Modelos Generativos de Linguagem na Construção de Ferramentas de Ensino de Computação com Interface Gráfica

Mateus Otavio Lisboa[1], Hugo Costa[1] ,Pedro Coura[1], Isabela Freitas[1], Maria Lúcia Bento Villela[1], Ricardo Ferreira[1]

[1]Departamento de Informática, Centro de Ciências Exatas e Tecnológicas (CCE)
email: ricardo@ufv.br, Universidade Federal de Viçosa, Brasil

Abstract. *This work performs a quantitative and qualitative analysis of four large language model (LLM) environments for creating educational materials focused on teaching computer science. Evaluation metrics include compilation, execution, and functionality errors, as well as request size, number of interactions, and number of lines of code generated. The evaluated environments are ChatGPT, Claude, Copilot, and Gemini. We explore the construction of incremental requests with directives for creating data input interfaces, algorithm simulations, and outputs with graphical visualizations and/or animations. The evaluated examples cover various domains. Initial results indicate an acceleration in the creation of interactive and attractive educational tools, with the generation of more than 240 functional interfaces from more than 350 trials, representing a success rate of 84%, excluding Gemini LLM, which showed poor performance, highlighting promising directions for using LLMs in the design of educational interfaces.*

Resumo. *Este trabalho realiza uma análise quantitativa e qualitativa de quatro ambientes de modelos de linguagem de grande escala (LLM) para a criação de materiais didáticos voltados ao ensino de computação. As métricas de avaliação incluem erros de compilação, execução e funcionalidade, além do tamanho da requisição, número de interações e número de linhas do código geradas. Os ambientes avaliados são ChatGPT, Claude, Copilot e Gemini. Exploramos a construção de requisições incrementais com diretivas para criação de interfaces de entrada de dados, simulação de algoritmos e saídas com visualizações gráficas e/ou animações. Os exemplos avaliados abrangem diversos domínios, todos documentados e disponibilizados como um conjunto de dados. Os resultados iniciais indicam uma aceleração na criação de ferramentas educacionais interativas e atrativas, com a geração de mais de 240 interfaces funcionais a partir de mais de 350 ensaios, ou seja, uma taxa de sucesso de 84%, excluindo a LLM Gemini, que apresentou baixo desempenho, destacando direções promissoras para uso de LLMs no desenho de interfaces didáticas.*

1. Introdução

O avanço da capacidade dos Modelos de Linguagem de Grande Escala (LLMs - *Large Language Models*) vem possibilitando a criação de diversas ferramentas computacionais. Estes modelos têm a capacidade de gerar respostas coerentes e relevantes para questões relacionadas a diferentes contextos, porém apresentam limitações importantes. A qualidade das respostas pode ser influenciada por diversos fatores, incluindo a requisição (ou

prompt) fornecida ao modelo, os hiperparâmetros utilizados e a diversidade dos dados de treinamento. Neste trabalho, iremos explorar quatro ambientes comerciais de ampla utilização em suas versões gratuitas: ChatGPT, Claude, GitHub Copilot e Gemini.

Além disso, temos os modelos multimodais (texto, som, vídeo, etc..) podem auxiliar na formulação e resolução de problemas, como o uso da metodologia RAG (*Retrieval-Augmented Generation*) para guiar estudantes na solução de problemas através da elaboração de uma série de questões [Yang and Zhu 2024]. Outro caminho é o uso da abordagem REACT (*Reasoning and Acting*)[Yao et al. 2022] para produção de *prompts* para as LLMs gerarem textos estruturados, quebrando problemas complexos em passos intermediários, juntamente com texto de ações que usam as LLMs para resolver os passos criados[Yang et al. 2023]. Entretanto, a geração de figuras estruturadas como diagramas de blocos ainda é pouco explorada neste domínio de LLMs [Zala et al. 2023, Al-Shetairy et al. 2024]. Neste trabalho, desenvolvemos um experimento para avaliação das LLMs comerciais de acesso gratuito para criação de exemplos de ferramentas de ensino interativas usando figuras estruturadas (diagramas, grafos e tabelas), como, por exemplo, uma ferramenta gráfica simples para o ensino da inserção e remoção em uma árvore binária.

