, McDermo, Petersen, Becker, Sanders, Cao, Muhling, Simon & Whalley, 2017).

No artigo E08 é abordado um viés diferente, que os principais problemas estão relacionados aos erros mais comuns cometidos pelos alunos, para resumir, podemos ter os seguintes tipos gerais de problemas com a descrição

dos exercícios: tratamento de erros indefinido, sequência imprecisa de ações e formato de saída indefinido (Tang, Smith, Warren & Rixner, 2016).

Uma explicação simples e direta é observada em E14, onde é afirmado que a base de suas dificuldades é a falta de habilidades cognitivas e conhecimentos necessários para realizar as tarefas básicas para desenvolver um programa (Boada, Soler, Prados & Poch, 2004).

Q2: Quais são as estratégias usadas para contornar essas falhas e melhorar o desempenho dos alunos de programação?

Dentre algumas das técnicas bastante utilizadas, estão os feedbacks instantâneos como em E01, pois acredita-se que ao se concentrar no feedback imediato de exercícios, é melhorada a experiência de aprendizado em turmas grandes, e assim, permite que os alunos reflitam e aumentem seu conhecimento de forma incremental (Krusche & Seitz, 2018).

No artigo E04 explora-se de forma mais aprofundada os erros mais cometidos pelos estudantes, pois ao identificar alguns dos erros mais práticos que os alunos cometem durante o primeiro curso de programação, pode-se ajudar os educadores a concentrar melhor sua atenção e melhorar seus exercícios (Kurvinen, Hellgren, Kaila, Laakso & Salakoski, 2016).

O E05 aposta num viés semelhante, onde é fornecido um estudo de caso exploratório sobre a possibilidade de usar análise de aprendizado e mineração de dados educacionais para inspecionar o comportamento e a aprendizagem dos alunos em ambientes de aprendizado construcionistas, não programados e construtivos, nos quais os métodos de avaliação tradicionais podem não capturar a evolução dos alunos (Blikstein, 2011).

Em E11, o sistema se adapta às necessidades e desempenho do aluno, no qual é determinado o caminho de aprendizagem de cada estudante com base em várias avaliações sobre o estado atual de conhecimento ou desempenho do discente. Portanto, um aluno pode ter seu próprio caminho de aprendizado diferente dos outros estudantes. Com uma continua coleta de dados sobre o desempenho e os resultados dos aprendizes, o caminho ou mapa de aprendizado pode ser modificado para garantir que tanto o conteúdo quanto o sequenciamento estejam possibilitando o sucesso do aluno (Cai, 2018).

Numa linha semelhante, mas usando outra forma de implementação, o E13 apresenta um modelo que descreve melhor o nível de aprendizado do

discente com base no que ele sabe antes e o que ele entende depois de aplicar o que aprende ao responder perguntas consideradas como um teste para medir seu nível de desempenho. Isto é onde a Rede Bayesiana desempenhou um grande papel, pois fornece o método no qual a entrada para o protótipo pode ser usada para determinar as probabilidades na rede (Alday, 2018).

Q3: Como essas técnicas e estratégias são aplicadas com a utilização da Aprendizagem de Máquina?

Diferentes técnicas e estratégias foram aplicadas, dentre as consideradas mais relevantes encontra-se o E02, em que é empregado um método supervisionado de classificação de aprendizado de máquina. Na aprendizagem supervisionada, primeiro um conjunto de instâncias conhecidas, chamado conjunto de treinamento, é introduzido no sistema. O sistema classifica cada instância do conjunto, associa cada classe aos atributos de cada instância e aprende a qual classe cada instância pertence. Com base no que o sistema treinado aprendeu na fase de aprendizado, ele é capaz de classificar instâncias anteriormente não vistas de um conjunto de dados na fase de teste ou avaliação (Taherkhani, Korhonen & Malmi, 2012).

No E03 é apresentada uma abordagem AIG (Geração Automática de Itens em inglês) baseada em semântica para a geração automática de exercícios de programação contextualizada que podem ser usados para questionários e trabalhos de casa em cursos introdutórios de programação. A abordagem amplia o AIG, baseado em modelo tradicional, conectando-se a existentes fontes de dados abertos para gerar diferentes contextos para um modelo de exercício de prática de programação (Zavala & Mendoza, 2018).

