UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

PEDRO OTÁVIO PROBST ZAMPIER

COMPARATIVO DE MODELOS DE LINGUAGEM AMPLA NO DIAGNÓSTICO AUTOMÁTICO DE ERROS EM FUNDAMENTOS DE PROGRAMAÇÃO

GUARAPUAVA

2025

PEDRO OTÁVIO PROBST ZAMPIER

COMPARATIVO DE MODELOS DE LINGUAGEM AMPLA NO DIAGNÓSTICO AUTOMÁTICO DE ERROS EM FUNDAMENTOS DE PROGRAMAÇÃO

Comparison of Wide Language Models in Programming Fundamentals Automatic Error Diagnosis

Proposta de Trabalho de Conclusão de Curso de Graduação apresentada como requisito parcial para obtenção do título de Tecnólogo em Sistemas para Internet pela Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Eleandro Maschio

GUARAPUAVA

2025

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização do Projeto

O ensino de programação representa um dos pilares na formação dos cursos de Computação. No contexto das universidades brasileiras, as disciplinas de fundamentos de programação apresentam elevadas taxas de reprovação e consequente evasão. Nesse sentido, pesquisas indicam que as dificuldades no processo de aprendizagem estão associadas, principalmente, ao "alto grau de abstração, além do tempo e esforço exigidos pela disciplina" (ARIMOTO; OLIVEIRA, 2019).

As dificuldades específicas em disciplinas de programação constituem um fator relevante que pode ser mitigado por meio de ferramentas de apoio educacional (ALVIM; BITTENCOURT; DURAN, 2024). Esse cenário evidencia a necessidade de soluções tecnológicas que facilitem o processo de aprendizagem e reduzam as barreiras iniciais no aprendizado de programação.

A análise automatizada de código representa um tema de crescente interesse na Educação em Computação, especialmente com a disseminação de tecnologias baseadas em Inteligência Artificial (IA). Modelos de linguagem ampla (LLMs)[1] têm apresentado capacidades promissoras na análise de código, trazendo novas possibilidades para o desenvolvimento de ferramentas educacionais relevantes (LEINONEN *et al.*, 2023).

No ambiente educacional, professores enfrentam o desafio de fornecer *feedback* individualizado e de qualidade para turmas numerosas. Tal limitação compromete o processo de aprendizagem, uma vez que erros mais relevantes educacionalmente permanecem sem diagnóstico e tratamento adequados, conforme evidenciado em estudos sobre identificação automatizada desses erros (WATSON; LI; GODWIN, 2025).

1.2 Definição do Problema

O problema central abordado encontra lugar na ausência de estudos comparativos sobre a capacidade de resposta de LLMs no diagnóstico automático de erros em programação. Até então, as pesquisas disponíveis concentram-se em LLMs individuais, não trazendo uma análise comparativa que permita identificar qual apresenta melhor desempenho nesse contexto de diagnóstico de erros (SUN *et al.*, 2024; RAIHAN *et al.*, 2025; PRATHER *et al.*, 2023). Existe potencial de pesquisa em um comparativo que considere fatores como assertividade na identificação de erros, qualidade do *feedback* fornecido (em português), bem como custos operacionais dos LLMs no diagnóstico de erros em fundamentos de programação.

[1] Do inglês, *Large Language Models* (LLMs).

1.3 Relevância do Problema

A relevância da pesquisa é justificada pelos avanços surpreendentes da área de IA nos últimos anos. Faz-se necessário entender as capacidades de diagnóstico dos LLMs (ainda que em constante evolução), para que sejam propostas soluções educacionais factíveis, que integrem harmônica e eficientemente o conhecimento humano (professores) às ferramentas computacionais (IA). Além disso, uma pesquisa recente da Associação Brasileira de Mantenedoras do Ensino Superior (2024) constatou que 71% dos universitários brasileiros usam IA frequentemente nos estudos, sugerindo receptividade e crescente demanda por ferramentas educacionais baseadas nessas tecnologias.

1.4 Justificativa

A escolha do tema é justificada na convergência de fatores tecnológicos, educacionais e sociais que tornam o momento propício para compreender as potencialidades e limitações dos LLMs. O avanço significativo de cada um dos diversos LLMs (e.g., GPT, DeepSeek e Claude), bem como a concentração de estudos que exploram apenas características individuais de cada modelo (SUN *et al.*, 2024; RAIHAN *et al.*, 2025; PRATHER *et al.*, 2023), sugerem a relevância de análises comparativas que considerem capacidades específicas – como o diagnóstico de erros em programação. Do ponto de vista das políticas públicas nacionais, a pesquisa alinha-se com as diretrizes do Plano Brasileiro de Inteligência Artificial 2024-2028, que estabelece como prioritária a “integração de soluções de IA em ambientes educacionais para personalização do aprendizado e melhoria dos resultados educacionais” (BRASIL. Ministério da Educação, 2024).

