Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Curso de Bacharelado em Ciência da Computação– BCC

Nathaelly Boni

Inteligência Artificial para Testes de Software

São Carlos
2024

Nathaelly Boni

Inteligência Artificial para Testes de Software

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Orientador: Fabiano Cutigi Ferrari

São Carlos
2024

Agradecimentos

Ao meu companheiro, Mateus, por todo o cuidado, amor e conhecimento que recebo diariamente, e que me tornam uma pessoa sempre melhor.

À minha mãe, Ginesia, e aos meus irmãos, Nathalia e Nathael, por serem família que se faz presente mesmo de longe, e me darem todo o apoio, ajuda e amor em todos os momentos da minha vida.

Aos meus amigos Gabriel e Guilherme, pelas risadas, conselhos, e amizade que construímos durante a faculdade, fundamental para que eu chegasse ao fim dela.

A todos os tantos outros amigos dos cursos de computação e também da química, que de alguma forma contribuíram para essa jornada.

Ao professor Fabiano Cutigi Ferrari, por toda a ajuda necessária para a execução deste trabalho e o fim da minha graduação.

"Trabalho duro é inútil para aqueles que não acreditam em si mesmos."
(Naruto Uzumaki)

Resumo

O processo de desenvolvimento de software é essencial para as aplicações usadas no dia-a-dia. Parte desse processo envolve a etapa de testes de software, responsável por revelar a presença de defeitos e melhorar a eficiência da aplicação para o uso. Essa etapa pode ser muito custosa e trabalhosa, e portanto dispõe de várias maneiras para otimizar o tempo. A inteligência artificial (IA) pode ser utilizada de forma conjunta a essa etapa, trazendo vários benefícios e melhorias tanto de orçamento quanto da otimização do próprio software. Este trabalho abordará a intersecção entre essas duas áreas, tratando de revisar alguns artigos selecionados que experimentam maneiras diferentes de aplicar IA para testes de software. Os resultados indicam que a área é muito promissora e possui grandes avanços, mas ainda tem um grande campo a ser explorado com maiores detalhes. Os resultados explorados demonstram que a inteligência artificial pode ajudar, cada vez mais, a desenvolver softwares de boa qualidade.

Palavras-chave: Inteligência artificial, Testes de Software, Engenharia de Software.

Lista de ilustrações

Sumário

Capítulo 1

Introdução

Neste capítulo apresentam-se o contexto e a motivação para o trabalho realizado (Seção 1.1), os objetivos pretendidos (Seção 1.2), a metodologia seguida para a realização do trabalho (Seção 1.3) e uma breve descrição da organização do restante do trabalho (Seção 1.4).

1.1 Contexto e Motivação

Um software de alta qualidade é essencial para poder atender às necessidades dos usuários e garantir a eficácia do produto. Portanto, testes de software desempenham um importante papel no ciclo de desenvolvimento do produto, pois ajudam a revelar a presença de defeitos e melhoram a estabilidade e a eficiência do software (HOURANI; HAMMAD; LAFI, 2019). Porém, com o passar do tempo, a complexidade das aplicações de software aumentou exponencialmente, e testá-los representa um grande valor no orçamento do projeto. Recentemente, a Inteligência Artificial (IA) passou a ser vista como uma importante ferramenta a ser integrada às etapas de desenvolvimento de software, pois consegue desempenhar importantes evoluções na garantia de qualidade e eficiência na entrega do produto (DURELLI et al., 2019). Dentre essas etapas, a de teste de software é muito importante para expor erros de execução e revelar a presença de defeitos no código e, consequentemente, em funcionalidades pedidas nos requisitos que não estão operando da maneira correta. Usar IA para essa etapa de teste é vista como uma abordagem inovadora, podendo revolucionar o ciclo de desenvolvimento ao melhorar a confiabilidade e desempenho dos produtos (FELDT; NETO; TORKAR, 2018).

1.2 Objetivos

Os objetivos deste trabalho consistem em analisar os avanços e discutir tendências entre as áreas de IA e de teste de software, revisando os métodos e vantagens descritos em literatura, podendo analisar de que forma a IA está avançando no contexto de desenvolvimento de software, especificamente na área de testes.

1.3 Metodologia

Para atingir o objetivo definido, inicialmente foi evidenciada a relevância do tema por meio de debates e pesquisas não sistematizadas em bases indexadas de estudos científicos da área de Ciência da Computação. Em seguida, pesquisou-se nas bases de literatura *ACM Digital Library* e *IEEExplore* palavras chaves como "*artificial intelligence*" e "*software testing*" de forma conjunta. Além disso, juntou-se a essas palavras chaves o termo chave "*systematic review*" para encontrar trabalhos mais generalizados. Após isso, filtraram-se os artigos de maior relevância para o tema, além de uma busca nas referências das revisões sistemáticas encontradas para encontrar quais materiais foram debatidos (ROBLES-AGUILAR et al., 2021). Por fim, material para o trabalho também foi pesquisado em "literatura cinzenta" (por exemplo, blogs/fóruns de tecnologia, newsletter, entre outros) principalmente para encontrar como o tema é aplicado de forma prática, envolvendo aplicações de software e tecnologias atuais (GAROUSI; FELDERER; MäNTYLä, 2019).

1.4 Organização

O restante do trabalho está organizado da seguinte forma:

O *Capítulo 2: Fundamentação teórica* abordará as bases teóricas de IA que são fundamentais para a construção da argumentação de como a IA aplicada a testes pode representar um avanço no tema. Para isso, serão discutidos alguns princípios necessários para o tema, como Aprendizado de Máquina (AM), Redes Neurais, Processamento de Linguagem Natural (PLN), entre outros. Além disso, neste capítulo também serão abordados os conceitos de teste de software, incluindo os princípios, as fases e as técnicas utilizadas.