No caso de E13, é utilizado no estudo um modelo interativo, no qual a probabilidade individual das respostas é computada e, em seguida, é usada como entrada no modelo para que a avaliação precisa do desempenho do aluno seja alcançada. Na sequência, é realizada uma avaliação do aluno usando a rede bayesiana, em que um sistema de avaliação determina o que um aluno sabe. Essa informação é usada por um assessor (ou seja, o professor do aluno, o aluno, pesquisadores da educação, etc...) para tomar decisões. Com uma avaliação confiável de seu desempenho, é produzido um modelo de estudante, que é essencialmente um programa de computador baseado em regras que

calcula respostas a problemas reais da mesma forma que o discente (Alday, 2018).

Q4: Como a utilização desses sistemas melhora a performance dos alunos e diminui a desistência dos mesmos?

Cada artigo conseguiu atingir o mesmo objetivo, que era melhorar o desempenho dos alunos, e consequentemente, diminuir a desistência nos cursos de programação utilizando diferente abordagens. Segundo o E03, ao se comparar os resultados obtidos por um grupo de intervenção e por um grupo de controle, observa-se os benefícios de dedicar tempo extra à prática. Os estudantes que participaram da estratégia de intervenção apresentaram uma melhoria maior no aprendizado de código sobre os alunos do grupo de controle. Isso ocorreu apenas utilizando uma abordagem AIG baseada em semântica, que gera exemplos com contextualização simples (Zavala & Mendoza, 2018).

Em relação aos resultados obtidos em E10, observou-se que forçar os alunos a enviar soluções para problemas de programação não tem nenhum efeito sobre os resultados do exame. Em vez disso, os alunos têm um desempenho melhor se quiserem praticar sozinhos. Portanto, o ideal é aumentar a própria motivação de um aluno (Almeida, Cunha, Macedo, Pacheco & Proença, 2018).

Com o objetivo de melhorar a performance dos aluno o E13 apresenta um método de desenvolvimento do modelo de estudante para qualquer classe de programação obtida usando um protótipo simples, que aprimora as estratégias de ensino dos membros do corpo docente na faculdade, pois descreve as necessidades de aprendizado dos alunos em geral, o que também podem servir de base para qualquer aula de programação no futuro (Alday, 2018).

Ao analisar os resultados do E17, foi possível perceber que os alunos enviaram seus programas pelo menos duas vezes em média (uma submissão e vários reenvios) e o número de reapresentações ficou alto para tarefas difíceis. Isso significa que os alunos melhoraram seus programas repetindo os envios especialmente para tarefas difíceis (Akahane, Kitaya & Inoue, 2015).

4.1.2. Tendência do Conteúdo dos Artigos Selecionados

Ao avaliar todos os artigos selecionados, foi constatado que a utilização de sistemas para auxiliar os estudantes de programação não exclui a presença de um professor em sala de aula, e sim, é complementar às aulas tradicionais.

É feita uma comparação da utilização desses tipos de sistemas entre dois grupos diferentes, um que apenas assistia às aulas, e outro que complementava os estudos com algum sistema proposto pelo tutor, e pôde-se perceber um resultado positivo ao se **testar a efetividade** a partir da comparação os dois grupos, seja com um aumento da média da turma que utilizava o sistema complementar, ou seja um índice superior de aprovação. Essa comparação, que tem o objetivo de provar que se trata de uma ferramenta eficiente no que é proposta, está presente em 8 dos 17 artigos analisado, 47% (E01, E02, E03, E04, E06, E08, E10, E11).

Já em outros artigos, é possível perceber que o foco dele é fazer uma análise do desempenho dos alunos, para fornecer **feedbacks instantâneos** do desempenho dos estudantes. Para manter os níveis de motivação e conforto, os alunos também precisam de orientação adequada e feedback frequente o mais cedo possível, para que possam aprender a entender e corrigir defeitos em seus programas, formando hipóteses e testando-os experimentalmente (Bergin e Reilly, 2005). Este cenário ajuda os estudantes a corrigir rapidamente os erros que estão sendo cometidos e evitam a reincidência dos mesmos, cenário presente em 3 das ferramentas revisadas (E02, E10, E16).