Mostramos que, a partir de um *prompt* de 3 a 8 linhas, é possível convergir para o código de uma ferramenta didática com visualização gráfica e interativa. Avaliamos a capacidade de geração e compreensão das LLMs que atenderam às requisições de 5 usuários para os mesmos problemas, seguindo caminhos ligeiramente distintos nas requisições. Para avaliar as quatro LLMs, o experimento envolveu a geração de um conjunto de dados com mais de 350 ensaios, contemplando pelo menos oito problemas distintos, usando as quatro LLMs e no mínimo duas técnicas distintas na escrita dos *prompts*. Como contribuições, podemos destacar: primeiro, a criação de um conjunto de dados (*dataset*) inicial de ferramentas visuais para problemas clássicos. Existem ainda poucos trabalhos na área de *datasets* para diagramas estruturados [Sato et al. 2024], sendo que nossa abordagem vai além, ao incorporar simulação interativa aos diagramas. Uma segunda contribuição é o uso de métricas quantitativas (tamanho do prompt, linhas de código, ...) e qualitativas (interface funcional) para guiar o desenvolvimento. A terceira contribuição é uma metodologia para avaliação da capacidade das LLMs na criação de código para algoritmos com visualização. Enquanto a avaliação do uso de LLMs para ensino de programação já é bem estudada [Kiesler and Schiffner 2023], existe uma lacuna na criação de material interativo e visual.

Este trabalho está estruturado da seguinte forma: a seção 2 apresenta os fundamentos das técnicas de *prompt*, a seção 3 introduz a metodologia, as seções 4 e 5 mostram os resultados visuais para uma avaliação qualitativa e as métricas de avaliação quantitativa, respectivamente. Por fim, a seção 7 apresenta as conclusões e direções futuras.

2. Fundamentos

O desenvolvimento das requisições ou *prompts* pode seguir algumas diretrizes para facilitar e melhorar as respostas das ferramentas de LLM. As diretivas mais usuais [Logan IV et al. 2021] seguem o seguinte fluxo: **Contexto:** fornece informações para que a IA entenda o cenário ou a situação; **Tom:** define o estilo desejado para a resposta (formal, informal, amigável, técnico, etc.), ajudando a IA a adequar-se ao tipo de

comunicação pretendido; **Clareza e Especificidade:** estabelece a importância de ser claro e específico sobre o que se deseja obter, evitando perguntas vagas que podem gerar respostas imprecisas; **Objetivo:** indica a finalidade do *prompt*, como buscar uma explicação, sugestão ou análise; **Exemplos:** fornece modelos do tipo de resposta esperada, auxiliando a LLM na compreensão da tarefa; **Limitações:** especifica restrições que o usuário deseja impor à resposta (como tópicos ou estilos a serem evitados); **Formato:** define a estrutura desejada para a saída (lista, programa, parágrafo, tabela, etc.). A seguir, ilustraremos nossa abordagem com um exemplo de *prompt*.

1. **Contexto**: Desenvolver uma ferramenta de ensino do método de Kruskal para árvore geradora usando Google Colab, iwidget para interface interativa e Graphviz.
2. **Tom**: Com interface para executar passo a passo com um botão ”passo”e um de reset para sortear novo grafo.
3. **Clareza e Especificidade**: Mostrar com Graphviz gerando PNG para visualizar com destaque os vértices e arestas incluso a cada passo.
4. **Objetivo**: Ferramenta para ensino do algoritmo de Kruskal de árvore geradora.
5. **Exemplos**: Gere como exemplos grafos com 10 a 15 vértices.
6. **Limitações**: Sempre que executar um passo, atualizar o desenho.
7. **Formato**: Usar Graphviz para os grafos.