No *Capítulo 3: Planejamento de estudo*, serão melhor detalhadas as etapas para o planejamento e desenvolvimento do trabalho, incluindo a metodologia e os procedimentos para encontrar e filtrar as informações utilizadas para a construção do trabalho. Além disso, também será explicitado qual foi o método para encontrar e analisar quais os materiais relevantes para a construção desse trabalho.

O *Capítulo 4: Resultados*, Análise e Discussão demonstrará quais os avanços da área até o momento, seja de forma prática e/ou teórica, e as discussões sobre o que pode ser feito no futuro.

O *Capítulo 5: Conclusão* vai sumarizar o que podemos concluir a partir do material levantado, e o que é possível esperar do futuro na área.

Capítulo 2

Fundamentação Teórica

Para a produção deste trabalho, serão abordados os conceitos importantes para o entendimento do tópico, tanto sobre testes de software quanto sobre as bases teóricas de IA, assim sendo possível argumentar como IA pode ser aplicada a testes (AMALFITANO et al., 2023).

2.1 Princípios de Testes de Software

O teste de software, em suas várias formas de ser aplicado, representa uma fase essencial no ciclo de vida de um software, podendo contribuir na garantia de qualidade e de confiabilidade do produto. Para isso, existem várias formas de se testar um software, com focos em diferentes perspectivas do software, para auxiliar nessa etapa do desenvolvimento. A escolha da forma de se testar depende dos requisitos e do contexto, e pode ser efetuada de forma manual e/ou automatizada (a partir de frameworks e simulações de ambientes), além de auxiliada pela IA, como é possível ver no decorrer deste documento (TRUDOVA; DOLEZEL; BUCHALCEVOVA, 2020).

A seguir são sucintamente descritas duas perspectivas relacionadas ao teste de software. A primeira, referente às fases de teste, aborda a granularidade do teste em função dos elementos que compõem o software sob teste. A segunda, referente às técnicas de teste, aborda a representação do software que serve como base para se derivarem os requisitos de teste.

2.1.1 Fases de Teste

- Teste de unidade: Focados em unidades individuais do código, o teste de unidade (também conhecido popularmente como testes unitários) é executado durante o desenvolvimento e verifica se cada parte do software (uma função, por exemplo) funciona conforme o esperado.

- Teste de Integração: O teste de integração verifica se essas diferentes unidades de software trabalham de forma correta em conjunto e se há algum problema de interação entre as classes ou os módulos.

- Teste de Sistema: Ao final do desenvolvimento do software e antes de sua implantação, é possível realizar o teste de sistema, com o software sendo testado em seu ambiente real de execução (incluindo a infraestrutura tecnológica e as pessoas que vão utilizar a aplicação), a fim de verificar se os requisitos pedidos foram cumpridos.

2.1.2 Técnicas de Teste

- Teste funcional (caixa preta): Nos testes funcionais, ou caixa preta, o testador verifica as funcionalidades do software baseado nos requisitos e comportamentos esperados, sem ter conhecimento de sua estrutura interna, ou seja: o objetivo é verificar se o sistema funciona conforme se espera, sem precisar entender todos os detalhes de sua implementação.

- Teste estrutural (caixa branca): Já nos testes estruturais, ou caixa branca, eles são conduzidos com esses conhecimentos de sua implementação e detalhes da estrutura interna e do código fonte, examinando o código do software e garantindo que ele esteja funcionando nos quesitos de lógica. Ambos os métodos são complementares.

É importante dizer que os testes não são efetuados apenas uma vez de forma isolada, e sim ao longo do ciclo do desenvolvimento, de forma contínua e integrada. Uma estratégia para isso é a de testes de regressão, que repetem os testes listados acima após alterações no código, para poder garantir que as mudanças não quebraram (isto é, pararam de funcionar parcial ou completamente) funcionalidades já existentes. Depois do software ser implementado, por exemplo, usuários podem perceber problemas ou então possíveis sugestões, então novos testes, de diferentes tipos, como testes de usabilidade ou de segurança, além dos testes já existentes, são efetuados a fim de garantir a qualidade da entrega do software. Além disso, testes possuem métricas para avaliar se foram executados da maneira que se esperava, que podem incluir número de defeitos encontrados e corrigidos, tempo médio de detecção e taxas de cobertura.

2.2 Princípios de IA

Nesta segunda parte deste capítulo descrevem-se as bases teóricas de IA que podem auxiliar na temática de teste de software. O conceito de inteligência artificial não é recente; o mesmo data de décadas atrás, desde então impactando uma grande variedade de áreas. Alguns dos princípios mais relevantes para a construção deste trabalho estão descritos abaixo (BATARSEH et al., 2020):

- Redes Neurais: modelos inspirados no cérebro, compostos por unidades conectadas entre si, trabalhando em conjunto para resolver problemas e capazes de reconhecer padrões.

- Aprendizado de Máquina: uma subárea que tenta desenvolver algoritmos e modelos que permitem que os sistemas aprendam com os dados que possuem (HALL; BOWES, 2012).

- Processamento de linguagem natural (PLN): essa subárea se concentra em desenvolver sistemas que compreendem e geram linguagem humana natural.

- Algoritmos genéticos: esta abordagem utiliza conceitos inspirados na evolução para buscar otimizar as soluções de problemas. Utilizando-se de conceitos como seleção, cruzamento e mutação, criam-se novas gerações de indivíduos, evoluindo populações para retornar o indivíduo mais apto para solucionar o problema proposto (RODRIGUES et al., 2018).

Incorporar esses fundamentos nas etapas de desenvolvimento de software possibilita testes mais confiáveis e melhores, podendo inclusive automatizar algumas etapas, permitindo que os testes sejam mais inteligentes na maneira que otimizam a detecção de bugs e se adaptam a diferentes situações na criação de sistemas.