Justifica-se a motivação pessoal pela oportunidade de aprofundamento em um tema atual e não contemplado pela matriz curricular do curso. Além disso, foi valorizada a chance de realizar pesquisa ainda durante a graduação. Entende-se que há benefício tanto técnico (para o mercado de trabalho) quanto científico (considerando a possibilidade de pós-graduação).

1.5 Desafios do Projeto

O primeiro desafio reside em definir um subconjunto representativo de erros comuns em fundamentos de programação, na linguagem TypeScript, a serem diagnosticados. Esses erros precisam ir além do escopo léxico e sintático, mas se entende que nem todo erro do subconjunto deva ser semântico. Existem, inclusive, questões terminológicas sobre como designar esses erros. O termo, em inglês, *misconception* tem se mostrado uma alternativa, no sentido de “equívoco conceitual” (SILVA, 2024). Toda essa distinção será oportunamente apresentada no projeto futuro.

Desafios relacionados à avaliação incluem o estabelecimento de métricas comparativas, bem como a configuração dos LLMs considerados, que possuem diferentes interfaces e limitações. Nisso, será necessário estudar sobre engenharia de *prompts* e assumir que as respostas podem apresentar componentes probabilísticos (em que, ao se fazer duas vezes uma mesma pergunta, nem sempre se tem a mesma resposta).

Outro desafio reside no próprio processo de formalização e escrita científica, que não é abordado nos quatro primeiros semestres do curso. Também há preocupação em adequar o projeto à correspondência de carga-horária das disciplinas de TCC1 e TCC2, visto que o acadêmico trabalha.

1.6 Contribuição

A contribuição da pesquisa se dá por estabelecer um comparativo entre as LLMs, considerando a capacidade de diagnóstico automático de erros em programação, em um momento no qual estudos se concentram nas características individuais de cada modelo. Embora a pesquisa seja realizada em nível de graduação, acredita-se que o comparativo possa auxiliar em escolhas no desenvolvimento de ferramentas educacionais para o ensino de programação, como também em entender sobre quais aspectos a IA pode contribuir na avaliação de códigos feita por professores (humanos).

1.7 Objetivos

1.7.1 Objetivo Geral

Estabelecer um comparativo, sobre a capacidade de resposta, de três modelos de linguagem ampla no diagnóstico automático de um subconjunto de erros em fundamentos de programação, considerando a linguagem TypeScript.

1.7.2 Objetivos Específicos

No momento da proposta, consideram-se os seguintes objetivos específicos que serão novamente ponderados para o projeto:

1. Selecionar e configurar três LLMs para uma análise comparativa no escopo da pesquisa proposta;
2. Definir, a partir da literatura, um subconjunto representativo de erros frequentes em fundamentos de programação;
3. Adaptar o subconjunto de erros definido para a linguagem TypeScript;

4. Compor um conjunto padronizado (*corpus*) de códigos de teste que contenha, estrategicamente, os erros identificados no subconjunto;
5. Avaliar o percentual de acerto de cada LLM selecionado na identificação e diagnóstico dos erros, estabelecendo métricas quantitativas de desempenho;
6. Analisar qualitativamente a natureza do *feedback* fornecido por cada LLM, considerando clareza e relevância pedagógica;
7. Correlacionar as avaliações qualitativas e quantitativas com os custos operacionais de cada modelo avaliado;
8. Implementar um protótipo de mínimo produto viável (MVP)[2] de API REST, utilizando Laravel, integrado ao modelo de melhor desempenho geral identificado.

[2] *Minimum Viable Product*.

2 PROPOSTA

Apresenta-se uma proposta de análise comparativa sobre a capacidade de diagnóstico automatizado de erros em fundamentos de programação, na linguagem TypeScript, considerando diferentes modelos de linguagem ampla (LLMs). Pretende-se, ao final, implementar um protótipo de mínimo produto viável (MVP) que utilize o modelo com melhor desempenho identificado.

