Avaliando o Papel do GPT-4o na Aprendizagem de Engenharia de Software: Um Relato de Experiência

Bruno Mendes de Carvalho Castelo Branco
Departamento de Computação
Universidade Federal do Piauí
Teresina, Piauí, Brasil
bruno.branco@ufpi.edu.br

Antonio Lisbôa de Melo Neto
Departamento de Computação
Universidade Federal do Piauí
Teresina, Piauí, Brasil
antonion.2048@ufpi.edu.br

Guilherme Amaral Avelino
Departamento de Computação
Universidade Federal do Piauí
Teresina, Piauí, Brasil
gaa@ufpi.edu.br

Vinícius Ponte Machado
Departamento de Computação
Universidade Federal do Piauí
Teresina, Piauí, Brasil
vinicius@ufpi.edu.br

RESUMO

Desde o surgimento dos GPTs no mercado, há uma demanda crescente por estudos que investiguem seu impacto em diversas áreas, incluindo a educação. Este estudo é um relato de experiência estruturado que explora o impacto de ferramentas baseadas em IA — especificamente o modelo GPT-4o via API da OpenAI — na aprendizagem de alunos em uma disciplina de Engenharia de Software. A ferramenta proposta foi projetada com *system prompts* específicos para reduzir a carga cognitiva dos alunos, permitindo que se concentrassem na tarefa pedagógica. Dados de uso de 28 participantes foram coletados e analisados utilizando métodos estatísticos e processamento de linguagem natural. Os resultados indicam que alunos sem experiência profissional escreveram *prompts* mais longos, enquanto aqueles com tal experiência tenderam a curtir mais interações. *Prompts* curtidos tinham maior probabilidade de conter código, e alunos com experiência profissional também se mostraram mais propensos a incluir trechos de código em suas consultas. Este estudo contribui ao propor uma metodologia estruturada para coletar, processar e analisar dados de interação com IAs generativas em contextos educacionais, ajudando a orientar futuras pesquisas empíricas na área.

PALAVRAS-CHAVE

Inteligência Artificial Generativa, Engenharia de Software, Educação, Estatística, Levantamento de Requisitos, Testes, Processamento de Linguagem Natural.

1 Introdução

A Inteligência Artificial (IA) tem se consolidado como uma ferramenta promissora para o desenvolvimento de software, especialmente com o advento da arquitetura *Transformer*, proposta no artigo *Attention is All You Need* [54]. Modelos generativos baseados nessa arquitetura, como o GitHub Copilot e o ChatGPT, têm se destacado pela capacidade de receber descrições em linguagem natural e retornar código coerente [60].

Diversos estudos têm explorado o desempenho dessas ferramentas, avaliando sua precisão [8], usabilidade [52], estratégias de *prompt engineering* [29], e sua comparação com desenvolvedores humanos [3]. No contexto da Engenharia de Software, Nguyen-Duc et al. [32] realizaram um mapeamento sistemático baseado no SWEBOK [47], identificando prioridades de pesquisa que incluem Engenharia de Requisitos, Garantia de Qualidade, Implementação de Software e Educação em Engenharia de Software.

A adoção de novas tecnologias no ensino levanta preocupações sobre o impacto na aprendizagem dos alunos, especialmente após a aceleração imposta pela pandemia de Covid-19 [10]. A facilidade de acesso à informação proporcionada por elas aumenta a independência dos estudantes, mas também pode comprometer a profundidade da análise e compreensão dos conteúdos [39]. Nesse sentido, é essencial investigar como essas ferramentas de IA influenciam o aprendizado, identificando possíveis riscos de dependência tecnológica e limitações no processo de aprendizagem [43], bem como estratégias adequadas para seu uso pedagógico de forma a potencializar seus benefícios e mitigar seus efeitos negativos.

O objetivo central deste trabalho é avaliar o uso de uma ferramenta baseada em LLMs no contexto do ensino de Engenharia de Software. A ferramenta foi projetada com *system prompts* específicos para cada atividade, visando reduzir a carga cognitiva dos alunos e permitir que se concentrassem nas tarefas a eles atribuídas no decorrer da disciplina. Para alcançar este objetivo, o estudo busca responder às seguintes questões de pesquisa:

- **RQ1:** Quais características dos *prompts* estão associadas a uma maior chance de a interação ser avaliada positivamente pelos alunos?
- **RQ2:** Existem diferenças na forma de uso da ferramenta entre alunos com e sem experiência profissional?
- **RQ3:** O nível de experiência profissional influencia na satisfação percebida das interações?

Este estudo contribui com uma abordagem empírica aplicada em sala de aula para investigar o uso de LLMs no ensino de Engenharia de Software, com base na coleta de dados das interações dos estudantes, mediadas por uma ferramenta personalizada, e a sua posterior análise. As principais contribuições incluem: (i) uma metodologia replicável para analisar o uso de LLMs em contextos educacionais reais; (ii) evidências empíricas de que a experiência profissional influencia na forma com a qual os *prompts* são elaborados e na

percepção de qualidade das interações; e (iii) a identificação de padrões linguísticos associados à formulação de *prompts* percebidos como mais eficazes. Esses achados oferecem subsídios práticos para orientar estratégias pedagógicas e promover o uso mais consciente e produtivo de LLMs no ensino de Engenharia de Software.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os fundamentos teóricos relevantes, abordando o processo de desenvolvimento de software, a aplicação de inteligência artificial generativa nesse contexto e as principais técnicas utilizadas para análise dos dados. A Seção 3 discute estudos relacionados ao uso de LLMs no ensino da Engenharia de Software e outras disciplinas da área de Tecnologia da Informação, mapeando o estado da arte e elencando as lacunas. A Seção 4 detalha a metodologia do estudo, incluindo o ambiente de avaliação, o desenvolvimento da ferramenta, a estrutura de coleta de dados e os procedimentos de análise. A Seção 5 apresenta os resultados obtidos a partir das três questões de pesquisa, com análises quantitativas e qualitativas das interações dos alunos com a ferramenta. A Seção 6 discute as implicações dos achados. A Seção 7 analisa as ameaças à validade dos resultados encontrados no estudo. Por fim, a Seção 8 sintetiza as conclusões e, por fim, propõe direções para possíveis trabalhos futuros.

2 Fundamentação Teórica

2.1 Processo de Desenvolvimento de Software

O desenvolvimento de software é um processo estruturado que envolve diversas atividades metodológicas, agrupadas em ações de Engenharia de Software. Essas ações incluem tarefas que visam garantir a produção de artefatos de qualidade, como especificações de requisitos, desenvolvimento, validação e evolução de sistemas [48, 53]. Os processos são:

- **Especificação de Software**: Definição da funcionalidade do software e das restrições operacionais.
- **Projeto e Implementação**: Produção do software conforme as especificações.
- **Validação de Software**: Verificação para garantir que o software atenda às necessidades do cliente.
- **Evolução de Software**: Modificação do software para atender aos requisitos dos clientes que vierem a surgir.

Durante a disciplina na qual foi executado o experimento, os alunos desenvolvem uma solução, tal como o fariam em um ambiente profissional, aplicado cada um destes processos. Cada um deles é dividido em subprocessos, dentre os quais dois foram o foco do presente estudo: Levantamento de Requisitos e Testes.

O Levantamento de Requisitos é a etapa de definir o que o sistema deve fazer, como fazer e sob quais restrições será feito [53] — ou seja, seus requisitos, os quais dividem-se em Requisitos Funcionais e Não-Funcionais [48]. Problemas comuns nessa etapa incluem requisitos incompletos, comunicação ineficiente e mudanças frequentes nos objetivos do projeto [13].