Capítulo 3

Planejamento do Estudo

3.1 Objetivos

Este trabalho se explora os avanços e o futuro na intersecção entre as áreas de IA e de testes de software, revisando os métodos e vantagens descritos em literatura, podendo analisar de que forma a IA está avançando nesse contexto, especificamente na área de testes.

3.2 Metodologia

Para a construção desse trabalho, vários artigos foram selecionados para poderem ser debatidos. Esses artigos foram encontrados nas bases de literatura *ACM Digital Library* e *IEEE Xplore* a partir das palavras chave "*artificial intelligence*" e "*software testing*", e mais tarde também "*systematic review*" junto a eles. Com esses termos em mãos, 1,747 resultados foram obtidos da *ACM Digital Library* e 581 na *IEEE Xplore*, englobando várias facetas de teste de software e de inteligência artificial. Os critérios utilizados para selecionar quais seriam os artigos relevantes para fazer este trabalho incluem:

- número de citações;
- publicação em uma revista de relevância;
- tecnologia de IA utilizada;
- assuntos que não tangenciem ao tema principal que a busca foi planejada;
- experimentos bem detalhados.

Além disso, para a busca de itens de literatura cinzenta, foi feita uma busca no Google a partir dos termos “artificial intelligence in software testing” e “inteligência artificial em testes de software”. Os critérios para selecionar os itens comentados foram:

- falarem sobre temáticas que não haviam sido citadas nos artigos de literatura;

- serem do último ano (2023);

- debaterem sobre a aplicabilidade do tema, e não sobre ciência de base, já que esta já foi discutida nos artigos de literatura;

- escrita bem feita e não de forma sensacionalista, a fim de tratar o tema de maneira fatalista.

De forma conjunta às buscas feitas por itens de literatura cinzenta, também foram pesquisadas tecnologias atuais utilizadas na área. A seleção dessas tecnologias foi feita inicialmente usando de experiências prévias, e a partir dos softwares já conhecidos, uma nova busca foi feita para encontrar tecnologias similares.

3.3 Procedimentos

Para extrair as informações desejadas dos artigos, todos foram lidos e organizados por método(s) de IA utilizado(s). A organização se deu por meio de uma tabela que será apresentada no próximo capítulo.

Capítulo 4

Resultados, Análise e Discussão

Neste capítulo, serão expostos os resultados encontrados nas buscas feitas para este trabalho. O capítulo está dividido em duas partes: uma seção para descrever os resultados oriundos da análise de artigos encontrados nas bases indexadas (Seção 4.1), uma seção para descrever os resultados derivados de itens de literatura cinzenta (Seção 4.2), e uma seção para abordar as tecnologias mais utilizadas na atualidade (Seção 4.3).

4.1 Artigos Extraídos da Bases Indexadas

Nesta seção, os artigos selecionados para a construção desse trabalho são sintetizados para, mais adiante, embasarem a conclusão a respeito do progresso da área de intersecção entre Inteligência Artificial e Teste de Software e qual um possível futuro do campo. Nas Figura 1 e 2, respectivamente, foram incluídos dados estatísticos sobre países e os anos mais frequentes nas publicações. As informações foram extraídas da base de dados *Web of Science*, que engloba as bases que os artigos foram encontrados (*ACM Digital Library* e *IEEE Xplore*) e fornece essas informações estatísticas, além de gráficos.

Na figura 1, mostra-se os 15 países com mais publicações segundo os termos pesquisados e a quantidade de artigos para cada um. Com essas informações, é possível perceber que a China é, com grande margem, o país que mais produz literatura sobre o tema, seguida pelos Estados Unidos.

Na figura 2, é mostrado os quinze anos com mais publicações acerca do tema, sendo 2016 o com mais artigos e 2006 com menos. É possível ver que apesar do número ter crescido muito de 2015 para 2016, ele cai substancialmente depois disso.