Alguns dos sistemas analisados, foram feitos para possuir suporte a diversas linguagens de programação, outros foram implementados para dar assistência no aprendizado dos alunos sobre algumas linguagens de programação específica, já outros, são apenas estudos sobre pseudocódigos. Porém, algumas ferramentas com suporte a diversas linguagens, foi notado que alguns acabavam apesentando alguns *bugs* que acabaram afetando o resultado das pesquisas, isso se deve ao fato de haver diferenças nas sintaxes, sendo necessário um grande esforço para a correção do mesmo. Dentre os sistemas que descrevem explicitamente as linguagens avaliadas, encontramos como a **linguagem de programação mais analisada**, o Java, presente em 8 estudos (E02, E04, E07, E09, E14, E15, E16, E17).

O método que alguns sistemas procuram melhorar o desempenho dos alunos, é a partir do **fornecimento de material de ensino**, presente no software adotado pelo professor. A principal função desse conteúdo, é apresentar um ensino mais direcionado para embasar o aluno a fim de deixa-lo apto a resolver os exercícios que são apresentados em sequência, aspecto presente em 7 artigos (E06, E09, E10, E11, E14, E15, E16).

Um aspecto também considerado bastante importante é diminuir a carga de trabalho do professor, que muitas vezes tem que corrigir centenas de exercícios manualmente, todos os artigos abordados nesta revisão propõem exercícios aos estudantes, mas pude notar que alguns **detectam padrões**. A partir de uma avaliação automatizada dos códigos implementados pelos estudantes, ou até mesmo a partir do log de monitoramento do aluno resolvendo o exercício, permitiu que fossem identificados comportamentos que causam a dificuldade no aprendizado, como por exemplo, a categorização da forma do alunos programarem, de problemas presentes no código, ou principalmente, a partir dos erros mais comumente cometidos pelos alunos. Desta forma, o tutor consegue perder menos tempo com correções e focar mais na resolução dos problemas analisados. Dos artigos que detectam algum tipo de padrão, nós temos 5 artigos abordando isto diretamente (E02, E12, E13, E15, E16).

4.2. Ameaças à Validade do Estudo

Após a revisão sistemática ter o processo bem definido no capítulo 3 e ser executada de acordo com a seção 4.1, foi constatado que há algumas limitações que ameaçam a validade do estudo.

A primeira ameaça envolve o fato que as bases de dados escolhidas, podem não retornar os estudos mais relevantes para a área.

A segunda ameaça ocorre na condição da *string* de busca utilizada não tenha sido a mais adequada para a realização deste estudo, e assim, não englobar os estudos mais relevantes.

A terceira ameaça pode estar relacionada ao fato que foram encontrados muito artigos que não apresentavam sua configuração completa, ou seja, estava disponibilizado apenas o resumo de forma gratuita. Ao analisar o conteúdo destes resumos, muitos apresentavam informações muito relevantes, que na minha opinião, apresentavam grande potencial de sua configuração completa integrar esta revisão, mas que foram excluídos pelo critério de exclusão CE2.

O quarto viés identificado como ameaça a este estudo, seria a subjetividade do avaliador ao selecionar os estudos. Pois ao serem aplicados os critérios de inclusão e exclusão, a interpretação do próprio avaliador pode ter ocasionado na exclusão alguns estudos potencialmente relevantes.

5. DISCUSSÃO E OPORTUNIDADES FUTURAS

O objetivo deste capítulo é discutir o resultado obtido pela análise do conteúdo dos artigos selecionados, nos quais foram citados os principais pontos tratados durante todo o processo de revisão.