Já os Testes consistem em instruções planejadas à partir dos requisitos, de modo a atestar que o software cumpre com seu papel e, caso contrário, seja corrigido antes do seu uso comercial [48]. No início, eram feitos em uma etapa separada, porém o desenvolvimento ágil impulsionou a adoção de práticas como o Desenvolvimento Dirigido a Testes (TDD), onde testes são escritos antes da implementação do código. Essa abordagem promove a escrita de códigos mais testáveis e melhora a qualidade do sistema [48, 53].

Os testes podem ser realizados em diferentes níveis (unidade, integração e sistema) e técnicas (caixa-preta e caixa-branca) [41]. Aos estudantes da disciplina são ensinados estes conceitos, bem como a como desenvolver estes testes à partir do TDD. Assim, a ferramenta do presente estudo foi desenvolvida visando essa realidade.

2.2 Inteligência Artificial Generativa em Desenvolvimento de Software

A Inteligência Artificial Generativa (GenAI) refere-se a modelos capazes de produzir conteúdo novo e coerente a partir de dados de entrada, baseando-se predominantemente na arquitetura Transformer [14, 54], que utiliza mecanismos de autoatenção para o processamento de informações textuais [45].

O modelo *Codex*, derivado do GPT e treinado em dados públicos do GitHub, é o responsável pela geração de código de ferramentas como o Github Copilot e ChatGPT [17, 34]. Além de serem utilizadas para automação de tarefas, geração de código, tradução de linguagens e análise de padrões [6], com ganhos relatados em produtividade e no auxílio nas tomadas de decisão [9], também são aplicáveis em diversas fases do Engenharia de Software [32].

Dentre as aplicações já avaliadas, estão a automatização da elicitação e classificação dos requisitos [12], sugestão de padrões arquiteturais adequados para um dado sistema [32], geração de código funcional à partir de descrições em linguagem natural [7, 19], auxiliar nos processos de refatoração e documentação do código [1, 27, 33], automatizar testes e analisar cobertura de código [24, 55, 61] e, por fim, traduzir código em linguagens legadas para outras mais modernas [24, 55, 61].

Apesar do que foi elencado anteriormente, ainda há desafios relevantes dentro do contexto analisado, como os riscos de alucinação, a falta de dados para ajuste fino dos modelos, a falta de estudos dentro de contextos reais de trabalho, na compreensão limitada das entradas dos usuários, na garantia da segurança do código gerado e nas limitações com relação ao contexto fornecido ao modelo durante seu uso [32].

2.3 Técnicas de Análise de Dados

2.3.1 Processamento de Linguagem Natural. No contexto da Inteligência Artificial, o Processamento de Linguagem Natural (PLN) é um campo que trabalha com o desenvolvimento de modelos computacionais para resolver problemas de compreensão da linguagem, dentro do contexto, falada por humanos [5, 37]. O campo da PLN fornece técnicas para esses problemas sem que seja necessária uma análise manual dos textos, sendo, portanto, essencial para o entendimento dos *prompts* gerados pelos alunos [5]. Neste estudo, foram utilizadas técnicas como nuvem de palavras (*wordclouds*), N-gramas e análise de coocorrência.

As *wordclouds* são representações visuais que destacam os termos mais frequentes de um texto, com o tamanho de cada palavra proporcional à sua ocorrência no corpus analisado. São ferramentas exploratórias úteis para identificar termos-chave em grandes

volumes de texto [4]. Apesar de suas limitações — como a não agregação de sinônimos, ausência de contexto e sub-representação de termos menos frequentes —, são adequadas como ponto de partida para análises mais aprofundadas e comunicação visual de achados [18, 38].

O uso de N-gramas permite a análise de sequências de *n* palavras dentro do corpus textual [5]. No presente estudo, os unigramas e bigramas foram utilizados para extrair padrões linguísticos nos *prompts*, conforme recomendado na literatura sobre mineração de texto [30, 51].

A análise de coocorrência foi conduzida por meio de matrizes palavra-palavra, quantificando a frequência com que pares de palavras ocorrem conjuntamente nos *prompts* dos alunos. Essa técnica é amplamente empregada na literatura para identificar relações semânticas e padrões recorrentes em textos [15, 28, 46], e foi utilizada neste estudo com o mesmo objetivo.

2.3.2 Estatística Descritiva e Testes Estatísticos. A Estatística Descritiva tem como objetivo organizar, resumir e representar os dados de maneira compreensível, fornecendo uma visão inicial das características da amostra analisada [44]. Neste estudo, as análises foram centradas na avaliação dos dados quantitativos, segmentando-os por classes, o que permitiu comparações entre subgrupos por meio de razões e percentuais.

Os dados foram organizados em tabelas e, quando possível, ilustrados à partir de um gráfico de barras, de modo a facilitar a visualização dos elementos e, assim, obter uma síntese informativa dos dados observados [31, 44].

Para reforçar as análises realizadas, foram aplicados testes estatísticos. Inicialmente, avaliou-se a normalidade das distribuições das variáveis associadas ao tamanho dos *prompts* e das respostas, utilizando o Teste de Shapiro-Wilk, escolhido por apresentar melhores resultados para esse fim [23, 49].

Por ele ter indicado ausência de normalidade nas distribuições testadas, optou-se pelo uso da correlação de *Spearman* para investigar as relações entre variáveis, dado que esse teste não pressupõe distribuição normal dos dados [49], o que permitiu verificar associações entre os atributos dos *prompts* e respostas e as métricas de engajamento por parte dos alunos.

3 Trabalhos Relacionados

Estudos anteriores propuseram padrões de *prompt* reutilizáveis para aplicações dentro do desenvolvimento de software, de modo a auxiliar no processo de desenvolvimento, sendo algumas das aplicações a simulação de APIs para auxiliar no desenvolvimento e a modularização do código [57, 58]. Também já foi destacado o uso de LLMs, como o *GitHub Copilot*, em tarefas tais como a tradução de código entre linguagens de programação e a conversão de linguagem natural em código, chamando atenção para a explicabilidade das respostas [50]. O impacto das configurações destes modelos, como a temperatura e os próprios *prompts*, na qualidade do código, também já foi analisado, evidenciando a necessidade da adaptação delas conforme o contexto onde o modelo será utilizado [11].

A adoção de LLMs no ensino tem produzido resultados diversos. No curso CS50, um tutor baseado em IA foi incorporado, com acesso regulado por um sistema de "vidas", e os relatos indicam impacto positivo na aprendizagem dos alunos [25]. Ferramentas como o *CodeTutor* e o uso de LLMs em sistemas como o *APAS Artemis* trouxeram melhorias, porém os autores também levantaram preocupações com a possível dependência dos alunos, bem como com as respostas genéricas fornecidas pelos modelos [16, 21, 22, 26, 42, 56]. Estratégias desenvolvidas nas ferramentas, como o modelo *Reflect-Respond*, mostraram-se eficazes na construção de conhecimento [20], assim como atividades que incentivam a crítica das respostas da IA [40].

Diferentemente de abordagens mais genéricas, como as discutidas em [22] e [16], a ferramenta proposta neste estudo incorpora *system prompts* elaborados especificamente para contextualizar as interações de acordo com os conteúdos da disciplina. Essa abordagem reduz a carga cognitiva dos alunos ao eliminar a necessidade de eles próprios fornecerem informações contextuais complementares, permitindo, assim, que concentrem seus esforços diretamente nas demandas das atividades.