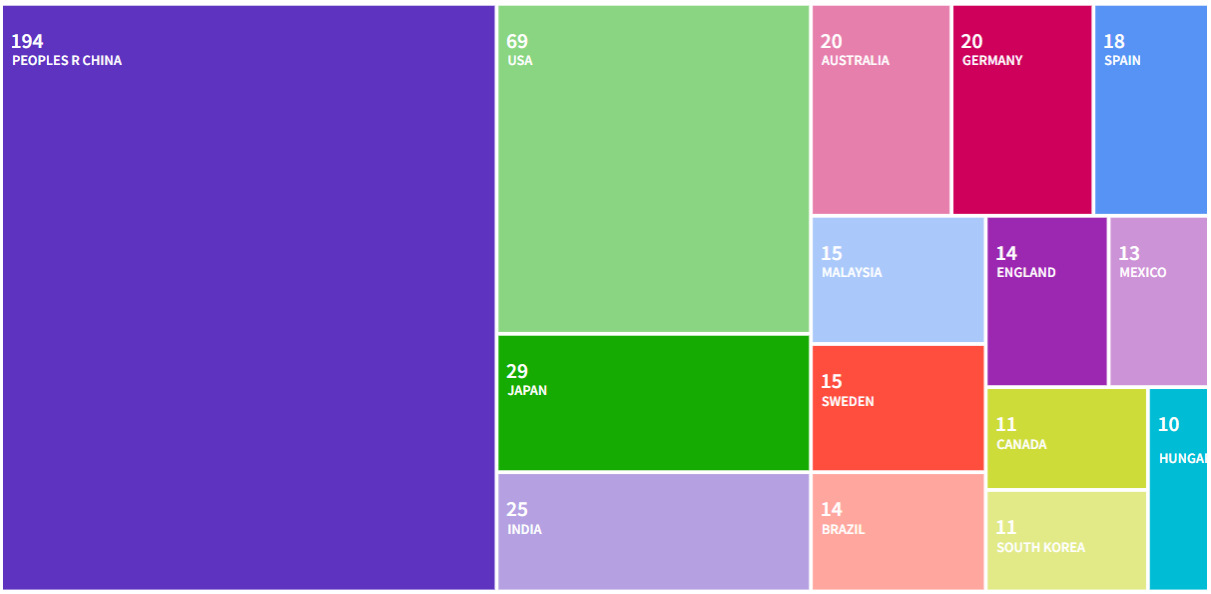

Figura 1 – Países mais frequentes nas publicações encontradas

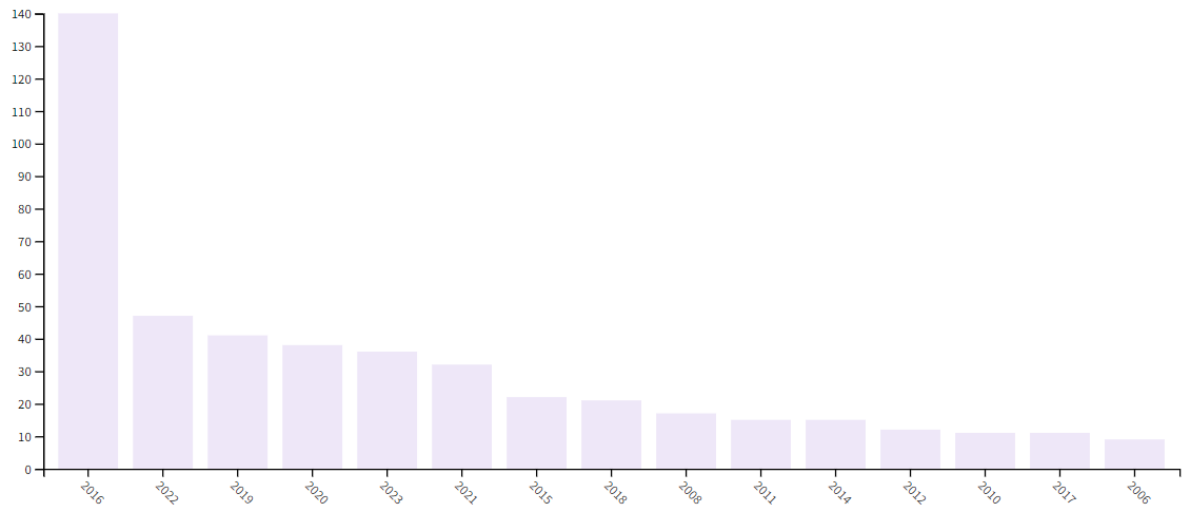

Figura 2 – Anos mais frequentes nas publicações encontradas

Na Figura 3, é demonstrado um gráfico que contém o percentual das tecnologias de IA mais encontradas nos artigos que foram selecionados para serem analisados na construção deste trabalho:

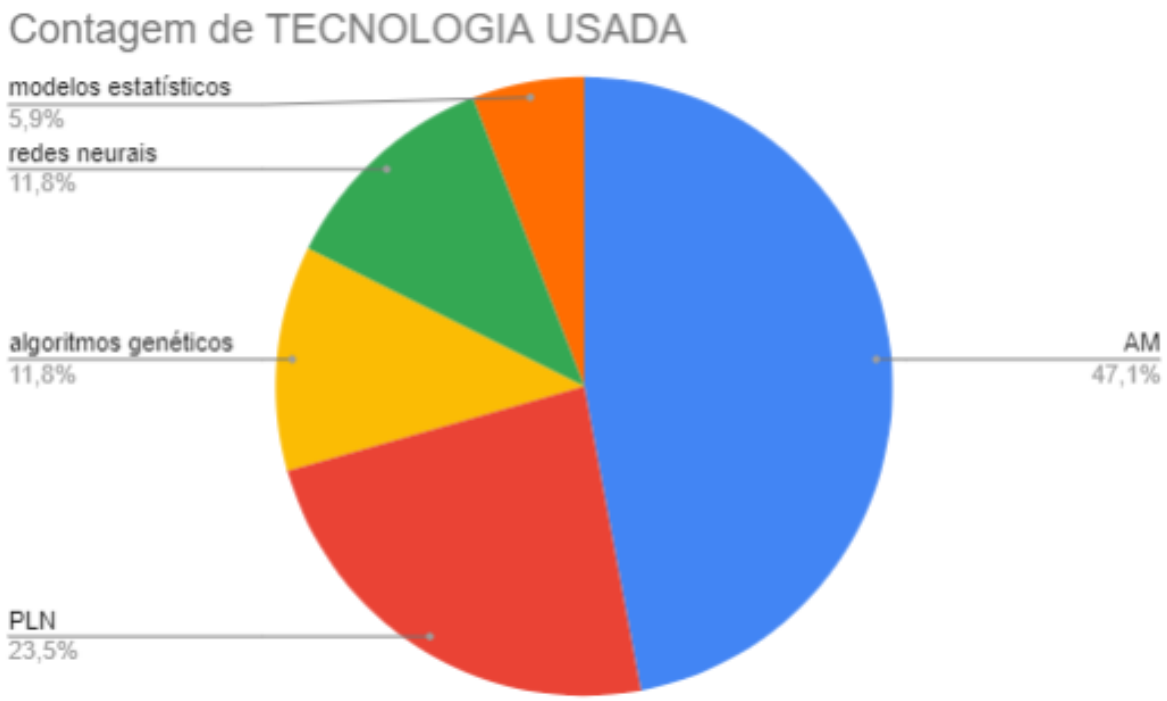

Figura 3 – Gráfico com o percentual das temáticas dos artigos discutidos neste trabalho

A figura 3 demonstra, dentre os dezessete artigos debatidos neste trabalho, o percentual de cada tecnologia específica dentro do campo de inteligência artificial que foi abordada. Ou seja, de dezessete artigos:

- 8 abordam aprendizado de máquina;
- 4 abordam PLN;
- 2 abordam algoritmos genéticos;
- 2 abordam redes neurais;
- 1 aborda modelos estatísticos.