2.1 Descrição do Trabalho

Conforme estudos sobre o uso de LLMs para explicação de códigos e melhoria de mensagem de erro (LEINONEN *et al.*, 2023), torna-se oportuna uma análise comparativa de LLMs proeminentes, que avalie a respectiva capacidade de diagnóstico automatizado de erros em fundamentos de programação. O intuito é fornecer evidências empíricas (alcançadas por meio de experimento e/ou observação) que embasem escolhas para a criação de ferramentas educacionais nesse sentido.

Para maior clareza, seguem exemplos que serão considerados na composição do subconjunto de erros em fundamentos de programação:

1. Não usar variáveis declaradas previamente;
2. Não usar nomes significativos para identificadores;
3. Fazer atribuição sem efeito;
4. Usar o operador de atribuição (=) ao invés do operador de comparação (==);
5. Retestar, em uma estrutura condicional `if-else`, condições já verificadas;
6. Usar um laço `for` somente com a expressão condicional, sem os outros parâmetros, de modo que funcione como um `while`.

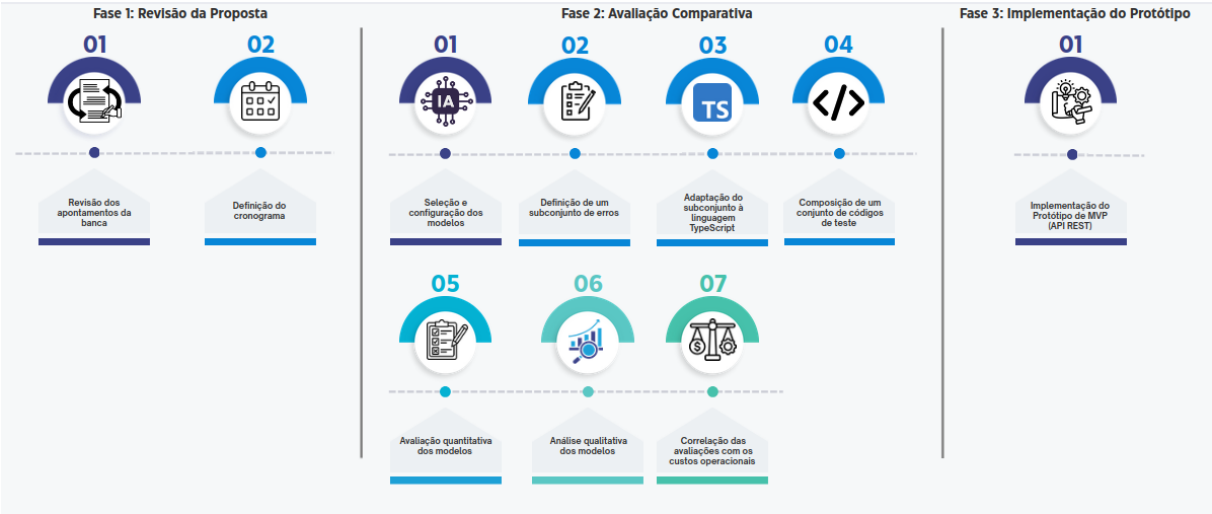
A pesquisa será estruturada em duas fases principais: (1) avaliação comparativa de três LLMs usando um conjunto padronizado de testes; e (2) implementação de um protótipo de MVP funcional de uma API REST, usando o LLM com melhor desempenho.

Haverá esforço para que a abordagem metodológica (definição do subconjunto de erros, composição do conjunto de teste, avaliação qualitativa e quantitativa dos modelos, bem como a comparação e correlação de custo operacional) aproxime-se da ideia de um *benchmark* que possa ser replicável e também atualizável. Isso é importante em função de novos LLMs que possam surgir, como também de novas versões daqueles que serão considerados na pesquisa.

2.2 Abordagem Proposta

Neste momento de elaboração da proposta, preveem-se os seguintes passos metodológicos para o desenvolvimento da pesquisa, divididos em três fases, sendo a primeira preliminar: (1) revisão da proposta; (2) avaliação comparativa e (3) implementação do protótipo de MVP. Serão posteriormente consideradas as atividades de elaboração e defesa tanto do projeto quanto do Trabalho de Conclusão de Curso propriamente dito. As três fases, bem como os passos relacionados, são detalhados na sequência e também pela Figura 1.

Figura 1 – Fases previstas para o desenvolvimento da pesquisa



Fonte: Autoria própria.

2.2.1 Fase 1: Revisão da Proposta

Passo 1: Revisão dos apontamentos da banca

A proposta será adequada levando-se em consideração o que foi observado e sugerido pela banca;

Passo 2: Definição de um cronograma

Na sequência, será definido um cronograma com atividades, entregas fracionadas e reuniões periódicas, até a conclusão da pesquisa. Esse cronograma será revisado no início da disciplina de Trabalho de Conclusão de Curso 2.