Outros estudos destacam a necessidade de um uso equilibrado para evitar dependências tecnológicas [56], o que levou nossa metodologia a implementar um sistema de gerenciamento de uso, adotando a limitação por "vidas", já relatada antes como positiva [25], e por contexto, buscando incentivar o uso inteligente por parte dos usuários e direcioná-lo para a prática da disciplina.

Os alunos participantes deste estudo receberam um treinamento em *prompt engineering* com base em outros estudos [57, 58] e nas documentação oficial disponível [35], com o objetivo de maximizar os ganhos por meio da adoção de estratégias de *prompts* e da alfabetização em IA, mais especificamente LLMs, como sugerido em outros trabalhos [26, 40], o que constitui mais um diferencial.

Por fim, por meio da coleta dos dados realizada durante o uso da ferramenta pelos estudantes, foram feitas análises do conteúdo produzido à partir da Estatística, aliados a análises textuais por meio de Processamento de Linguagem Natural. Além de ser mais um diferencial do estudo, também é uma contribuição, visto que pode servir como um norte para experimentos futuros dentro do campo das LLMs aplicadas ao ensino em geral.

A Tabela 1 mostra as principais diferenças entre o presente estudo e aqueles mais relacionados, conforme os seguintes critérios:

- **Critério 1**: Utilizou-se de técnicas de PLN para analisar o conteúdo dos *prompts*?
- **Critério 2**: Os participantes foram treinados para usar a ferramenta/tecnologia analisada?
- **Critério 3**: O foco do estudo era no Ensino em Engenharia de Software?

Tabela 1: Quadro comparativo entre o presente estudo e os mais relacionados.

Trabalho	Critério 1	Critério 2	Critério 3
Liu et al. [25]	Não	Sim	Não
Lyu et al. [26]	Não	Sim	Não
Frankford et al. [16]	Não	Não	Sim
Waseem et al. [56]	Não	Não	Sim
Joštet al. [21]	Não	Não	Sim
Este trabalho	Sim	Sim	Sim

4 Metodologia

O presente estudo trata-se de um relato de experiência estruturado, conduzido em um ambiente real de ensino, na disciplina de Engenharia de Software II do curso de Ciência da Computação da Universidade Federal do Piauí. Nesta disciplina, os alunos utilizaram uma ferramenta de chat, que consome a API da OpenAI, desenvolvida para auxiliá-los durante as atividades.

A escolha por um relato de experiência estruturado é apropriada para contextos de pesquisa emergentes, como o uso de LLMs na educação, onde ainda não existem modelos teóricos consolidados que permitam a condução de experimentos controlados. Optou-se por uma abordagem empírica em um ambiente real de ensino para coletar dados autênticos sobre o uso da tecnologia, abordando lacunas identificadas na literatura que apontam para a necessidade de mais estudos em contextos práticos.

Durante a disciplina, os alunos, organizados em grupos, foram incumbidos de desenvolver uma solução de software completa, simulando um processo real de engenharia de software. A ferramenta de IA foi disponibilizada como um recurso de apoio individual em duas etapas específicas do projeto: *Levantamento de Requisitos* e *Testes*. Em cada uma dessas etapas, cada aluno elaborava *prompts* relacionados às demandas específicas das soluções desenvolvidas pela sua equipe, utilizando a ferramenta para sanar dúvidas ou gerar artefatos.

A ferramenta foi desenvolvida utilizando *Python* com *FastAPI* no *back-end* para processar as requisições dos alunos e transformar os dados para o formato correto de armazenamento, *Firebase* para armazenamento de dados, *Flutter* no *front-end* para receber as perguntas e mostrar as respostas, e a API da OpenAI, para utilizar o modelo GPT-4o.

Foi implementado um sistema de gamificação e limitação de uso. Cada aluno tinha um saldo inicial de 10 "vidas", e uma vida era consumida a cada interação com a ferramenta. As vidas eram recuperadas a uma taxa de uma a cada cinco minutos. Optou-se por essa mecânica com o duplo objetivo de incentivar a elaboração cuidadosa dos *prompts*, dada a sensação de escassez, e, ao mesmo tempo, coletar um volume de dados suficiente para análise. Adicionalmente, as respostas do modelo foram limitadas em número de *tokens* para estimular a análise crítica do conteúdo gerado.

A construção dos *system prompts* seguiu as diretrizes recomendadas pela OpenAI [35], e foram estruturados em três partes principais, conforme ilustrado na Figura 1. A primeira parte dizia respeito ao tema principal sobre o qual a ferramenta iria dialogar com o usuário. A segunda parte é o conhecimento da ferramenta sobre esse tema, consistindo em um resumo sobre os principais conceitos relacionados a esse tema.

Por fim, a terceira parte dizia respeito às restrições na resposta, limitando o número de *tokens* e proibindo expressamente o modelo de responder sobre outro tema que não fosse o que constava no *system prompt*. A seleção de um tema na interface da ferramenta pelo aluno (e.g., "Testes") resultava na associação automática do *system prompt* correspondente à conversa, garantindo que o modelo estivesse sempre contextualizado sem exigir esforço adicional do estudante.

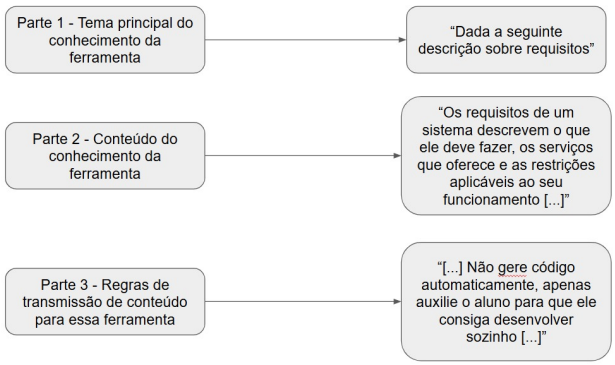

Figura 1: Estrutura básica de um *system prompt*.

4.1 A Ferramenta de Apoio

A jornada do usuário na aplicação, ilustrada na Figura 2, inicia-se com a seleção de um tema, o qual está vinculado a um *system prompt* específico. Em seguida, o aluno insere seu *prompt* e o envia pela interface. A aplicação registra essa entrada no banco de dados e a encaminha à API da OpenAI, que processa a solicitação e retorna uma resposta. Essa resposta é então armazenada e vinculada ao *prompt* original, sendo exibida ao usuário na interface da aplicação.

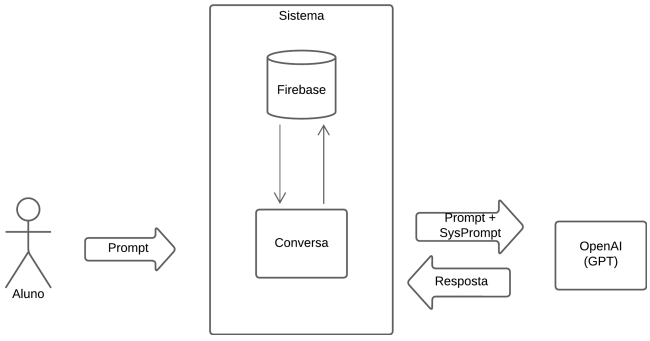

Figura 2: Jornada de uso da ferramenta pelo usuário.

O armazenamento da aplicação foi estruturado em três coleções principais:

- **Users:** Contém informações sobre os usuários, incluindo e-mail, matrícula, nível de experiência, horário da última interação e número de "vidas" disponíveis.
- **Conversations:** Armazena o identificador único de cada conversa, o tema, o *system prompt* utilizado e o timestamp de criação.
- **Prompts:** Inclui os *prompts* enviados, as respostas geradas e as avaliações realizadas pelos usuários.

