josif 2025
17ª Jornada Científica e Tecnológica
14º Simpósio de Pós-Graduação
do IFSULDEMINAS
ISSN: 2319-0124
REALIZAÇÃO
INSTITUTO FEDERAL
Sul de Minas Gerais
APOIO
CNPq
FAPEMIG

# A APLICAÇÃO DE LLMS NA DETECÇÃO E CORREÇÃO DE BUGS: Uma revisão da literatura

**Gabriel R. RODRIGUES[1]; Paulo C. SANTOS[2]**

## RESUMO

Este trabalho, do tipo Revisão Bibliográfica, investiga a produção científica sobre a aplicação de Modelos de Linguagem de Grande Escala (LLMs), como GPT-4 e Code Llama, na detecção e correção de bugs. A partir de uma revisão da literatura, foram analisadas pesquisas recentes que evidenciam o desempenho superior dos LLMs em comparação aos métodos tradicionais, com destaque para a alta taxa de precisão. Entretanto, os estudos também apontam desafios recorrentes, como a dificuldade em lidar com falhas de raciocínio lógico e a dependência da qualidade dos dados. O processo de seleção incluiu 72 artigos inicialmente recuperados, dos quais 20 compuseram a amostra final após aplicação de filtros de inclusão, exclusão e critérios de qualidade científica. A análise mostra que os LLMs representam um avanço significativo para a automação da depuração, embora a validação humana e o controle de vieses permaneçam cruciais.

**Palavras-chave:** Depuração Automática; Engenharia de Software; Inteligência Artificial; Modelos de Linguagem; Revisão de Literatura.

## 1. INTRODUÇÃO

A detecção e correção de bugs é uma etapa crítica no ciclo de vida do software, impactando na sua confiabilidade e segurança. Esses processos demandam tempo e esforço manual, mesmo com o suporte de ferramentas automatizadas baseadas em análise estática, dinâmica. Essas abordagens ainda enfrentam limitações ao lidar com erros sutis, contextuais ou lógicos.

Nesse contexto, a aplicação de técnicas de Inteligência Artificial (IA) tem se mostrado uma alternativa promissora. Em especial, os Modelos de Linguagem de Grande Escala (Large Language Models – LLMs), como GPT-4, Code Llama e Gemini 1.0, vêm sendo amplamente utilizados para compreender estruturas de código, prever erros e sugerir correções. Tais modelos são treinados com grandes volumes de dados, incluindo repositórios de software, documentações técnicas e fóruns de desenvolvedores, o que os torna capazes de atuar como ferramentas de depuração inteligentes.

Este trabalho tem como objetivo revisar os principais estudos que abordam a aplicação de LLMs na detecção e correção de bugs, discutindo os avanços, desafios e perspectivas dessa tecnologia no contexto da Engenharia de Software.

## 2. FUNDAMENTAÇÃO TEÓRICA

Para compreender as bases da aplicação da Inteligência Artificial (IA) na correção e detecção de bugs, é essencial explorar conceitos fundamentais que sustentam essa abordagem, os

[1]Bolsista PIBIC/FAPEMIG, IFSULDEMINAS – *Campus* Muzambinho. E-mail: gabriellaucher@gmail.com.
[2]Orientador, IFSULDEMINAS – *Campus* Muzambinho. E-mail: paulo.santos@muz.ifsuldeminas.edu.br.

quais serão apresentados neste tópico.

A IA é uma área da ciência da computação dedicada ao desenvolvimento de sistemas capazes de realizar tarefas que normalmente demandam inteligência humana, como análise, raciocínio e tomada de decisões. Cozman, Plonski e Neri (2021) definem a IA como o campo que busca construir artefatos capazes de tomar decisões e resolver problemas de maneira autônoma. No contexto de correção e detecção de bugs, a IA é aplicada para identificar padrões em códigos, analisar comportamentos anômalos e propor soluções para problemas encontrados.

A detecção de bugs é uma tarefa crítica no ciclo de desenvolvimento de software. Tradicionalmente, envolve inspeções manuais ou ferramentas baseadas em regras fixas. Contudo, métodos de IA oferecem uma abordagem dinâmica e adaptativa, empregando algoritmos capazes de identificar erros complexos que podem passar despercebidos por métodos tradicionais.

O Aprendizado de Máquina é uma subárea da IA que permite que sistemas aprendam a partir de dados sem a necessidade de programação explícita. No contexto de detecção de bugs, modelos de aprendizado supervisionado ou não supervisionado analisam códigos-fonte e logs de execução para identificar anomalias.

O Aprendizado Profundo, uma extensão do aprendizado de máquina, utiliza redes neurais profundas para tarefas que envolvem dados complexos e não estruturados, como códigos-fonte. Chaves (2021) destaca que as arquiteturas de redes neurais recorrentes (RNN) e convolucionais (CNN) têm se mostrado especialmente eficazes em tarefas de processamento de linguagem natural.

Diversas ferramentas baseadas em IA vêm sendo desenvolvidas para auxiliar na correção e detecção de bugs. Essas ferramentas combinam algoritmos de aprendizado de máquina, técnicas de PLN e modelos de aprendizado profundo para fornecer análises mais completas e precisas. Além disso, abordagens híbridas que utilizam dados históricos de bugs resolvidos, juntamente com insights gerados por IA, têm se mostrado eficazes para recomendar correções de forma proativa.