Por fim, uma tabela com os artigos selecionados de acordo com os objetivos e a metologia definidas para o trabalho, organizados por nome do artigo, autor, e a temática de IA utilizada:

ID	NOME	AUTOR	TEMÁTICA DE IA	ANO
A01	A Learning-Based Method for Combining Testing Techniques	COTRONEO; PIETRANTUONO; RUSSO	AM	2013
A02	A Machine Learning Approach to Generate Test Oracles	BRAGA et al.	AM	2018

ID	NOME	AUTOR	TEMÁTICA DE IA	ANO
A03	An Exploration of Statistical Models for Automated Test Case Generation	SANT; SOUTER; GREENWALD	modelos estatísticos	2005
A04	Artificial Neural Network for Automatic Test Oracles Generation	HU et al.	redes neurais	2008
A05	Automatic software test case generation: An analytical classification framework	KEYVANPOUR; HOMAYOUNI; SHIRAZEE	AM	2012
A06	Automatic Generation of Test Case based on GATS Algorithm*	SHEN et al.	algoritmos genéticos	2009
A07	Automatically Documenting Unit Test Cases	LI et al.	PLN	2016
A08	Automatic Testing of GUI-Based Applications	MARIANI et al.	AM	2014
A09	Constructing Test Cases Using Natural Language Processing	ANSARI et al.	PLN	2017
A10	How High Will It Be? Using Machine Learning Models to Predict Branch Coverage in Automated Testing	GRANO et al.	AM	2018
A11	Generation of Test Oracles using Neural Network and Decision Tree Model	VINEETA; SINGHAL; BANSAL	redes neurais	2014
A12	Generation of Test Cases from Software Requirements Using Natural Language Processing	VERMA; BEG	PLN	2013
A13	Investigating the Accuracy of Test Code Size Prediction using Use Case Metrics and Machine Learning Algorithms: An Empirical Study	BADRI et al.	AM	2017

ID	NOME	AUTOR	TEMÁTICA DE IA	ANO
A14	Identifying Effective Test Cases Through K-means Clustering for Enhancing Regression Testing	PANG; XUE; NAMIN	AM	2013
A15	Investigating NLP-based Approaches for Predicting Manual Test Case Failure	HEMMATI; SHARIFI	PLN	2018
A16	Linking software testing results with a machine learning approach	LENZ; POZO; VERGILIO	AM	2013
A17	Machine Learning in Value-Based Software Test Data Generation	ZHANG	algoritmos genéticos	2006

A seguir, os artigos descritos estão organizados cronologicamente por técnica de IA explorada.

Em Keyvanpour et al, de 2012 (KEYVANPOUR; HOMAYOUNI; SHIRAZEE, 2012), os autores mostram dois frameworks para classificar abordagens diferentes de testes automatizados de software a partir do tipo deles e de qual algoritmo cada um usa, além de fazer uma comparação entre eles. Os resultados mostram intersecções entre diferentes métodos de testes e suas formas automatizadas, mostrando comparações sobre em qual caso cada uma se encaixa melhor, dependendo dos requisitos.

O artigo de Cotroneo e seus colaboradores, de 2013 (COTRONEO; PIETRANTUONO; RUSSO, 2013), apresenta uma maneira usando-se de aprendizado de máquina e inferência bayesiana para combinar métodos de testagem de forma a eles aprenderem com seus erros e se adaptarem para o teste atual que estão processando. Seus resultados apresentam que, por meio de algoritmos de AM, e integrando esses resultados aos testes, a cobertura e a efetividade destes são melhores do que sem o uso dessa técnica. É dito que que os custos para isso funcionar envolvem duas fases: a offline, que extrai as métricas do software que está sendo testado e a seleção do melhor classificador para esses testes; e uma fase online, que inicia os casos de teste para cada critério determinado a partir do modelo aprendido na fase offline, além das predições. Também assume-se que todo o processo pode ser automatizado. A conclusão chegada é de que esse método de combinar técnicas é mais eficiente do que o emprego de apenas um método, especialmente depois da fase inicial de testes (já que o modelo aprende com seus próprios erros).

No trabalho de Lenz et al, de 2013 (LENZ; POZO; VERGILIO, 2013), os autores propõem uma abordagem baseada em AM e clustering para relacionar as informações obtidas a partir dos testes estruturais e dos testes baseados em falhas com os aspectos

funcionais que o software possui. Essas informações são cruzadas para poder verificar a cobertura dos testes e se eles foram suficientes, e também servem para treinar modelos que podem, eventualmente, ajudar na etapa de testes.

O trabalho de Pang et al, de 2013 (PANG; XUE; NAMIN, 2013) apresenta uma proposta usando de agrupamento de k-means para classificar quais testes são efetivos para o software em questão, isto é, medir a Distância de Hamming entre as coberturas dos testes da versão atual do software sendo testado e de suas versões anteriores. No agrupamento, a ideia é separar em dois grupos os dados, e em cada grupo os objetos são mais similares a outros do mesmo grupo do que do outro grupo. Após aplicar o algoritmo proposto aos experimentos que os autores fizeram, o resultado é que esses agrupamentos funcionaram e conseguiram uma alta acurácia.

O trabalho de 2014 de Mariani et al (MARIANI et al., 2014) descreve uma técnica para gerar automaticamente testes para a interface de usuário do software, usando muito de Q-Learning para conseguir interagir com a aplicação e simular suas funcionalidades. Testes em interfaces gráficas são geralmente manuais, e ainda não são tão comumente aplicados usando-se de técnicas de IA. O trabalho usa de testes de caixa preta em cima de protótipos das telas do software para simular ações de usuários e suas respectivas respostas esperadas, demonstrando bons resultados na geração de testes e detecção de falhas que passam despercebidas pelos desenvolvedores.