O estilo mais adequado para um *prompt* é fornecer instruções claras e precisas [Xu et al. 2023], utilizando delimitadores como aspas ou chaves para separar as instruções de exemplos ou trechos que se deseja melhorar. As estratégias são classificadas como **one-shot** quando acompanhadas de um exemplo, ou **few-shot** quando utilizam múltiplos exemplos [Logan IV et al. 2021]. Por outro lado, são denominadas **zero-shot** quando utilizam um *prompt* simples sem exemplos.

Uma das estratégias mais populares é a “Chain-of-Thought” (CoT)[Wei et al. 2022], na qual a requisição de um problema complexo é decomposta em uma sequência de etapas menores, explicando o raciocínio em cada etapa durante a interação com a LLM. Outra técnica semelhante é a “least-to-most prompting”[Zhou et al. 2022], que consiste em decompor um problema complexo em uma série de subproblemas mais simples, que são então resolvidos sequencialmente. Existe também a “golden chain-of-thought”, que, além de decompor o problema em uma sequência de etapas lógicas, fornece explicações explícitas sobre o raciocínio e o processo de pensamento em cada estágio [Del and Fishel 2022]. Outra técnica que busca enriquecer os *prompts* para fornecer mais contexto é o “generated knowledge”[Liu et al. 2021], que aproveita a capacidade das LLMs para gerar informações potencialmente úteis sobre uma determinada pergunta ou prompt antes de gerar uma resposta final. Na mesma linha, existe a “tree of thoughts” (ToT)[Yao et al. 2024], que é uma abordagem estruturada para guiar os LLMs explorando múltiplos caminhos de raciocínio. Ao contrário dos prompts lineares tradicionais, o ToT permite que os LLMs considerem várias soluções e estratégias possíveis, incluindo olhar para a frente, retroceder e fazer autoavaliações, tornando-o mais interativo e adaptável à complexidade da tarefa em questão.

Existem também trabalhos que buscam catalogar os prompts [White et al. 2023] para derivar padrões que podem ser usados de forma sistemática no projeto das estratégias de prompts. Outra possibilidade é usar a própria LLM para criar e ajustar os prompts

antes de usá-los, em um processo de otimização de prompts, ajustando sistematicamente a precisão e a relevância deles, reduzindo a necessidade de tentativa e erro manual.

3. Metodologia

Nesta seção, descrevemos a metodologia adotada para avaliar LLMS e prompts para a geração de ferramentas gráficas e interativas para o ensino de computação. Primeiro, selecionamos um conjunto de 9 temas comuns em fundamentos de ciência da computação: ordenação por inserção, ordenação por seleção, inserção/remoção (I/R) em árvore binária, I/R em heap, I/R em tabela hash, menor caminho, árvore geradora mínima, máquina de Turing e o jogo da velha. Apesar de o jogo da velha ser uma exceção entre os temas, ele representa uma matriz como base para manipulação, que é um tópico presente no ensino. Além dos 9 exemplos básicos, avaliamos também alguns exemplos de outros tópicos para avaliar isoladamente alguns aspectos.

Cinco usuários, sendo 1 docente e 4 discentes, elaboraram os prompts. Cada usuário trabalhou de forma independente, o que nos permitiu verificar a consistência das ferramentas. Apesar de a ideia básica do prompt ter sido compartilhada, cada usuário fez a descrição com seu próprio estilo, criando prompts simples de 3-8 linhas.

Além disso, cada usuário avaliou uma técnica de prompt avançada diferente, como prompt detalhado [Chen et al. 2023], zero shot Chain-of-Thought [Wei et al. 2022], golden Chain-of-Thought [Del and Fishel 2022], generated knowledge [Liu et al. 2021] e least-to-most [Zhou et al. 2022]. Todos os experimentos foram documentados e estão disponíveis anonimamente para revisão. Devido a restrições de espaço, apresentaremos apenas alguns exemplos e um resumo dos principais resultados, maiores detalhes estão disponíveis da documentação suplementar [de Viçosa 2024].