4.2 Participantes

Participaram do estudo 28 alunos matriculados na disciplina. Não foram coletados dados demográficos detalhados, como idade ou gênero. No entanto, pode-se inferir um perfil relativamente homogêneo, uma vez que a disciplina é ofertada para alunos do 5º período do curso de graduação.

No ato do cadastro na ferramenta, os participantes auto-declararam seu nível de experiência profissional, o que permitiu a divisão da amostra em dois grupos para fins de análise:

- **Apenas Estudante:** 21 alunos que se identificaram como não possuindo experiência profissional no mercado.
- **Atua no Mercado:** 7 alunos, sendo 4 estagiários e 3 profissionais já inseridos no mercado de trabalho.

4.3 Procedimentos e Coleta de Dados

Para cada uma das duas atividades para as quais a ferramenta foi disponibilizada, foi ministrada uma aula de treinamento, que consistiu em aulas expositivas com demonstrações práticas, apresentação de padrões de *prompt* e sugestões para formulação de perguntas eficazes. O material didático, baseado em diretrizes da literatura e da indústria [2, 35, 36, 59], foi disponibilizado a todos, garantindo um entendimento comum sobre o uso da ferramenta.

A interface permitia que os alunos avaliassem cada interação com "curtir"ou "não curtir". Essa avaliação era opcional. Os critérios para a avaliação foram subjetivos, e os alunos foram orientados a curtir a interação caso a considerassem útil para sua atividade e a não curtir caso contrário.

4.4 Análise dos Dados

A última etapa da metodologia envolveu a análise dos dados coletados durante as fases anteriores. A linguagem de programação *Python* foi utilizada devido às suas bibliotecas especializadas em ciência de dados e processamento de linguagem natural, como *Pandas*, *Scipy*, *Numpy* e *NLTK*. Um *endpoint* específico foi desenvolvido na API da aplicação, para exportar os dados de maneira estruturada no formato CSV, conforme descrito na Tabela 2.

Tabela 2: Campos exportados em CSV das coleções `conversations`, `prompts` e `users`.

Campo	Tipo	Descrição
enrollment	string	Número de matrícula do usuário
expertise_level	string	Nível de experiência do usuário [1]
conversation_id	string	Identificador da conversa associada
theme	string	Tema da conversa
liked_interaction	booleano	Indica se a interação foi curtida (verdadeiro/falso)
disliked_interaction	booleano	Indica se a interação foi reprovada (verdadeiro/falso)
prompt	string	Texto do *prompt* enviado ao modelo
response	string	Resposta do modelo ao *prompt*
prompt_tokens	inteiro	Quantidade de *tokens* do *prompt*
response_tokens	inteiro	Quantidade de *tokens* da resposta

Para a análise comparativa, as interações foram divididas em duas categorias. A primeira, "Curtidas", contém as 66 interações que receberam uma avaliação positiva explícita. A segunda, "Não curtidas", agrupa as 14 interações avaliadas negativamente com as 88 que não receberam avaliação alguma. Adotou-se esta heurística devido ao baixo número de avaliações negativas em comparação com as demais. Reconhece-se que esta abordagem introduz um potencial viés, que é discutido na Seção 7.

O processamento de linguagem natural foi realizado apenas nos *prompts*, já que são os textos produzidos diretamente pelos alunos. Após a remoção das *stopwords* em português e das palavras comuns aos padrões de *prompt* ministradas em aula, foram gerados *word-clouds* para identificar palavras mais frequentes, identificados os bigramas para encontrar padrões comuns nos *prompts* e analisadas as coocorrências para avaliar a associação entre palavras nos textos enviados.

[1]Embora o tipo da variável tenha sido definido como string, esse campo consistia em um *enum* com três valores possíveis: "APENAS_ESTUDANTE", "ESTAGIÁRIO"e "PROFISSIONAL", selecionado no ato de cadastro dos alunos na ferramenta.

5 Resultados

Nesta seção, são apresentados os resultados do experimento, organizados de modo a responder às três questões de pesquisa propostas. Ao todo, foram analisadas 167 interações válidas, sendo 135 produzidas por alunos do grupo "Apenas Estudante"e 32 do grupo "Atua no Mercado".

5.1 RQ1: Quais características dos prompts estão associadas a uma maior chance de curtida?

Os *prompts* analisados estão distribuídos, de acordo com os valores das colunas *liked_interaction* e *disliked_interaction*, conforme a Tabela 3.

Tabela 3: Distribuição dos *prompts*, de acordo com a avaliação das interações.

Tipo de Interação	Quantidade
Total de Prompts	168
Curtidas (C)	66
Reprovadas (R)	14
N/A	88
Não curtidas (R + N/A)	102

Haja vista a baixa quantidade de interações marcadas como "Reprovadas", foi adotada a seguinte separação dos *prompts*, para esta e outras análises:

- **Curtidas**: *prompts* que resultaram em interações que foram, ativamente, curtidas, com 66 registros.
- **Não curtidas**: aqueles que, ou não foram avaliados pelos alunos (88 registros), ou foram reprovadas (14 registros), resultando em 102 *prompts* no total.

Além desta heurística, durante a análise dos textos, foi feita a contagem de palavras e, com isso, além das *stopwords*, também foram removidas as formas singular e plural das palavras cujas frequências destoavam bastante das demais (ex.: "livros"e sua forma singular "livro"), ou que estavam relacionadas com o conteúdo de Engenharia de Software (ex.: "requisitos"e "requisito"), e não da aplicação dos alunos em si, buscando facilitar a observação do conteúdo realmente produzido por eles. As palavras removidas estão ilustradas na Tabela 4.

Foi testada a normalidade, por meio do Shapiro-Wilk, dos seguintes dados: *tokens* dos *prompts* e das respostas, quantidade de palavras dos *prompts* e das respostas, obtendo-se como resultado que nenhuma delas segue uma distribuição normal. Por conta disso, o teste de correlação escolhido foi o de Spearman, que mostrou correlações muito fracas entre essas variáveis e a interação ser curtida,

Tabela 4: Frequência das palavras mais recorrentes nos textos analisados.

Palavra	Frequência
usuário	181
sistema	98
usuários	88
livros	82
uso	71
requisitos	66
pode	41
casos	41
caso	33
deve	26
usecase	22
requisito	12
teste	11
testes	10
livro	7

indicando que não há relação entre o tamanho dos *prompts* e das respostas com a probabilidade de serem curtidos, dentro da amostra analisada.

Ao comparar o conteúdo dos *prompts* à partir das wordclouds (Figuras 3 e 4), é possível observar a presença mais forte de termos relacionados à programação (como *import*, *const* e *self*), ou ao domínio do problema (como glicemia, medicamentos e paciente) no primeiro grupo, enquanto que, no segundo, apesar de ainda haver trechos de código com relativo destaque (*string* e *text*), a predominância foi de palavras mais genéricas relacionadas, por exemplo, a ações (permitir e criar), indicando que o contexto fornecido foi mais genérico neste grupo que no anterior.

Comparando os 10 bigramas mais frequentes entre ambos os grupos de *prompts* (Figuras 5 e 6), observa-se que as interações curtidas tendem a conter expressões mais técnicas (como *self assertequal* e *import org*) e relacionadas ao contexto da aplicação (como *dose insulina* e *rede social*). Em contraste, os *prompts* não curtidos apresentaram bigramas com uma natureza mais conceitual (como *diagrama classes* e *especificação software*) ou genérica (como *interesses literários* e *devem poder*), o que pode indicar menor objetividade nesses casos.

Figura 3: Nuvem de palavras dos *prompts* para interações curtidas.

Figura 4: Nuvem de palavras dos *prompts* para interações não curtidas.