Em Badri com colaboradores, de 2017 (BADRI et al., 2017), o trabalho ressalta a importância de testes de software e também como é uma etapa cara pois envolve muitas etapas para ser feita de maneira efetiva. Para isso, propõe usos de diferentes algoritmos de AM (como k-NN, regressão linear, Perceptron Multicamadas, Random Forest, Naive Bayes e C4.5) para montar modelos de predição a fim de conseguir prever a quantidade de Test Lines Of Code (TLOC), isto é, o tamanho do software. O experimento foi aplicado em trabalhos escritos em Java e demonstrou que as predições foram mais acuradas do que os pontos de caso de uso, técnica já conhecida para prever o tamanho do software.

Mais tarde, em R. Braga et al, de 2018 (BRAGA et al., 2018), o trabalho descreve como um dos grandes desafios de testagem de softwares é a criação da técnica do oráculo, e sua técnica apresentada é o uso de AM para automatizá-lo. A abordagem é baseada em capturar as ações performadas no software pelos seus usuários e pelo oráculo sendo testado. Assim, a ideia é comparar as informações de ambos para ver se há alguma inconsistência em como a aplicação deveria funcionar. Os resultados dos experimentos descritos foram satisfatórios e indicaram que esse é um método que pode ser bastante efetivo para a resolução dos problemas apresentados.

Em 2018, o trabalho de Grano e colaboradores (GRANO et al., 2018) explora modelos de AM que consigam prever a cobertura da geração de testes. Para isso, usaram modelos como Regressão de Huber, Perceptron multicamadas e Support Vector Regression. Por fim, os autores concluem que há viabilidade na utilização desses algoritmos para facilitar

o conhecimento de até onde vai a cobertura dos testes designados a um software antes de serem executados, mas o campo da pesquisa ainda é muito aberto para exploração.

Em Verma et al, de 2013 (VERMA; BEG, 2013), os autores ressaltam o fato de que, apesar de a linguagem natural ser mais compreensível, usá-la para escrever o documento de requisitos pode transformar alguns deles em ambíguos ou incompletos. Assim, se propõem a gerar os casos de teste derivados dos requisitos do software usando de PLN para processar o que está escrito. Os requisitos são pré -processados, analisados e por fim combinados para conseguir gerar os testes necessários para cobrir o documento de requisitos. Por fim, concluem que o modelo executado pode ser muito útil para poder estudar o documento de requisitos e analisar se está completo para, enfim, poder gerar os testes.

O trabalho de 2016 de Li et al (LI et al., 2016) ressalta a importância de manter o histórico de testes aplicados a um software bem documentado para diminuir problemas e permitir aos desenvolvedores uma facilidade maior de encontrar quais foram os testes utilizados em funções específicas. O artigo também apresenta uma pesquisa feita com desenvolvedores para demonstrar que essa documentação é muito valorizada, mas raramente é feita. A abordagem utiliza de processamento de linguagem natural, análise estática, backward slicing e code summarization para gerar descrições dos testes unitários e assim documentar os métodos focais usados e as dependências deles.

Em Ansari et al, de 2017 (ANSARI et al., 2017), os autores propõem uma geração automatizada de testes utilizando-se de PLN, capturando palavras chave dos requisitos do software em questão, a fim de diminuir o tempo e o custo da execução desses testes. O artigo conclui que apesar de conseguirem otimizar o tempo para esses testes, ainda há muito a se fazer para aumentar a acurácia do sistema.

No trabalho de Hemmati et al, de 2018 (HEMMATI; SHARIFI, 2018), os autores propõem uma maneira de prever se o caso de teste falhará usando de PLN a partir das descrições dos casos, demonstrando que esse modelo de PLN consegue melhorar as predições. A metodologia desse modelo é extrair palavras chaves (principalmente substantivos) dos scripts de testes (escritos em linguagem natural) e aplicando pesos às palavras que mais aparecem, usando de TF-IDF, ou seja, o produto da Frequência do Termo e da Frequência Inversa do Documento e, por fim, usando modelos de predição para conferir se os casos de teste funcionarão ou não, demonstrando resultados promissores.

O trabalho de Zhang, de 2006 (ZHANG, 2006) define um framework capaz de gerar informações de testes por meio de algoritmos genéticos. A intenção do artigo é fazer este framework em cima de value-based softwares, ou seja, softwares que valorizam o ROI (retorno sobre o investimento) e otimizam financeiramente os objetivos da aplicação. O framework escolhe e prioriza quais são os testes mais importantes, na visão de stakeholders, a fim de aumentar a rentabilidade da aplicação.

Em Shen e colaboradores (2007) (SHEN et al., 2009), o artigo propõe uma maneira de automatizar e gerar casos de teste usando, de forma híbrida, algoritmos genéticos e o algoritmo de Busca Tabu (TB), utilizando-se do TB como a etapa de mutação do algoritmo genético, podendo obter bons resultados na geração automatizada dos testes.

Em Jin com colaboradores, de 2008 (HU et al., 2008), redes neurais foram usadas para ajudar na automatização dos oráculos. Após os estudos, um modelo de redes neurais (ANN) foi utilizado para formar um oráculo. Os experimentos mostraram que essa ANN conseguiu fazer alguns oráculos automatizados, a depender se os requisitos do software estão bem definidos ou não.

Em 2014, o trabalho de Vineeta (VINEETA; SINGHAL; BANSAL, 2014) também aborda o uso de redes neurais para gerar oráculos, com o auxílio do MATLAB e do Jimple. Ele também faz uma abordagem de mineração de dados, usando uma árvore de decisão que consegue chegar a níveis maiores de acurácia, mas funciona apenas com softwares menores.