É importante destacar que, ao variarmos ligeiramente as descrições com usuários diferentes, testamos a capacidade e reprodutibilidade das ferramentas durante um período de 2 meses. Como as ferramentas estão em constante atualização e aprendem com o perfil de cada usuário, esse aspecto também influenciou o desempenho. Nosso objetivo é testar a robustez em diferentes cenários, considerando a versão sem assinatura, pois é provável que ocorra uma maior restrição de recursos sem assinatura devido aos altos custos de uso de LLMs [Khowaja et al. 2024].

4. Ferramentas com Interface Visual Geradas

Devido à restrição de espaço, não é possível apresentar as mais de 250 ferramentas visuais geradas. Portanto, destaca-se apenas as características visuais e funcionais mais relevantes de alguns exemplos. A figura 1 apresenta 5 interfaces para o ensino de tabela hash aberta (que permite múltiplos elementos por entrada). Nestas interfaces, o usuário pode realizar operações de inserção, remoção e consulta na tabela. As interfaces mostradas nas figuras 1(a,c,d) utilizam a biblioteca **svgwrite** para criar desenhos gráficos vetoriais e animados. Entre estas, apenas a interface da figura 1(c) exibe a progressão quadro a quadro, enquanto as demais implementam sobreposição de elementos, criando um efeito de animação. Já as interfaces ilustradas nas figuras 1(b,e) foram desenvolvidas utilizando a biblioteca **graphviz**. Para otimizar o processo de desenvolvimento e obter resultados satisfatórios com menos iterações, a maioria dos prompts incluiu especificações explícitas sobre o uso destas bibliotecas gráficas.

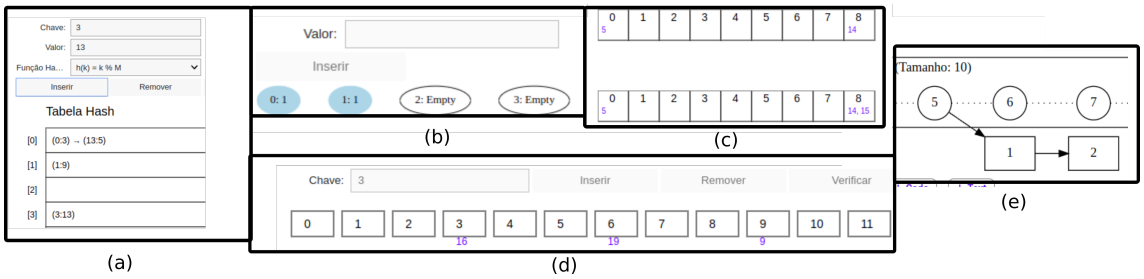

Figura 1. Hash: (a) Tabela Vertical; (b) Graphviz; (c) Horizontal quadro a quadro; (d) Horizontal com animação; (e) Graphviz com lista encadeada por entrada;

Por exemplo, um dos 10 prompts usados para criar uma ferramenta de visualização do hash está ilustrado no Quadro 1. Foram necessários três interações (3 ”shots”) usando a ferramenta *chatgpt* para obter um resultado satisfatório.

Quadro 1: Exemplo de uma sequência de Prompts para Tabela Hash

Shot 1. Por favor, você poderia escrever um código usando iwidget para google colab que mostre a inserção com hash aberto em um hashset?
Shot 2. Que tal representarmos o hashset graficamente usando graphviz?
Shot 3. Agora seria legal poder remover elementos. Pode escolher se vai usar o botão de adicionar, removendo o elemento se ele já existir, ou criar um botão separado para remover elementos (Ele escolheu o botão separado)

A figura 2 mostra três ferramentas adicionais geradas para o ensino de ordenação. As interfaces exibem o pseudo-código, que pode ser animado juntamente com a visualização dos elementos no vetor. O usuário pode trocar os elementos do vetor. As figuras 2(a-b) ilustram o algoritmo de inserção com a visualização do vetor, enquanto a figura 2(c) apresenta a visualização com barras representando os valores do vetor para o algoritmo de seleção. Essas interfaces demonstram que é possível acompanhar a execução passo a passo do algoritmo, com animações tanto no pseudo-código quanto no vetor.