Tabela 5: Os 10 bigramas mais frequentes dos *prompts* de interações curtidas.

Par de Palavras	Frequência
atores - envolvidos	11
envolvidos - paciente	10
rede - social	7
dose - insulina	7
usestate - const	7
self - assertEqual	6
import - org	6
public - void	6
when - tratamentoeventorepository	6
junit - jupiter	5

Por fim, ao comparar as 10 coocorrências mais frequentes de palavras extraídas dos *prompts* (Tabelas 7 e 8), nota-se uma diferença discreta no que diz respeito a quantidade de elementos com código (9 pares totalizando 1396 registros na primeira contra 8 pares totalizando 1545 registros na segunda). Apesar da frequência total de pares com código para interações que não foram curtidas ter sido levemente maior neste caso, é importante levar em conta que o número de *prompts* curtidos é de apenas 66, enquanto que o segundo grupo totaliza 102 *prompts*, logo, ao dividir as frequências totais pelas quantidades de interações, obtemos, respectivamente,

Tabela 6: Os 10 bigramas mais frequentes dos *prompts* de interações não curtidas.

Par de Palavras	Frequência
atores - envolvidos	11
envolvidos - paciente	10
diagrama - classes	8
especificação - software	8
atributos - id	8
id - int	8
pressão - arterial	7
devem - poder	6
interesses - literários	6
médicos - próximos	6

21,15 e 15,15, mostrando que as coocorrências concordam com os gráficos mostrados anteriormente.

Tabela 7: As 10 coocorrências mais frequentes dos *prompts* para interações curtidas.

Par de Palavras	Frequência
import - tratamentoeventorepository	204
const - usestate	189
import - remedius	170
eventmock - import	153
eventoid - import	136
import - localdatetime	136
import - org	136
import - usuarioid	136
import - when	136
descrição - paciente	133

Tabela 8: As 10 coocorrências mais frequentes dos *prompts* para interações não curtidas.

Par de Palavras	Frequência
medication - text	225
classname - text	223
id - string	212
text - view	205
const - string	195
comunidade - post	194
perfil - post	170
from - string	165
import - string	165
classname - medication	155

Resposta RQ1: Não foram encontradas correlações significativas entre o tamanho dos *prompts* ou das respostas e a probabilidade de curtida. No entanto, as análises qualitativas indicam que interações mais bem avaliadas tendem a conter termos técnicos, trechos de código e maior contextualização do domínio da aplicação. Esses elementos sugerem que a clareza e a especificidade do *prompt* têm maior influência na satisfação do usuário do que sua extensão textual.

5.2 RQ2: Existe diferenças no uso entre alunos com e sem experiência profissional?

A Tabela 9 mostra um quadro comparativo entre o *prompts* dos alunos de cada um dos dois grupos, com base na quantidade de palavras e na quantidade de *tokens* presentes no texto escrito por eles. Ela mostra que os estudantes sem experiência escreveram mais *prompts*, sendo eles mais longos, em média, que os de alunos que possuem experiência no mercado.

Ao comparar o conteúdo das nuvens de palavras dos *prompts* de alunos apenas estudantes e daqueles que atuam no mercado (Figuras 5 e 6), nota-se uma diferença marcante na natureza dos termos utilizados. Os estudantes tendem a utilizar palavras mais genéricas, como *criar*, *permitir* e *descrição* e alguns trechos de código (como *const* e *string*) com foco mais em ações amplas, além de vocabulário voltado aos fluxos de usuário (como *perfil* e *paciente*). Em contraste, os alunos que atuam no mercado utilizaram, quase que em sua totalidade, trechos de código (como *import*, *tratamentoeventorepository* e *usuarioid*), sem grandes destaques entre elas.

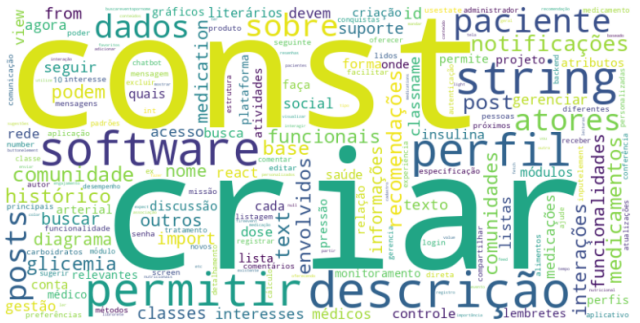

Figura 5: Nuvem de palavras dos *prompts* de alunos apenas estudantes.

Os 10 bigramas mais frequentes nos *prompts* de alunos apenas estudantes (Tabela 10) mostram que, nesse grupo, houve grande prevalência de pares relacionados ao domínio do problema (como *dose insulina* e *rede social*) e conceitos de Engenharia de Software (como *especificação software* e *diagrama classes*). Já no caso dos alunos com experiência profissional (Tabela 11), observam-se bigramas marcadamente técnicos e associados à implementação de software (como *import remedius* e *import org*), sendo a totalidade dos 10, sugerindo a predominância de *prompts* mais orientados ao desenvolvimento.

[2]Em média.

Tabela 9: Métricas comparativas por nível de expertise dos participantes.

Expertise	*Prompts*	Palavras[2]	Tokens[2]	*Prompts* / Palavras	*Prompts* / Tokens
Apenas Estudante	135	68.86	111.39	1.96	1.21
Atua no mercado	33	35.42	86.64	0.93	0.38

Figura 6: Nuvem de palavras dos *prompts* de alunos que atuam no mercado.

Tabela 10: Os 10 bigramas mais frequentes dos *prompts* de alunos apenas estudantes.

Par de Palavras	Frequência
atores - envolvidos	22
envolvidos - paciente	20
dose - insulina	12
pressão - arterial	12
rede - social	11
diagrama - classes	9
especificação - software	9
médicos - próximos	9
from - react	8
atributos - id	8

As 10 coocorrências mais frequentes dos *prompts* de alunos apenas estudantes e daqueles que atuam no mercado (Tabelas 12 e 13) reforçam as diferenças notadas anteriormente. No primeiro grupo, predominam pares de palavras mais descritivos, centrados no domínio do problema (como *descrição - paciente*, *atores - descrição* e *atores - envolvidos*). Já os alunos com experiência no mercado demonstram forte ênfase em termos técnicos, relacionados à implementação (como *import - tratamentoeventorepository*, *import - remedius* e *remedius - tratamentoeventorepository*).

Tabela 11: Os 10 bigramas mais frequentes dos *prompts* de alunos que atuam no mercado.

Par de Palavras	Frequência
remedius - remedius	10
import - remedius	9
import - org	8
import - java	7
self - assertEqual	6
public - void	6
list - eventoativoporousuarioDto	6
when - tratamentoeventorepository	6
junit - jupiter	5
java - util	5

Tabela 12: As 10 coocorrências mais frequentes dos *prompts* de alunos apenas estudantes.

Par de Palavras	Frequência
descrição - paciente	265
atores - descrição	264
descrição - envolvidos	264
atores - envolvidos	242
atores - paciente	242
const - import	242
envolvidos - paciente	242
const - from	233
medication - text	225
classname - text	223

Resposta RQ2: Foram identificadas diferenças claras entre os *prompts* produzidos por alunos com e sem experiência profissional. Aqueles pertencentes ao primeiro grupo tendem a escrever *prompts* maiores, com uma linguagem mais genérica, focando nos fluxos de usuário e em conceitos de Engenharia de Software. Em contraste, os alunos que atuam no mercado apresentam *prompts* com forte presença de trechos de código.

Tabela 13: As 10 coocorrências mais frequentes dos *prompts* de alunos que atuam no mercado.