2.2.2 Fase 2: Avaliação Comparativa

Passo 1: Seleção e configuração dos modelos

Serão selecionados três LLMs para a análise comparativa. Diante do que foi conside-

rado, há indícios de que sejam: GPT, DeepSeek e Claude. Depois disso, cada modelo será configurado para receber o subconjunto de erros e o conjunto de códigos de teste;

Passo 2: Definição do subconjunto de erros

A partir da literatura técnica da área, será definido um subconjunto representativo de erros frequentes em fundamentos de programação. Mediante critérios estabelecidos, sabe-se que alguns erros precisarão ser desconsiderados para os testes. Pretende-se documentar e categorizar esses erros;

Passo 3: Adaptação do subconjunto de erros à linguagem TypeScript

O subconjunto de erros, então documentado e categorizado, precisará ser adaptado às especifidades da linguagem TypeScript, a fim de fornecer informações relevantes para que o modelo proceda com o diagnóstico automático;

Passo 4: Composição de um conjunto padronizado (*corpus*) de códigos de teste

Esses códigos devem conter erros específicos de forma controlada, mantendo realismo pedagógico e representatividade de erros reais. Entende-se que deverá haver distribuição equilibrada dos erros por entre os códigos, como também repetição (mais de uma ocorrência de um mesmo erro no conjunto de testes);

Passo 5: Avaliação quantitativa dos modelos

Diante do conjunto de códigos de teste, será avaliado o percentual de acerto de cada modelo selecionado. Serão consideradas métricas quantitativas adicionais de desempenho;

Passo 6: Análise qualitativa dos modelos

Atendo-se ao mesmo conjunto de código de testes, será analisada qualitativamente a natureza do *feedback* fornecido por cada modelo, em relação tanto à clareza quanto à relevância pedagógica;

Passo 7: Correlação das avaliações com os custos operacionais dos modelos

Dados os comparativos dos passos 5 e 6, pretende-se detalhar os custos por análise de cada modelo, estendendo isso a projeções de custos para diferentes volumes de uso.

2.2.3 Fase 3: Implementação do Protótipo de MVP

Está sendo estudada a viabilidade, em termos de cronograma, para que a pesquisa avance até uma terceira fase. A intenção é validar o comparativo por meio da implementação de um protótipo de MVP de API REST que integre o modelo de melhor desempenho geral identificado. Deseja-se chegar a um protótipo funcional que realize o diagnóstico de erros de

fundamentos de programação em TypeScript. O protótipo deve fornecer uma resposta estruturada diante de um código submetido, de maneira que mostre, na prática, como um LLM pode ser usado em uma ferramenta desse sentido.

A previsão é que o protótipo seja desenvolvido com as seguintes tecnologias: Laravel 10; banco de dados MariaDB; e com serviços fornecidos por rotas HTTP (*endpoints*) RESTful padronizadas. Contudo, tratam-se de ideias bastante preliminares, que serão amadurecidas ao longo do desenvolvimento do projeto.

2.3 Público-alvo

Como público-alvo direto, primeiramente, tem-se os pesquisadores e desenvolvedores de ferramentas educacionais, interessados em implementar soluções baseadas em LLMs para o ensino de programação. Adicionalmente, professores de disciplinas de fundamentos em programação podem se beneficiar ao melhor entender o diagnóstico de erros por LLMs.

De maneira indireta, estudantes de programação podem ter benefício diante dos dois aspectos recém-mencionados. Primeiro, pela implementação de ferramentas educacionais que proporcionem *feedback* mais assertivo e pedagogicamente alinhado às necessidades apresentadas. Depois, porque, assim como os estudantes já usam LLMs para aprender idiomas (conversação), nada impede que também utilizem de maneira parecida com o uso dos professores de programação, a fim de que tenham diagnóstico e detalhamento de erros.

2.4 Resultados Esperados

Os principais resultados pretendidos pela pesquisa são:

- **Comparação quantitativa:** avaliando-se o percentual de acerto dos LLMs, dentre outras métricas de comparação estabelecidas;
- **Comparação qualitativa:** mediante análise da clareza e da relevância pedagógica do *feedback* fornecido por cada LLM;
- **Comparação de custos operacionais:** considerando-se os dois comparativos anteriores e detalhando-se os custos por análise de cada LLM, incluindo também projeções de custos para diferentes volumes de uso;
- **Protótipo de MVP funcional**: MVP de uma API REST implementada com o LLM de melhor desempenho geral, com código-fonte disponível para adaptação e extensão, tendo o intuito de mostrar a viabilidade de ferramentas no escopo abrangido.