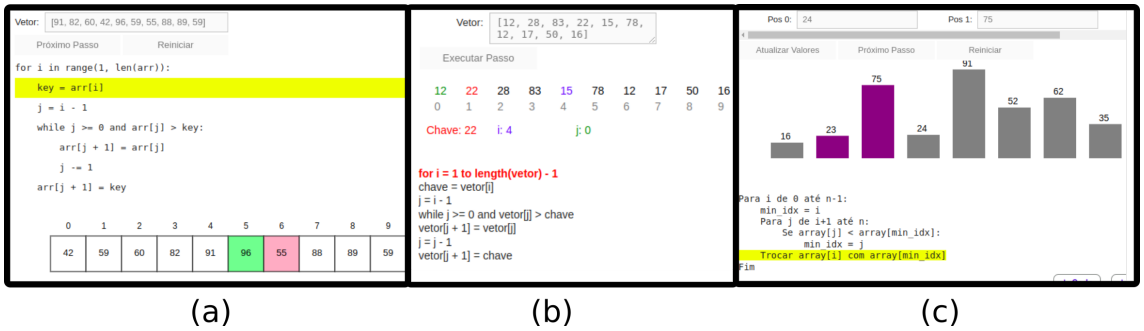

Figura 2. Ordenação com destaques: (a) Cores; (b) Indices; (c) Barras;

A figura 3 mostra três interfaces para o jogo da velha e uma interface para a máquina de Turing. A máquina possui uma entrada dinâmica, com animação sincronizada entre a fita e o diagrama de estados durante o processamento de cada caractere de entrada. A máquina do exemplo está programada para incrementar em uma unidade o número binário na fita.

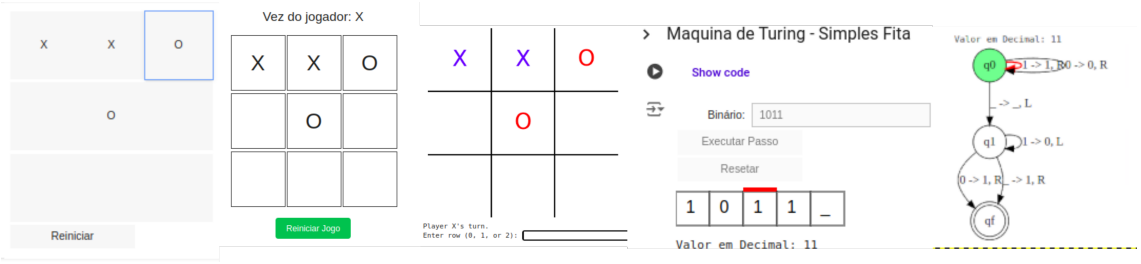

Figura 3. Três Interfaces de Jogo da Velha e um interface de Máquina de Turing

Para testar a capacidade para geração de jogos, solicitamos a criação do jogo resta-um ilustrado na figura 4. Podemos observar a diversidade de soluções apresentadas com uma boa apresentação e funcionalidade, gerando mais 4 exemplos de ferramentas visuais de ensino. Como o código gerado é comentado e estruturado, pode ser usado com material didático para gerar extensões ou adaptações para outros jogos similares.