Par de Palavras	Frequência
import - tratamentoeventorepository	303
import - remedius	260
remedius - tratamentoeventorepository	230
import - localdatetime	181
import - usuarioid	172
import - public	164
import - org	154
eventmock - import	153
localdatetime - tratamentoeventorepository	151
eventatualvpusuarioid - import	146

5.3 RQ3: O nível de experiência influencia na elaboração de *prompts* mais satisfatórios?

Para responder a essa questão, foram separados os *prompts* por nível de expertise e, então, calculadas as relações entre seus *prompts* e a quantidade de curtidas, conforme ilustrado na Tabela 14.

Tabela 14: Distribuição das curtidas por nível de experiência profissional dos participantes.

Expertise	*Prompts*	Curtidas	Percentual (%)
Apenas Estudante	135	49	36,3
Atua no mercado	33	17	51,52

Observa-se que 51,52% dos *prompts* criados por alunos com experiência profissional foram curtidos, contra apenas 36,3% entre os estudantes sem atuação no mercado. Essa diferença sugere que o nível de experiência influencia na formulação de *prompts* mais eficazes, possivelmente devido à maior familiaridade com a natureza das atividades.

As análises utilizando Processamento de Linguagem Natural, apresentadas na resposta à segunda questão de pesquisa, reforçam a afirmação anterior, evidenciando a maior presença de termos relacionados a código e, consequentemente, uma abordagem mais direta nas interações que, de acordo com o que fora descrito na resposta à primeira questão de pesquisa, também influenciou na avaliação positiva.

No entanto, é importante considerar que o número total de *prompts* desse grupo foi menor (33 contra 135), o que pode impactar a robustez da comparação. Além disso, como a curtida era uma ação opcional, é possível que existam interações bem-sucedidas que não tenham sido avaliadas. Ainda assim, os dados analisados sugerem que a experiência prática contribuiu para uma comunicação mais eficiente com a ferramenta e, consequentemente, para a maior frequência relativa de interações curtidas.

Resposta RQ3: Os alunos com experiência profissional tenderam a elaborar *prompts* que julgaram mais satisfatórios. No entanto, devido ao número de interações consideravelmente menor e ao caráter facultativo da avaliação delas, essa diferença pode estar sujeita a viés de amostragem.

6 Discussão

Os resultados deste estudo oferecem implicações práticas e levantam pontos importantes para o uso de IAs generativas no ensino de Engenharia de Software. Com base nos achados, apresentamos um conjunto de recomendações pedagógicas e reflexões para pesquisadores da área.

1. A Importância da Capacitação em Engenharia de Prompts. A análise dos dados revelou uma diferença notável no estilo dos *prompts*: alunos com experiência profissional tenderam a elaborar interações mais técnicas, enquanto estudantes sem essa vivência formularam perguntas mais genéricas, o que reforça a necessidade de, antes de adotar o uso de LLMs em sala de aula, serem desenvolvidas oficinas de engenharia de *prompts*. Ao ensinar clareza, precisão e a importância da contextualização dos *inputs*, elas podem nivelar a turma e melhorar significativamente o desempenho dos alunos ao utilizarem essas ferramentas.

2. O Conteúdo dos Prompts como Métrica de Avaliação Formativa. A análise do conteúdo por meio das coocorrências e dos bigramas mostrou um forte potencial para, a partir do que é escrito pelos alunos, identificar diferentes níveis de maturidade técnica. Os *prompts* deixam de ser apenas uma entrada para a ferramenta e se tornam um artefato de aprendizagem. Assim, sugerimos que o conteúdo dos *inputs* seja explorado como uma métrica de análise do progresso dos alunos. Ao observar a evolução da linguagem de um aluno, o professor pode entender como está a assimilação do conteúdo da disciplina.

3. A Necessidade de Ancoragem Teórica Antes do Uso da Ferramenta. Uma preocupação central no uso de IAs na educação é o risco de os alunos se tornarem excessivamente dependentes da ferramenta sem de fato cristalizar o conhecimento. Para mitigar esse risco, recomendamos fortemente que os alunos sejam ensinados sobre o conteúdo teórico e prático (e.g., o que é um requisito, como se estrutura um teste) antes de utilizarem a ferramenta de IA para auxiliar nessas tarefas. A tecnologia deve ser posicionada como um acelerador para aplicar o conhecimento adquirido, e não como um substituto para o raciocínio e a compreensão fundamental da matéria.

Implicações Adicionais e Reflexões. As diferenças observadas entre os grupos de estudantes apontam para a necessidade de abordagens pedagógicas personalizadas. O fato de que a experiência profissional parece impactar a forma como os alunos interagem com a IA sugere que o uso da ferramenta em sala de aula pode, se não for bem orientado, acentuar desigualdades de conhecimento preexistentes. Portanto, faz-se mister que a implementação de tais tecnologias seja acompanhada de orientações claras, exemplos de uso e, sempre que possível, *feedback* individualizado.

Por fim, vale reforçar a cautela com métricas de engajamento. Conforme observado, apesar de alunos com experiência terem apresentado uma taxa de curtidas proporcionalmente maior, o caráter opcional da avaliação e o volume desigual de interações indicam que métricas como "curtidas"devem ser interpretadas como um sinal, e não como uma verdade absoluta, necessitando sempre do complemento de análises qualitativas mais profundas.

7 Ameaças à Validade

O presente estudo está sujeito a algumas ameaças à validade, que devem ser consideradas durante a interpretação dos resultados.

Uma ameaça significativa à validade interna reside na métrica de avaliação das interações. Como a ação de "curtir"ou "não curtir"era opcional, os dados estão sujeitos a um viés de omissão. Não é possível garantir que as 88 interações não avaliadas sejam de fato neutras ou negativas. A decisão de agrupar interações "reprovadas"e "não avaliadas"na categoria "não curtidas", embora pragmaticamente necessária devido ao baixo volume de reprovações, pode ter introduzido um viés na análise, afetando as comparações feitas na RQ1. A satisfação do aluno é um construto complexo, e uma métrica binária e opcional pode não capturá-la integralmente.

Outra ameaça, relacionada à validade externa, refere-se à amostra. O número reduzido de participantes com experiência profissional (7 de 28) limita a robustez estatística das comparações e a capacidade de generalização dos resultados para populações maiores ou com perfis distintos. Além disso, a não coleta de dados demográficos detalhados impede uma caracterização mais profunda da amostra.

Por fim, como o uso da ferramenta não estava atrelado a notas, o engajamento foi voluntário. Isso pode significar que os alunos mais motivados ou curiosos interagiram mais, gerando uma amostra de dados de uso que pode não ser representativa do comportamento da turma como um todo.

8 Conclusão

Este estudo teve como objetivo investigar o uso de uma ferramenta baseada em LLMs no contexto de uma disciplina de Engenharia de Software, com foco na análise das interações dos alunos e na identificação de fatores que influenciam a formulação de *prompts* mais eficazes. A partir de uma metodologia aplicada em ambiente de ensino real, foram coletados e analisados 167 *prompts*, permitindo responder a três questões de pesquisa relacionadas à qualidade percebida das interações e ao papel da experiência profissional no uso da ferramenta.

Os resultados indicaram que, dentro da população analisada, o tamanho dos *prompts* e das respostas não esteve associado à avaliação positiva da interação. Em vez disso, aspectos como clareza, contextualização e presença de termos técnicos mostraram-se mais relevantes para a percepção de qualidade. Alunos com experiência profissional apresentaram uma tendência a elaborar *prompts* mais concisos e voltados à implementação, enquanto os demais utilizaram vocabulário mais genérico, orientado a conceitos teóricos ou fluxos de usuário.