No artigo de J. Sant (SANT; SOUTER; GREENWALD, 2005), de 2005, o estudo é baseado em modelos estatísticos de sessões que os usuários usam o software para, a partir disso, derivar testes. O estudo analisa se os testes gerados por esse modelo são bons o suficiente para determinar de forma efetiva qual o comportamento do usuário, além de analisar a cobertura e a acurácia desses testes.

4.2 Resultados Derivados da Busca por Itens de Literatura cinzenta

Esta seção sintetiza alguns dos materiais mais informais utilizados para a construção desse trabalho. Todos os materiais possuem como foco o estado atual do tema, ou seja, como a IA auxilia nos testes de software na atualidade.

No artigo do *Blog da Cedro* (TESTES. . . ,), publicado em 2023, o autor ressalta que, além das abordagens de testes de software usando de IA, é possível falar também de um próximo passo: a "a automação de autocorreção", isso é, uma IA que não só detecta os erros e bugs do software, e sim pode criar correções para códigos errados, sem a supervisão de um humano. Isso pode acelerar o processo de desenvolvimento, economizando tempo e esforços, aumentando os lucros da aplicação.

Já no blog da *AppMaster* (O. . . ,), também de 2023, é descrito como a IA pode contribuir para o campo da engenharia de software, mas que também existem vários desafios para tal. O autor elenca que alguns desses desafios podem ser técnicos, como a falta de dados abundantes e de qualidade nos softwares a serem testados (necessários para o bom funcionamento dos algoritmos) e a complexidade de implementar tecnologias novas às já existentes; como também podem ser pessoais, como falta de pessoas qualificadas para implementarem de forma eficaz a IA aos testes, bem como a resistência à mudança, que é

comum de acontecer com tecnologias novas e de pessoas habituadas a práticas tradicionais de teste.

O blog *MIT Technology Review* (A...,) publicou, em 2024, um artigo que demonstra a tendência cada vez maior de integração dos testes automatizados à engenharia de software. Um relatório da Gartner (AUTOMATED...,) baseado em entrevistas com líderes de equipes envolvidas em testes de softwares demonstra que essas práticas estão sendo cada vez mais aceitas e implementadas, e que a integração de IA nos processos de teste pode aprimorar ainda mais a qualidade dos softwares. O artigo ressalta a importância do aprimoramento dessas técnicas para tornar o ciclo de desenvolvimento cada vez mais ágil e eficiente, além de melhor adequado aos requisitos do cliente e mais lucrativo para a empresa.

4.3 Tecnologias atuais

Com o mercado cada vez mais se adaptando a novas demandas, surgem ferramentas e softwares com a promessa de auxiliarem o processo de testes de software usando IA para isso. Alguns podem ser citados, como:

- *Diffblue*: utiliza aprendizado de máquina para gerar testes unitários; (DIFFBLUE, 2024)
- *Testim*: utiliza aprendizado de máquina para criar scripts de testes automatizados; (TESTIM, 2024)
- *Functionize*: utiliza aprendizado de máquina para automatizar os testes na interface de usuário; (FUNCTIONIZE, 2024)
- *Applitools*: utiliza IA para automatizar a detecção de problemas visuais em interfaces de usuário, podendo detectar as mudanças visuais entre cada versão da aplicação; (APPLITOOLS, 2024)
- *Appvance IQ*: usa de IA para gerar scripts de teste automatizados e detectar problemas em aplicativos na web; (APPVANCEIQ, 2024)
- *Tricentis Tosca*: busca melhorar os testes automatizados por meio de IA, a fim de otimizar a cobertura e melhorar a eficiência deles. (TRICENTISTOSCA, 2024)

Ao analisar os estudos selecionados, é possível enxergar que a maioria das abordagens de AM buscam verificar se a cobertura e a efetividade dos testes está sendo o suficiente para testar o software. Alguns autores também citam o uso de AM para testes na interface de usuário, esta sendo uma área pouco abordada dentro de IA, e também para a técnica de oráculo.

O ponto mais recorrente nos estudos que abordam PLN é a verificação de requisitos, sejam eles os técnicos do software como também os de interfaces visuais; ou seja, se existem maneiras que a IA pode auxiliar na verificação desses documentos de requisitos. Ainda em PLN, alguns autores também sugerem usar a técnica para poder estudar as descrições de casos de teste a fim de prever se os casos falharão.

Os trabalhos que possuem como abordagem redes neurais propõem auxiliar na criação de oráculos, já que este é um campo de estudo altamente inspirado em um humano e que busca reconhecer padrões.

Os trabalhos que envolvem modelos estatísticos ou algoritmos genéticos, por sua vez, visam a geração de testes mais adequados de acordo com os requisitos pedidos.

É possível analisar que a fase de teste mais recorrente nos artigos é a de testes de sistema. Em contrapartida, os problemas envolvendo testes em interfaces de usuário são pouco comentados, já que essa é uma atividade que ainda envolve muita interferência humana. Apesar disso, é possível perceber, com os softwares atuais citados anteriormente, que há demanda por ferramentas que possam auxiliar nessa etapa de testes.

Os materiais encontrados em literatura cinzenta ressaltam a importância não só de todas as técnicas descritas na literatura, como também da aplicação adequada delas nos times de desenvolvimento, para que a eficiência das técnicas possa sempre ser aprimorada. As ferramentas atuais encontradas também reforçam a ideia de que há muita demanda por melhorias nos processos de desenvolvimento, já que a automatização de certos processos facilita não só o desenvolvimento da aplicação, como também otimiza o trabalho dos desenvolvedores e melhora o lucro do software.

Capítulo 5

Conclusão

Neste trabalho, foi proposta uma revisão de artigos e de literatura cinzenta na temática de Inteligência Artificial aplicada a Testes de Software seguindo alguns critérios definidos. Além disso, conceitos básicos de ambas as áreas também foram brevemente descritos para melhor compreensão.