5.1. Discussão de Resultados

Após a analise do conteúdo dos artigos selecionados, em 4.1.1, houve uma mudança na visão inicial sobre o uso de ferramentas no ensino. Antes da realização da revisão, era esperado que a maioria do conteúdo encontrado estaria englobando os softwares funcionando de forma autônoma, ou seja, como substitutos de professores, mas ficou bem esclarecido que a melhor forma de ter um melhor aproveitamento das ferramentas, é utilizá-los como complemento. Ele serve de assistência não só do aluno, como principalmente do professor.

Sobre as características de todas os sistemas analisados, é recomendável que que algumas tendências sejam seguidas, como as destacadas em 4.1.1. É importante que ao se construir uma ferramenta baseada em aprendizagem de máquina com o objetivo de melhorar o desempenho dos alunos de programação, estas contenham ferramentas de feedback instantâneo, a princípio seja focada em uma única linguagem de programação, para evitar bugs inesperados, identifique comportamentos fora do normal, para que os tutores possam focar na correção dos problemas mais comum entre os alunos, e por fim, oferecer uma seção de ensino, para que os alunos possam fazer uma revisão mais focada na resolução dos exercícios. Com isso, o objetivo de melhorar o desempenho dos alunos tende a ser potencializado.

Hoje, as bases de dados mais influentes possuem poucos artigos que seriam realmente relevantes para este trabalho, pois foi muito custoso encontrar dentre milhares, poucos artigos que atendam aos critérios de inclusão e exclusão descritos no capítulo 3. Sobre todos estes artigos que atendem aos critérios, foi feita uma análise qualitativa, que ordena aos artigos que mais atendem às expectativas inicialmente proposta, aos que menos atendem.

5.1. Oportunidades Futuras

Apesar da área de aprendizado de máquina já ter mais de 50 anos, acredito que há uma lacuna muito grande quando se trata de sua convergência com a área de educação, por isso, sugere-se que sejam realizadas nesta área, pois a quantidade de estudo ainda é ínfima. Muitas vezes, os estudos focam apenas na geração de notas sobre provas e exercícios para os alunos, mas esquecem da parte mais importante, como auxiliá-lo a obter êxito nessa nota gerada de forma automatizada.

Felizmente é possível notar que este cenário está sendo revertido, ao se analisar este gráfico na Figura 2. Ele foi montado a partir dos estudos utilizados neste trabalho, e mostra a correspondência da quantidade de artigos por ano. Pode-se notar que mais da metade dos estudos selecionados foram publicados nos últimos 3 anos. Além disso, há uma linha listrada, que mostra uma tendência crescente.