Além de ter contribuído com evidências empíricas sobre o uso de LLMs no ensino de Engenharia de Software, este trabalho propôs uma metodologia estruturada de coleta e análise de dados, combinando técnicas de estatística e de processamento de linguagem natural. Tais abordagens demonstraram-se eficazes para caracterizar padrões de uso e identificar possíveis indicadores de maturidade técnica dos estudantes.

Os achados reforçam a importância de estratégias pedagógicas que incluam a capacitação em *prompt engineering*, o uso de *prompts* como evidências de aprendizagem e o ensino dos conceitos fundamentais antes da introdução da ferramenta. Tais práticas visam promover o uso mais eficaz e consciente dessas tecnologias, potencializando a formação dos alunos.

8.1 Trabalhos Futuros

Trabalhos futuros podem explorar diversas direções a partir dos resultados obtidos neste estudo, bem como nas suas limitações. A replicação do experimento com um número maior e mais diverso de participantes permitiria análises mais robustas, especialmente em relação ao impacto da experiência profissional. Além disso, estratégias para incentivar a avaliação das interações, como lembretes automáticos ou integração com atividades avaliativas, podem aumentar o engajamento dos alunos e gerar métricas de satisfação mais confiáveis, reduzindo o viés de omissão e proporcionando resultados mais representativos.

Estudos futuros também podem investigar a relação entre a qualidade dos *prompts* e o desempenho acadêmico dos alunos, utilizando as notas obtidas nas atividades como métrica adicional de avaliação da eficácia das ferramentas. Além disso, é possível ampliar essa análise por meio da correlação com dados extraídos de repositórios de software, como histórico de *commits*, cobertura de testes ou qualidade do código produzido, oferecendo uma visão mais abrangente sobre o impacto dessas ferramentas no processo de aprendizagem.

Por fim, seria interessante replicar o experimento utilizando outros modelos, como o Gemini ou o Deepseek, de modo a avaliar a generalização das descobertas deste estudo para outros modelos.

DISPONIBILIDADE DE ARTEFATO

Os dados analisados no estudo, bem como o código utilizado para processá-los, estão públicos e disponíveis no seguinte repositório: https://zenodo.org/records/15477416.

AGRADECIMENTOS

Aos alunos que participaram voluntariamente deste estudo e, aos revisores, pelas considerações.

REFERÊNCIAS

[1] T. Ahmed and P. Devanbu. 2023. Few-Shot Training LLMs for Project-Specific Code-Summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3551349.3559555

[2] Xavier Amatriain. 2023. Prompt Engineering 201: Advanced methods and toolkits. https://amatria.in/blog/prompt201#cot. https://amatria.in/blog/prompt201#cot Acessado em: 2 de dezembro de 2024.

[3] Owura Asare, Meiyappan Nagappan, and N. Asokan. 2023. Is GitHub's Copilot as Bad as Humans at Introducing Vulnerabilities in Code? arXiv:2204.04741 [cs.SE]

[4] Robert L. Atenstaedt. 2023. Word cloud analysis of Family Practice. Does the journal fulfil its editorial policy? *Family Practice* (2023). doi:10.1093/fampra/cmad020 Accessed via SciSpace.

[5] H. M. Caseli and M. G. V. Nunes (Eds.). 2023. *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*. BPLN. https://brasileiraspln.com/livro-pln.

[6] Mark Chen et al. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]

[7] Armin Moradi Dakhel, Vahid Majdinasab, Ashkan Nikanjam, Foutse Khomh, Michael C. Desmarais, and Zhen Ming (Jack) Jiang. 2023. GitHub Copilot AI Pair Programmer: Asset or Liability? *Journal of Systems and Software* 203 (2023), 111734. doi:10.1016/j.jss.2023.111734

[8] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1136–1142. doi:10.1145/3545945.3569823

[9] Thomas Dohmke, Marco Iansiti, and Greg Richards. 2023. Sea change in software development: Economic and productivity analysis of the ai-powered developer lifecycle. *arXiv preprint arXiv:2306.15033* (2023).

[10] Cleriston Izidro dos ANJOS and Deise Juliana Francisco. 2021. Educação infantil e tecnologias digitais: reflexões em tempos de pandemia. *Zero-a-seis* 23, 2 (2021), 125–146.

[11] Jean-Baptiste Döderlein, Mathieu Acher, Djamel Eddine Khelladi, and Benoit Combemale. 2022. Piloting Copilot and Codex: Hot Temperature, Cold Prompts, or Black Magic? (10 2022). http://arxiv.org/abs/2210.14699

[12] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh. 2023. AI-Based Question Answering Assistance for Analyzing Natural-Language Requirements. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, Los Alamitos, CA, USA, 1277–1289. doi:10.1109/ICSE48619.2023.00113

[13] D Méndez Fernández et al. 2017. Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical software engineering* 22 (2017), 2298–2338.

[14] Stefan Feuerriegel, Jochen Hartmann, Christian Janiesch, and Patrick Zschech. 2024. Generative ai. *Business & Information Systems Engineering* 66, 1 (2024), 111–126.

[15] Nathalia Mendes Gerotti Franco and Leandro Innocentini Lopes de Faria. 2019. Colaboração científica intraorganizacional: análise de redes por coocorrência de palavras-chave. *Em questão* (2019), 87–110.

[16] E. Frankford, C. Sauerwein, P. Bassner, S. Krusche, and R. Breu. 2024. AI-Tutoring in Software Engineering Education: Experiences with Large Language Models in Programming Assessments. https://arxiv.org/abs/2404.02548v2

[17] GitHub. 2023. Sobre o GitHub Copilot for Individuals. https://docs.github.com/pt/copilot/overview-of-github-copilot/about-github-copilot-for-individuals

[18] Florian Heimerl, Steffen Lohmann, Simon Lange, and Thomas Ertl. 2014. Word Cloud Explorer: Text Analytics Based on Word Clouds. In *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS)*. doi:10.1109/HICSS.2014.231 Accessed via SciSpace.

[19] S. Imai. 2022. Is GitHub Copilot a Substitute for Human Pair-Programming? An Empirical Study. In *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 319–321. doi:10.1145/3510454.3522684

[20] H. Jin, S. Lee, H. Shin, and J. Kim. 2024. Teach AI How to Code: Using Large Language Models as Teachable Agents for Programming Education. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, New York, NY, USA, Honolulu, HI, USA. doi:10.1145/3613904.3642349

[21] G. Jošt, V. Taneski, and S. Karakatič. 2024. The Impact of Large Language Models on Programming Education and Student Learning Outcomes. *Applied Sciences* 14, 4115 (2024). doi:10.3390/app14104115

[22] T. Kosar, D. Ostojic, Y. D. Liu, and M. Mernik. 2024. Computer Science Education in ChatGPT Era: Experiences from an Experiment in a Programming Course for Novice Programmers. *Mathematics* 12, 629 (2024). doi:10.3390/math12050629

[23] Vanessa Bielefeldt Leotti, Alan Rodrigues Birck, and João Riboldi. 2005. Comparação dos Testes de Aderência à Normalidade Kolmogorov-smirnov, Anderson-Darling, Cramer–Von Mises e Shapiro-Wilk por Simulação. *Anais do 11º Simpósio de Estatística Aplicada à Experimentação Agronômica* (2005).

[24] H. Liu, L. Liu, C. Yue, Y. Wang, and B. Deng. 2023. AutoTestGPT: A System for the Automated Generation of Software Test Cases Based on ChatGPT. Available at SSRN 4584792.