3 CONSIDERAÇÕES FINAIS

A presente proposta busca estabelecer um comparativo sobre a capacidade de diagnóstico automatizado de erros frequentes em fundamentos de programação, considerando códigos em TypeScript, por três modelos de linguagem ampla. Intenciona-se estabelecer critérios qualitativos e quantitativos, correlacionando-os com o custo operacional de cada modelo. Depois, pretende-se implementar um protótipo MVP funcional que mostre a aplicabilidade prática do modelo de melhor desempenho na comparação.

A relevância da pesquisa está na necessidade de compreender as capacidades e limitações dos LLMs na Educação em Computação e, mais especificamente, no diagnóstico de erros em programação. Como principal resultado, o estudo busca fornecer evidências empíricas para orientar escolhas tecnológicas em ferramentas educacionais. A partir disso, também se procura contribuir para melhor integrar o conhecimento humano e as potencialidades da IA. Nesse sentido, o protótipo de MVP implementado, com código-fonte disponível para adaptação e extensão, será um exemplo do uso das evidências fornecidas para o desenvolvimento de ferramentas educacionais no escopo abrangido.

Dentre as principais limitações, citam-se: (a) foco restrito ao subconjunto de erros considerado; (b) diagnóstico de erros concentrado na linguagem TypeScript, sem generalização imediata para outras linguagens de programação; e (c) avaliação direcionada aos três LLMs específicos selecionados, considerando as versões atuais.

REFERÊNCIAS

ALVIM, V.; BITTENCOURT, R. A.; DURAN, R. S. Evasão nos cursos de graduação em computação no brasil. *In*: **Anais do IV Simpósio Brasileiro de Educação em Computação**. Evento Online: Sociedade Brasileira de Computação, 2024. p. 1–11.

ARIMOTO, M.; OLIVEIRA, W. Dificuldades no processo de aprendizagem de programação de computadores: um survey com estudantes de cursos da Área de computação. *In*: SOCIEDADE BRASILEIRA DE COMPUTAçãO. **Anais do XXVII Workshop sobre Educação em Computação**. Belém: Sociedade Brasileira de Computação, 2019. p. 244–254.

ASSOCIAÇÃO BRASILEIRA DE MANTENEDORAS DO ENSINO SUPERIOR. **Inteligência Artificial na Educação Superior**. Brasília, 2024. Relatório de Pesquisa em parceria com Educa Insights.

BRASIL. Ministério da Educação. **Plano Brasileiro de Inteligência Artificial (PBIA) 2024-2028**. Brasília, 2024. Disponível em: https://www.gov.br/mec/pt-br/assuntos/noticias/2024/julho/mec-fara-parte-do-plano-brasileiro-de-inteligencia-artificial.

LEINONEN, J. *et al.* Using large language models to explain programming error messages. **Learning and Individual Differences**, Elsevier, v. 105, p. 102310, 2023.

PRATHER, J. *et al.* The robots are here: Navigating the generative ai revolution in computing education. *In*: **Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education**. New York, NY, USA: Association for Computing Machinery, 2023. (ITiCSE-WGR '23), p. 108–159. ISBN 9798400704055. Disponível em: https://doi.org/10.1145/3623762.3633499.

RAIHAN, N. *et al.* Large language models in computer science education: A systematic literature review. *In*: **Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1**. New York, NY, USA: Association for Computing Machinery, 2025. (SIGCSETS 2025), p. 938–944. ISBN 9798400705311. Disponível em: https://doi.org/10.1145/3641554.3701863.

SILVA, E. P. d. **Misconceptions in correct code: assisting instructors and students by shedding light on what is potentially overshadowed by automated correction**. 2024. 197 p. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas, 2024. Dissertação (Mestrado em Ciência da Computação). Disponível em: https://www.repositorio.unicamp.br/acervo/detalhe/1408046. Acesso em: 13 ago. 2024.

SUN, D. *et al.* Would chatgpt-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? an empirical study. **International Journal of Educational Technology in Higher Education**, Springer, v. 21, p. 14, 2024. Disponível em: https://doi.org/10.1186/s41239-024-00446-5.

WATSON, C.; LI, F. W.; GODWIN, J. L. Automated identification of logical errors in programs: Advancing scalable analysis of student misconceptions. **arXiv preprint arXiv:2505.10913**, 2025.