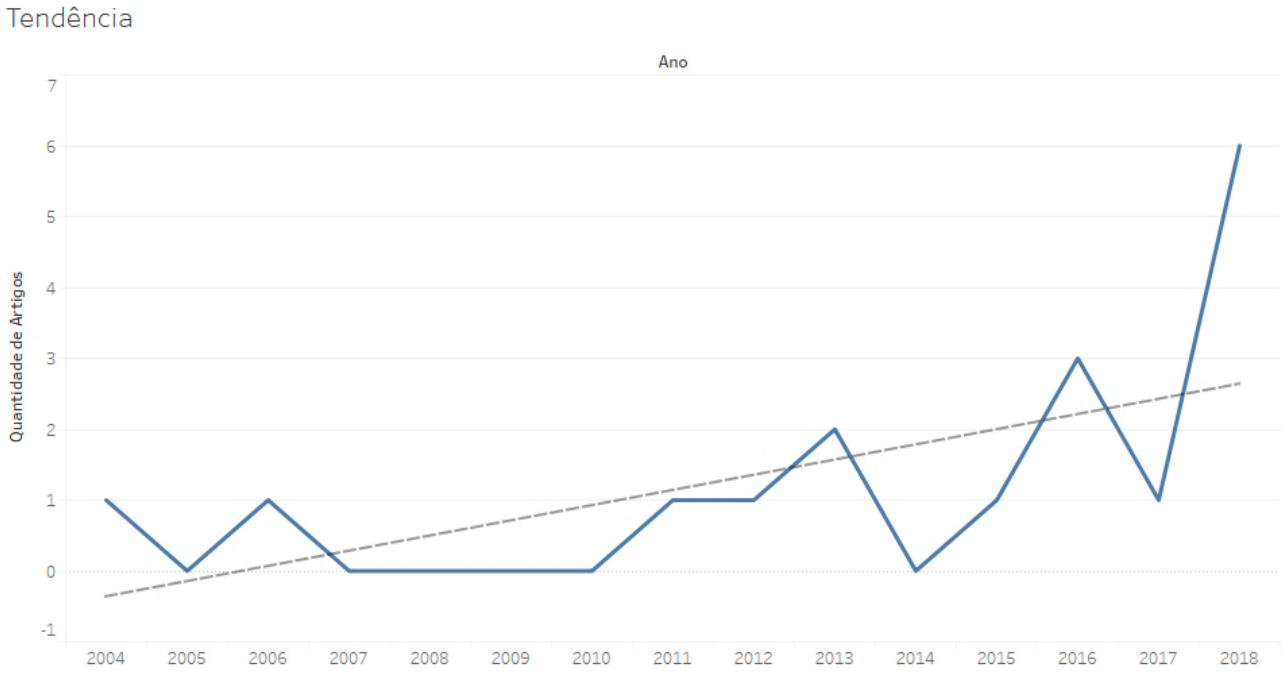

Figura 2: Quantidade de artigos analisados por ano de publicação

6. CONCLUSÃO

Foi apresentado neste trabalho uma revisão sistemática da literatura com o objetivo de identificar as principais características apresentadas nas ferramentas de aprendizado de máquina voltadas para a melhoria do desempenho dos estudantes de programação.

Baseado no resultado desta revisão, percebeu-se que diferente da ideia original, o uso desse tipo de ferramenta não tem o objetivo de substituir o professor de uma sala de aula tradicional, a função do uso desse tipo de ferramenta seria para complementar o trabalho do tutor, havendo diversas formas desta complementação ser posta em prática.

Por exemplo, uma forma de complementar o ensino em sala de aula seria estimulando o aluno a estudar e se desenvolver de forma autônoma a partir da assistência desse tipo de sistema. Como ocorre em ferramentas que utilizavam práticas de *gamification*. Outro viés seria facilitando o acesso do estudante ao conteúdo ensinado em sala de aula, expondo-os a vários exercícios relacionados ao tema recém-estudado. A partir de feedbacks instantâneos, em que o aluno possa identificar o erro e corrigir rapidamente, antes que uma dúvida ou um aprendizado ocorrido de forma distorcida se torne em um problema mais complexo.

A outra forma de complementar o trabalho do professor em aula é diminuindo a própria carga, pois foram identificadas várias ferramentas que cumpriam esta tarefa de diferentes formas, como a avaliação automatizada da implementação dos estudantes, análise dos códigos e identificação de comportamentos que causam a dificuldade no aprendizado, categorização de erros mais comumente cometidos pelos alunos, ou seja, permitindo que o trabalho do tutor seja focado na correção desses problemas analisados.

É esperado que o presente trabalho possa ser utilizado como base para novos estudos e avaliações, bem como para novas pesquisas na área de aprendizagem de máquina voltada para a área educacional.

7. REFERÊNCIAS

AKAHANE, A.; KITAYA, H.; INOUE, U.; Design and Evaluation of Automated Scoring Java Programming Assignments, IEEE, SNPD 2015, June 1-3 2015, Takamatsu, Japan

ALBRECHT, E.; GUMZ, F.; GRABOWSKI, J. Blended Learning in an Introductory Programming Course, ECSEE'18, June 14–15, 2018, Seeon/ Bavaria, Germany

ALDAY, R. B.; Bayesian Networks in Intelligent Tutoring Systems as an Assessment of Student Performance using Student Modeling, ICACS '18, July 27–29, 2018, Beijing, China, 2018

ALMEIDA, J. A.; CUNHA, A.; MACEDO, N.; PACHECO, H.; PROENÇA, J. Teaching How to Program using Automated Assessment and Functional Glossy Games