[25] R. Liu, C. Zenke, C. Liu, A. Holmes, P. Thornton, and D. Malan. 2024. Teaching CS50 with AI: Leveraging Generative Artificial Intelligence in Computer Science Education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*. ACM, New York, NY, USA, Portland, OR, USA. doi:10.1145/3626252.3630938

[26] W. Lyu, Y. Wang, T. R. Chung, Y. Sun, and Y. Zhang. 2024. Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24)*. ACM, New York, NY, USA, Atlanta, GA, USA. doi:10.1145/3657604.3662036

[27] A. Madaan, A. Shypula, U. Alon, M. Hashemi, P. Ranganathan, Y. Yang, G. Neubig, and A. Yazdanbakhsh. 2023. Learning Performance-Improving Code Edits. *arXiv:2302.07867* (2023).

[28] Francis Bento Marques, Yuri Bento Marques, and Benildes Coura Moreira dos Santos Maculan. 2021. Coocorrência de palavras-chave em dados abertos da Capes: teses e dissertações em Ciência da Informação. *Múltiplos Olhares em Ciência da Informação* (2021).

[29] Antonio Mastropaolo, Luca Pascarella, Emanuela Guglielmi, Matteo Ciniselli, Simone Scalabrino, Rocco Oliveto, and Gabriele Bavota. 2023. On the Robustness of Code Generation Techniques: An Empirical Study on GitHub Copilot. arXiv:2302.00438 [cs.SE]

[30] Maria Fernanda Moura, Bruno Magalhães Nogueira, M da S CONRADO, Fabiano Fernandes dos Santos, and Solange Oliveira Rezende. 2010. Um modelo para a seleção de n-gramas significativos e não redundantes em tarefas de mineração de textos. (2010).

[31] James E. De Muth. 2019. *Descriptive Statistics and Univariate Analysis*. Springer Science and Business Media LLC, 19–36. doi:10.1007/978-3-030-33989-0_2

[32] Anh Nguyen-Duc et al. 2023. Generative Artificial Intelligence for Software Engineering – A Research Agenda. arXiv:2310.18648 [cs.SE] https://arxiv.org/abs/2310.18648

[33] D. Noever and K. Williams. 2023. Chatbots as Fluent Polyglots: Revisiting Breakthrough Code Snippets. *arXiv:2301.03373 [cs]* (2023). doi:10.48550/arXiv.2301.03373

[34] OpenAI. 2024. Get answers. Find inspiration. Be more productive. https://openai.com/chatgpt/

[35] OpenAI. 2024. Prompt engineering best practices for ChatGPT. https://help.openai.com/en/articles/10032626-prompt-engineering-best-practices-for-chatgpt. Acessado em: 02 de setembro de 2024.

[36] OpenAI. 2024. Prompt Engineering Guide: Write Clear Instructions. https://platform.openai.com/docs/guides/prompt-engineering/strategy-write-clear-instructions. Acesso em: 2 set. 2024.

[37] Daniel W. Otter, Julian R. Medina, and Jugal K. Kalita. 2019. A Survey of the Usages of Deep Learning in Natural Language Processing. arXiv:1807.10854 [cs.CL]

[38] Kayal Padmanandam, Sai Priya V. D. S. Bheri, LaxmiHarshika Vegesna, and Kalakuntla Sruthi. 2021. A Speech Recognized Dynamic Word Cloud Visualization for Text Summarization. In *Proceedings of the International Conference on Inventive Computation Technologies*. doi:10.1109/ICICT50816.2021.9358693 Accessed via SciSpace.

[39] Artur Parreira, Lúcia Lehmann, and Mariana Oliveira. 2021. O desafio das tecnologias de inteligência artificial na Educação: percepção e avaliação dos professores. *Ensaio: avaliação e políticas públicas em educação* 29 (2021), 975–999.

[40] O. Petrovska, L. Clift, F. Moller, and R. Pearsall. 2024. Incorporating Generative AI into Software Development Education. In *Computing Education Practice (CEP '24)*. ACM, New York, NY, USA, Durham, United Kingdom. doi:10.1145/3633053.3633057

[41] Roger S Pressman and Bruce R Maxim. 2016. *Engenharia de software-8*. McGraw Hill Brasil.

[42] M. M. Rahman and Y. Watanobe. 2023. ChatGPT for Education and Research: Opportunities, Threats, and Strategies. *Applied Sciences* 13, 5783 (2023). doi:10.3390/app13095783

[43] Olira Saraiva Rodrigues and Karoline Santos Rodrigues. 2023. A inteligência artificial na educação: os desafios do ChatGPT. *Texto Livre* 16 (2023), e45997.

[44] Sheldon M. Ross. 2009. *Chapter 2 – DESCRIPTIVE STATISTICS*. 9–51. doi:10.1016/B978-0-12-370483-2.00007-2

[45] S. Russell and P. Norvig. 2022. *Inteligência Artificial: Uma Abordagem Moderna* (4 ed.). Pearson, São Paulo.

[46] Rafael Antunes dos Santos, Eliseo Berni Reategui, and Sonia Elisa Caregnato. 2022. Análise de coocorrência de palavras na pesquisa brasileira em HIV/AIDS indexada na Web of Science no período 1993-2020. *Informação & informação. Londrina, PR. Vol. 27, n. 2 (abr./jun. 2022), p. 248-273* (2022).

[47] IEEE Computer Society. 2014. *Guide to the Software Engineering Body of Knowledge* (3 ed.). IEEE, Piscataway, NJ, USA.

[48] Ian Sommerville. 2011. *Software Engineering, 9/E*. Pearson Education India.

[49] Murray R Spiegel. 1993. Estatística, 3ª edição. *São Paulo: Makron* 1994 (1993).

[50] J. Sun, Q. V. Liao, M. Muller, M. Agarwal, S. Houde, K. Talamadupula, and J. D. Weisz. 2022. Investigating Explainability of Generative AI for Code Through Scenario-Based Design. In *27th International Conference on Intelligent User Interfaces, IUI '22*. Association for Computing Machinery, 212–228. doi:10.1145/3490099.3511119

[51] Diego Antonio Rodríguez Torrejón and José Manuel Martín Ramos. 2010. Detección de plagio en documentos. Sistema externo monolingüe de altas prestaciones basado en n-gramas contextuales. *Procesamiento del lenguaje natural* 45 (2010), 49–57.

[52] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 332, 7 pages. doi:10.1145/3491101.3519665

[53] Marco Tulio Valente. 2020. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. Editora: Independente.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30. https://doi.org/10.1016/j.caeai.2023.100147

[55] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang. 2023. Software Testing with Large Language Model: Survey, Landscape, and Vision. *arXiv preprint arXiv:2307.07221* (2023).

[56] M. Waseem, T. Das, A. Ahmad, P. Liang, M. Fahmideh, and T. Mikkonen. 2024. ChatGPT as a Software Development Bot: A Project-based Study. https://arxiv.org/abs/2310.13648v2

[57] Jules White et al. 2023. ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design. (3 2023). http://arxiv.org/abs/2303.07839

[58] Jules White et al. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. (2 2023). http://arxiv.org/abs/2302.11382

[59] Jules White et al. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs.SE] https://arxiv.org/abs/2302.11382

[60] Burak Yetiştiren, Işık Özsoy, Miray Ayerdem, and Eray Tüzün. 2023. Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. arXiv:2304.10778 [cs.SE]

[61] Z. Yuan, Y. Lou, M. Liu, S. Ding, K. Wang, Y. Chen, and X. Peng. 2023. No More Manual Tests? Evaluating and Improving ChatGPT for Unit Test Generation. *arXiv preprint arXiv:2305.04207* (2023).