## 3. MATERIAL E MÉTODOS

### 3.1 Tipo de Estudo e Critérios de Seleção

Trata-se de uma revisão bibliográfica qualitativa, baseada em protocolos inspirados em revisões sistemáticas. Foram incluídos:

- Artigos publicados entre 2020 e 2025.
- Estudos revisados por pares ou pré-prints de impacto reconhecido (arXiv).
- Trabalhos que abordassem a aplicação de LLMs na detecção ou correção de bugs.
- Publicações em inglês ou português.

### 3.2 Critérios de Qualidade Científica

Para garantir rigor metodológico, os artigos foram avaliados segundo:

- Clareza do desenho experimental.
- Amostra mínima de ≥30 casos de teste ou fragmentos de código.
- Presença de métricas quantitativas (precisão, recall, F1-score, acurácia).

### 3.3 Bases de Dados Consultadas

As pesquisas foram conduzidas entre março e maio de 2025 nas seguintes bases:

- IEEE Xplore
- ACM Digital Library
- SpringerLink

Utilizaram-se os descritores: “LLMs bug detection”, “AI for debugging”, “code repair with GPT”, “deep learning software quality”,”detecção de bugs com LLMs”,”inteligência artificial para depuração de código”, combinados com operadores booleanos.

### 3.4 Fluxo de Seleção dos Estudos

Foram inicialmente recuperados **72 artigos**. Após leitura de títulos e resumos, **32 foram excluídos**, restando 40. Destes, **20 foram removidos** por não atenderem aos critérios de inclusão/qualidade. Assim, **20 artigos** compuseram a amostra final.

Tabela 1 – Fluxo de seleção dos artigos

| Etapa | Quantidade |
|---|---|
| Artigos recuperados (total) | 72 |
| Excluídos por título/resumo | 32 |
| Excluídos por critérios | 20 |
| Incluídos na análise final | 20 |

## 4. RESULTADOS E DISCUSSÃO

A análise dos estudos indicou que os LLMs vêm superando abordagens tradicionais de depuração. A alta precisão de 87,5% na detecção de bugs encontrada por Gustafsson e Flystam (2024) para o GPT-4 corrobora os 85% de acerto observados por Omari et al. (2024) em um estudo similar com código Python, sugerindo um desempenho consistente do modelo. Essa eficácia é frequentemente atribuída à capacidade dos LLMs de aprender com bases de código vastas e diversificadas, como destacado por Wang et al. (2023) .

Omari et al. (2024), ao testar o ChatGPT-4 com 50 trechos defeituosos de código Python,

obtiveram 85% de acerto com explicações satisfatórias para os erros encontrados. Outros estudos, como Wang et al. (2023), destacam a importância de bases de código limpas e diversificadas para garantir a eficácia dos modelos.

Contudo, apesar do alto desempenho em tarefas delimitadas, os modelos apresentam limitações significativas. Bubeck et al. (2023) apontam que falhas de raciocínio lógico e a sugestão de correções não ideais são comuns, especialmente em códigos de alta complexidade, um cenário não totalmente coberto pelos testes com trechos de código isolados dos outros estudos. Embora os LLMs demonstram maior adaptabilidade que ferramentas de análise estática, eles ainda requerem validação humana contínua para garantir a correção e a segurança do código gerado.

## 5. CONCLUSÃO

A presente revisão bibliográfica revelou que os Modelos de Linguagem de Grande Escala (LLMs) representam um avanço significativo na detecção e correção de bugs, apresentando melhores resultados que técnicas tradicionais em diversos contextos. Entre os modelos avaliados, o GPT-4 destacou-se como o mais eficaz, seguido por Code Llama Instruct e Gemini 1.0.

Apesar dos benefícios, ainda existem desafios que precisam ser superados, como a dependência da qualidade dos dados de treinamento, a falta de transparência dos modelos e a necessidade de validação humana. Os resultados apontam um futuro promissor, no qual os LLMs poderão ser integrados a ambientes de desenvolvimento como assistentes de depuração inteligente.

Como recomendação para trabalhos futuros, sugere-se o aprofundamento da pesquisa sobre integração desses modelos em IDEs, o estudo de abordagens híbridas com feedback humano e a análise do desempenho em linguagens menos difundidas.

## REFERÊNCIAS


BUBECK, S. et al. Sparks of Artificial General Intelligence: Early experiments with GPT-4. Microsoft Research, 2023. Disponível em: https://arxiv.org/abs/2303.12712. Acesso em: 10 maio 2025.

CHAVES, R. Redes Neurais Profundas: Aplicações práticas em PLN. Porto Alegre: Bookman, 2021.

COZMAN, F. G.; PLONSKI, G. A.; NERI, A. I. Inteligência Artificial. 2. ed. São Paulo: Blucher, 2021.

GUSTAFSSON, E.; FLYSTAM, I. Large language models and various programming languages: a comparative study on bug detection and correction. 2024. Disponível em: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1870855. Acesso em: 22 jan. 2025.

OMARI, S.; BASNET, K.; WARDAT, M. ChatGPT and debugging: Assessing AI's ability to detect and repair bugs in Python. 2024. Disponível em: https://arxiv.org/abs/2310.08837. Acesso em: 22 jan. 2025.

WANG, L. et al. CodeT: Code Editing with Pre-Trained Transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence. v. 37, n. 11, p. 13037–13045, 2023.