A partir dos trabalhos lidos e revistos, a conclusão a que se chega é que, apesar da área ter tido grandes avanços, ainda há muito que pode ser estudado com maiores detalhes. Uma boa parte dos experimentos descritos, apesar de bem detalhados, são feitos em baixa escala, por vezes não levando em consideração grandes softwares ou grandes massas de dados. Com isso, os resultados apresentados são de boa qualidade, mas indicam que a pesquisa neste campo pode chegar a estados ainda mais avançados. Como dito, a etapa de testes de software é custosa e demorada, mas fundamental, e os resultados apresentados neste trabalho demonstram que a inteligência artificial pode cada vez mais ajudar a desenvolver softwares de alta qualidade.

Referências

A evolução dos testes de software através da IA. Disponível em: <https://mittechreview.com.br/a-evolucao-dos-testes-de-software-atraves-da-ia/>. Acesso em: 16 jan. 2024.

AMALFITANO, D. et al. Artificial intelligence applied to software testing: A tertiary study. 2023.

ANSARI, A. et al. Constructing test cases using natural language processing. In: . [S.l.: s.n.], 2017.

APPLITOOLS. 2024. Disponível em: <https://applitools.com/>. Acesso em: 07 jan. 2024.

APPVANCEIQ. 2024. Disponível em: <https://appvance.ai/>. Acesso em: 07 jan. 2024.

AUTOMATED Software Testing Adoption and Trends. Disponível em: <https://www.gartner.com/peer-community/oneminuteinsights/automated-software-testing-adoption-trends-7d6>. Acesso em: 16 jan. 2024.

BADRI, M. et al. Investigating the accuracy of test code size prediction using use case metrics and machine learning algorithms: An empirical study. In: . [S.l.: s.n.], 2017.

BATARSEH, F. A. et al. The application of artificial intelligence in software engineering: A review challenging conventional wisdom. In: _____. [S.l.: s.n.], 2020.

BRAGA, R. et al. A machine learning approach to generate test oracles. In: . [S.l.: s.n.], 2018.

COTRONEO, D.; PIETRANTUONO, R.; RUSSO, S. A learning-based method for combining testing techniques. In: . [S.l.: s.n.], 2013.

DIFFBLUE. 2024. Disponível em: <https://www.diffblue.com/>. Acesso em: 07 jan. 2024.

DURELLI, V. H. et al. Machine learning applied to software testing: A systematic mapping study. 2019.

FELDT, R.; NETO, F. G. D. O.; TORKAR, R. Ways of applying artificial intelligence in software engineering. In: . [S.l.: s.n.], 2018.

FUNCTIONIZE. 2024. Disponível em: <https://www.functionize.com/>. Acesso em: 07 jan. 2024.

GAROUSI, V.; FELDERER, M.; MäNTYLä, M. V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. 2019.

GRANO, G. et al. How high will it be? using machine learning models to predict branch coverage in automated testing. In: . [S.l.: s.n.], 2018.

HALL, T.; BOWES, D. The state of machine learning methodology in software fault prediction. In: . [S.l.: s.n.], 2012.

HEMMATI, H.; SHARIFI, F. Investigating nlp-based approaches for predicting manual test case failure. In: . [S.l.: s.n.], 2018.

HOURANI, H.; HAMMAD, A.; LAFI, M. The impact of artificial intelligence on software testing. In: . [S.l.: s.n.], 2019.

HU, J. et al. Artificial neural network for automatic test oracles generation. In: . [S.l.: s.n.], 2008.

KEYVANPOUR, M. R.; HOMAYOUNI, H.; SHIRAZEE, H. Automatic software test case generation: An analytical classification framework. 2012.

LENZ, A. R.; POZO, A.; VERGILIO, S. R. Linking software testing results with a machine learning approach. 2013.

LI, B. et al. Automatically documenting unit test cases. In: . [S.l.: s.n.], 2016.

MARIANI, L. et al. Automatic testing of gui-based applications. 2014.

O papel da IA nos testes de software. Disponível em: <https://appmaster.io/pt/blog/ai-em-testes-de-software>. Acesso em: 07 jan. 2024.

PANG, Y.; XUE, X.; NAMIN, A. S. Identifying effective test cases through k-means clustering for enhancing regression testing. In: . [S.l.: s.n.], 2013.

ROBLES-AGUILAR, A. et al. Software design and artificial intelligence: A systematic mapping study. In: . [S.l.: s.n.], 2021.

RODRIGUES, D. S. et al. Using genetic algorithms in test data generation: A critical systematic mapping. 2018.

SANT, J.; SOUTER, A.; GREENWALD, L. An exploration of statistical models for automated test case generation. 2005.

SHEN, X. et al. Automatic generation of test case based on gats algorithm. In: . [S.l.: s.n.], 2009.

TESTES de software com IA: como a inteligência artificial está revolucionando o desenvolvimento. Disponível em: <https://www.cedrotech.com/blog/testes-de-software-com-ia-como-a-inteligencia-artificial-esta-revolucionando-o-desenvolvimento/>. Acesso em: 07 jan. 2024.

TESTIM. 2024. Disponível em: <https://www.testim.io/>. Acesso em: 07 jan. 2024.

TRICENTISTOSCA. 2024. Disponível em: <https://www.tricentis.com/>. Acesso em: 07 jan. 2024.

TRUDOVA, A.; DOLEZEL, M.; BUCHALCEVOVA, A. Artificial intelligence in software test automation: A systematic literature review. In: . [S.l.: s.n.], 2020.

VERMA, R. P.; BEG, M. R. Generation of test cases from software requirements using natural language processing. In: . [S.l.: s.n.], 2013.

VINEETA; SINGHAL, A.; BANSAL, A. Generation of test oracles using neural network and decision tree model. In: . [S.l.: s.n.], 2014.

ZHANG, D. Machine learning in value-based software test data generation. In: . [S.l.: s.n.], 2006.