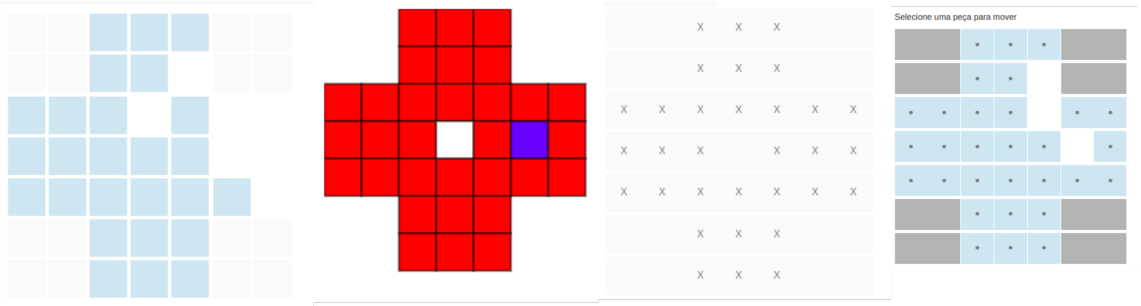

Figura 4. Quatro Interfaces de Jogo Resta-um

A figura 5 mostra duas interfaces para árvore geradora, a primeira mais simples com destaque apenas para as arestas, a segunda mostra os agrupamentos sendo formados pelo algoritmo de Kruskal e a visualização do pseudo-código.

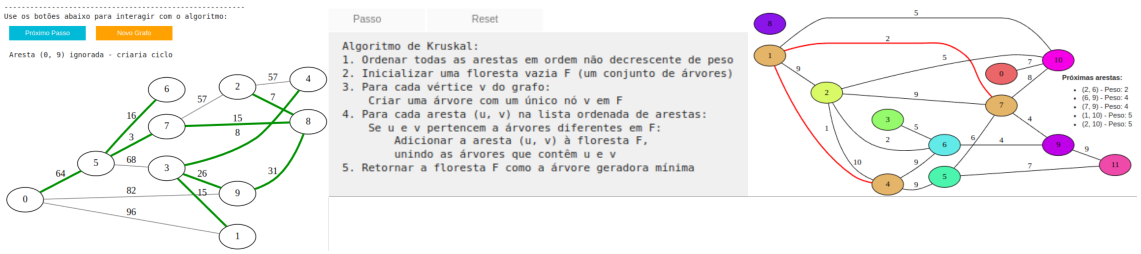

Figura 5. Duas Interfaces para Árvore Geradora Mínima

A figura 6 mostra três ferramentas adicionais que destacam a versatilidade das LLMs na geração de interfaces e ferramentas de simulação. A figura 6(a) exibe uma máquina de estados construída a partir da tabela de transição, juntamente com o simulador. A figura 6(b) ilustra o ensino de inserção e remoção em uma lista encadeada, e a figura 6(c) apresenta um simulador de RISC-V gerado a partir de um prompt com menos de 100 palavras, incluindo um campo para editar o código e simular um subconjunto de 15 instruções básicas que não foram descritas no prompt.

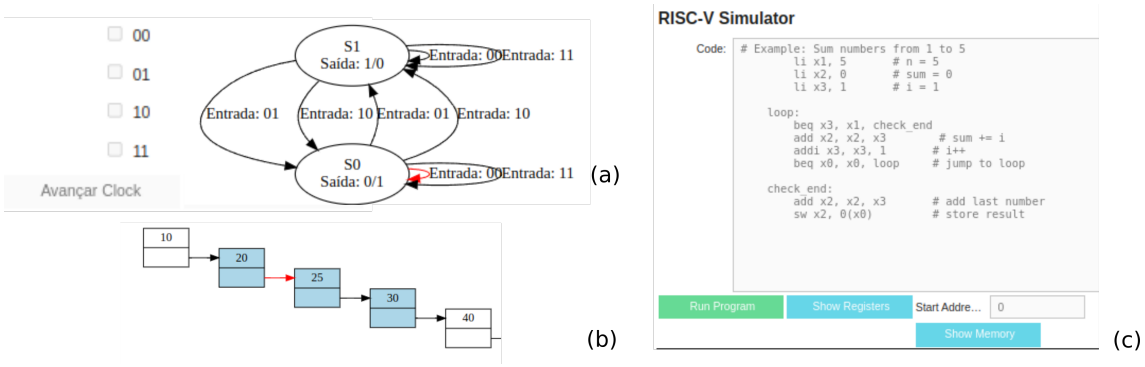

Figura 6. (a) Máquina de Estados; (b) Lista Encadeada; (c) Simulador Risc-V

Outro resultado é gerar material para extensões. A figura 7 mostra extensões para ensino de árvore binária onde ferramenta mostra o antes e depois da inserção e remoção, mostra o caminho para uma busca e como a árvore pode ser armazenada em um vetor.