BERGIN, S.; REILLY, R. The influence of motivation and comfort-level on learning to program. In Proceedings of the 26th Annual WorkShop of the Psychology of Programming Interest Group, Vol. 17. PPIG, 293, 2005

BLIKSTEIN, P. Using learning analytics to assess students' behavior in open-ended programming tasks. *Transformative Learning Technologies Lab*, pp. 110-116, 2011

BLUMENSTEIN, M.; GREEN, S.; FOGELMAN, S.; NGUYEN, A.; MUTHUKKUMARASAMY, V. Performance analysis of GAME: A generic automated marking environment, ScienceDirect, Computers & Education 50, 2006

BOADA, I.; SOLER, J.; PRADOS, F.; POCH, J. A teaching learning support tool for introductory programming courses Dep. Computer Science and Applied Mathematics. University of Girona Campus de Montilivi, 17071, Girona (Spain), 2004

CAI, R. Adaptive Learning Practice for Online Learning and Assessment, ICDEL '18 Proceedings of the 2018 International Conference on Distance Education and Learning Pages 103-108

GONZÁLEZ-SACRISTÁN, C.; MOLINS-RUANO, P.; DÍEZ, F.; RODRÍGUEZ, P.; SACHA, G. Computer-assisted assessment with item classification for programming skills, TEEM '13 Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality, Pages 111-117

HARTMANN, B., MACDOUGALL, D., BRANDT, J., & KLEMMER, S. R. What Would Other Programmers Do? Suggesting Solutions to Error Messages. *CHI 2010: Understanding and Supporting Programming*, pp. 1019-1028, Abril de 2010

HUNTER, G.; LIVINGSTONE, D.; NEVE, P.; ALSOP, G. LearnProgramming++: The Design, Implementation and Deployment of an Intelligent Environment for the Teaching and Learning of Computer Programming, 2013, 9th International Conference on Intelligent Environments

JENKINS, T. On the difficulty of learning to program, 2002, School of Computing University of Leeds

JONES, M. T. A beginner's guide to artificial intelligence, machine learning, and cognitive computing, June 2017, IBM.
Disponível em: https://developer.ibm.com/articles/cc-beginner-guide-machine-learning-ai-cognitive. Acessado em: 30 de novembro de 2018

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE 2007-001, Department of Computer Science Keele University, Keele, 2007.

KOHAVI, R.; PROVOST, F. Glossary of Terms, Machine Learning - Special issue on applications of machine learning and the knowledge discovery process archive, Volume 30 Issue 2-3, Volume 30, pp. 271-274, Fevereiro/Março de 1998

KRUSCHE, S.; SEITZ, A. ArTEMiS - An Automatic Assessment Management System for Interactive Learning. In SIGCSE'18: The 49th ACM Technical Symposium on Computing Science Education, February, 21–24, 2018, Baltimore, MD, USA. ACM, New York, NY, USA, 6 pages.

KURVINEN, E.; HELLGREN, N.; KAILA, E.; LAAKSO, M.J.; SALAKOSKI, T. Programming Misconceptions in an Introductory Level Programming Course Exam, ITiCSE '16, July 09-13, 2016, Arequipa, Peru.

LUXTON-REILLY, A.; MCDERMOtt, R.; PETERSEN, A.; BECKER, B. A.; SANDERS, K.; MIROLO, C.; CAO, Y.; MÜHLING, A.; SIMON; WHALLEY, J. Developing Assessments to Determine Mastery of Programming Fundamentals - ITiCSE '17, July 3-5, 2017, Bologna, Italy

MENDOZA, B.; REYES-ALAMO, J.; WU, H.; CARRANZA, A.; ZAVALA, L. iPractice: a self-assessment tool for students learning computer programming in an urban campus, Journal of Computing Sciences in Colleges, 2016

MICHALSKI, R.S.; BRATKO I.; KUBAT M. Machine Learning, Data Mining and Knowledge Discovery: Methods and Applications, John Wiley & Sons, 1998

MICHIE, D.; SPIEGELHALTER, D.J., TAYLOR, C.C. Machine learning, Neural and Statistical Classification, Ellis Horwood, 1994.

MITCHELL, T. M. Machine Learning - McGraw-Hill Science/Engineering/Math; (March 1, 1997)

MOREIRA, R. T. Um perfil de capacidade para a melhoria do processo em micro e pequenas organizações orientadas à manutenção e evolução de produtos de software, Universidade Federal de Pernambuco, 13-Mar-2015, disponível em: https://repositorio.ufpe.br/handle/123456789/14956

MOTA, M. S. G.; PEREIRA, F. E. L. Desenvolvimento e Aprendizagem - Processo de Construção do Conhecimento e Desenvolvimento Mental do Indivíduo, Abril de 2013

SAMUEL, A. L. Some Studies in Machine Learning Using the Game of Checkers, IBM Journal of Research and Development, Volume: 3, Issue: 3, Julho de 1959.

SHAVLIK, J.W.; DIETTERICH, T.G. Readings in machine learning, Morgan Kaufmann, 1990.

TAHERKHANI, A.; KORHONEN, A.; MALMI, L. Automatic Recognition of Students' Sorting Algorithm Implementations in a Data Structures and Algorithms Course, 2012

TANG, T.; SMITH, R.; WARREN, J.; RIXNER, S. Data-Driven Test Case Generation for Automated Programming Assessment ITiCSE '16, July 09-13, 2016, Arequipa, Peru

ZAVALA, L.; BENITO M. On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises, SIGCSE'18: The 49th ACM Technical Symposium on Computing Science Education, February 21–24, 2018

APÊNDICE

APÊNDICE 1 - INFORMAÇÕES GERAIS SOBRE ARTIGOS SELECIONADOS

ID	Título	Base de Dados	Autoria	Ano	Fonte	Tipo
E01	ArTEMiS - An Automatic Assessment Management System for Interactive Learning	ACM Digital Library	Stephan Krusche Andreas Seitz	2018	SIGCSE'18: The 49th ACM Technical Symposium on Computing Science Education, February 21–24, 2018	Conferência
E02	Automatic Recognition of Students' Sorting Algorithm Implementations in a Data Structures and Algorithms Course	ACM Digital Library	Ahmad Taherkhani Ari Korhonen Lauri Malmi	2012	Koli Calling '12, November 15–18, Tahko, Finland	Conferência
E03	On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises	ACM Digital Library	Laura Zavala Benito Mendoza	2018	SIGCSE'18: The 49th ACM Technical Symposium on Computing Science Education, February 21–24, 2018	Conferência
E04	Programming Misconceptions in an Introductory Level Programming Course Exam	ACM Digital Library	Einari Kurvinen, Niko Hellgren, Erkki Kaila, Mikko-Jussi Laakso, Tapio Salakoski	2016	ITiCSE '16, July 09-13, 2016, Arequipa, Peru	Conferência
E05	Using learning analytics to assess students' behavior in open-ended programming tasks	ACM Digital Library	Paulo Blikstein	2011	LAK'11, February 27–March 1, 2011, Banff, Alberta, Canada.	Conferência
E06	iPractice: a self-assessment tool for students learning computer programming in an urban campus	ACM Digital Library	Benito Mendoza, José Reyes-Alamo Huixin Wu, Aparicio Carranza, Laura Zavala	2016	Journal of Computing Sciences in Colleges	Periódico
E07	Developing Assessments to Determine Mastery of Programming Fundamentals	ACM Digital Library	Andrew Luxton-Reilly, Roger McDermo, Andrew Petersen,	2017	ITiCSE '17, July 3-5, 2017, Bologna, Italy	Conferência

			Brett A. Becker, Kate Sanders, Yingjun Cao, Andreas Muhling, Simon, Jacqueline Whalley			
E08	Data-Driven Test Case Generation for Automated Programming Assessment	ACM Digital Library	Terry Tang, Rebecca Smith, Joe Warren, Scott Rixner	2016	ITiCSE ’16, July 09-13, 2016, Arequipa, Peru	Conferência
E09	Experiences in Introducing Blended Learning in an Introductory Programming Course	ACM Digital Library	Ella Albrecht, Fabian Gumz, Jens Grabowski	2018	ECSEE’18, June 14–15, 2018, Seeon/ Bavaria, Germany	Conferência
E10	Teaching How to Program using Automated Assessment and Functional Glossy Games	ACM Digital Library	José Bacelar Almeida, Alcino Cunha, Nuno Macedo, Hugo Pacheco, José Proença	2018	Proceedings of the ACM on Programming Languages archive Volume 2 Issue ICFP, September 2018 Article No. 82	Periódico
E11	Adaptive Learning Practice for Online Learning and Assessment	ACM Digital Library	Riachard Cai	2018	ICDEL '18 Proceedings of the 2018 International Conference on Distance Education and Learning Pages 103-108	Conferência
E12	Computer-assisted assessment with item classification for programming skills	ACM Digital Library	C. González-Sacristán, P. Molins-Ruano, F. Díez, P. Rodríguez, G. M. Sacha	2013	TEEM '13 Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality Pages 111-117	Conferência
E13	Bayesian Networks in Intelligent Tutoring Systems as an Assessment of Student	ACM Digital Library	Roselie B. Alday	2018	ICACS '18, July 27–29, 2018, Beijing, China, 2018	Conferência

	Performance using Student Modeling					
E14	A teaching learning support tool for introductory programming courses	IEEE Xplore	Imma Boada, Josep Soler, Ferran Prados, Jordi Poch	2004	Dep. Computer Science and Applied Mathematics. University of Girona Campus de Montilivi	Periódico
E15	Performance analysis of GAME: A generic automated marking environment	IEEE Xplore	Michael Blumenstein, Steve Green, Shoshana Fogelman, Ann Nguyen, Vallipuram Muthxukkumarasamy	2006	ScienceDirect, Computers & Education 50, 2006	Periódico
E16	LearnProgramming++: The Design, Implementation and Deployment of an Intelligent Environment for the Teaching and Learning of Computer Programming	IEEE Xplore	Gordon Hunter, David Livingstone, Paul Neve, Graham Alsop	2013	9th International Conference on Intelligent Environments	Conferência
E17	Design and Evaluation of Automated Scoring Java Programming Assignments	IEEE Xplore	Yuki Akahane, Hiroki Kitaya Ushio Inoue	2015	SNPD 2015, June 1-3 2015, Takamatsu, Japan	Periódico

ID: Identificação do estudo.

APÊNDICE 2 - AVALIAÇÃO POR CRITÉRIO DE QUALIDADE SOBRE OS ARTIGOS SELECIONADOS

ID	CQ1	CQ2	CQ3	CQ4	CQ5	CQ6	CQ7	CQ8	CQ9	CQ10	Total
E16	1	1	1	1	0,5	1	1	1	1	1	9,5
E09	1	0,5	1	1	1	1	1	1	0,5	1	9,0
E10	1	0,5	1	1	1	1	1	1	0,5	1	9,0
E17	1	0,5	1	1	1	1	0,5	1	1	1	9,0
E06	1	0	1	0,5	1	1	1	1	1	1	8,5
E13	1	1	0	1	1	1	1	1	0,5	1	8,5
E01	1	0	0,5	0,5	1	1	1	1	1	1	8,0
E05	0,5	0,5	0,5	0,5	1	1	1	1	1	1	8,0
E07	1	0	1	0	1	1	1	1	1	1	8,0
E14	1	0	1	1	1	0,5	0,5	1	1	1	8,0
E04	1	0	1	1	0,5	0,5	1	1	0,5	1	7,5
E02	0,5	1	0	0,5	1	1	1	0,5	0,5	1	7,0
E03	0,5	0	1	0,5	0,5	1	0,5	1	1	1	7,0
E08	0	0,5	1	0,5	0,5	1	1	0,5	1	1	7,0
E11	1	0	0	1	0,5	0,5	0,5	1	1	1	6,5
E12	0,5	0,5	0	0,5	1	0,5	1	1	0,5	1	6,5
E15	0	0	1	1	0,5	1	1	0,5	0,5	1	6,5

ID: Identificação do estudo.