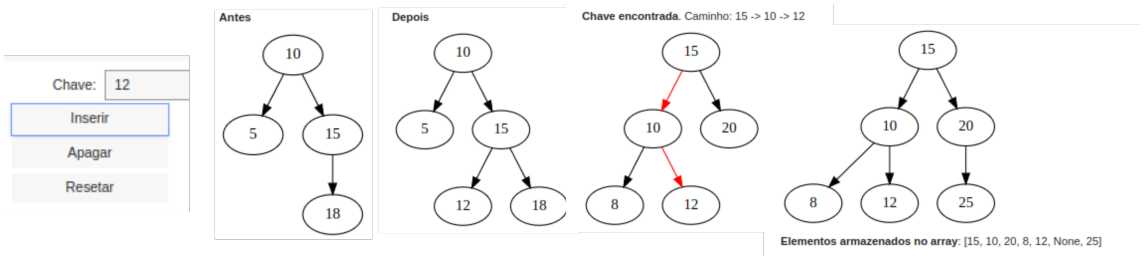

Figura 7. Extensão da Ferramenta Árvore Binária

5. Resultados Experimentais

5.1. Quantitativos

Mostraremos primeiro os resultados quantitativos, nos quais buscamos estabelecer métricas para comparar as LLMs utilizadas. A tabela 1 apresenta um resumo com um total de 361 prompts para a geração de ferramentas utilizando as 4 LLMs comerciais. Obtivemos com sucesso 274 ferramentas, das quais ilustramos apenas 21 na seção 4, devido às restrições de espaço. As colunas ”erros de compilação”(**Comp.**) e ”erros de execução”(**Exec.**) mostram a porcentagem de erros por tentativa de prompt para cada LLM. Podemos observar que as LLMs Copilot e Claude apresentaram apenas 5% e 12% de erros de compilação, e 20% e 26% de erros de execução, respectivamente. No geral, as LLMs Claude, Copilot e ChatGPT tiveram um desempenho semelhante, resultando em mais de 81,8% de ferramentas gráficas e interativas funcionais, em contraste com o Gemini, que resolveu apenas 40% dos prompts. Em termos de tamanho total dos prompts em palavras, após a troca de mensagens para ajustes, o Copilot foi o mais bem-sucedido, com uma média de 145 palavras por problema. Em média, 50 palavras adicionais são suficientes para ajustar o prompt inicial na interação com as LLMs. Devido a limitações de entrada do Gemini, poucas técnicas com prompts mais detalhados foram avaliadas, o que resultou em um tamanho menor dos prompts, já que os maiores não foram testados. A última coluna mostra o número de ferramentas para cada LLM que foram testadas. Por

Tabela 1. Análise de Desempenho por Ferramenta

LLM	Erros		Funcionou	Tam. Prompt		Testes
	Comp.	Exec.	Correto (%)	Primeiro	Total	
Claude	0.12	0.26	85.6	114.4	165.2	97
Copilot	0.05	0.20	83.2	114.4	145.5	101
Chat Gpt	0.05	0.29	81.8	114.7	164.2	99
Gemini	0.19	0.43	40.6	81.2	154.0	64

exemplo, 97 testes de geração foram realizados com a LLM Claude, que respondeu com sucesso e qualidade em 85,7% dos casos, gerando 83 ferramentas.

A tabela 2 apresenta os resultados por problema, excluindo-se a LLM Gemini devido ao seu desempenho insatisfatório. Cada problema foi submetido a avaliação com no mínimo 29 prompts distintos, com alguns recebendo testes adicionais, mantendo um padrão de comportamento consistente entre as avaliações. A máquina de Turing revelou-se o desafio mais complexo, apresentando uma taxa de sucesso de 56.7%, enquanto a implementação da árvore binária alcançou o melhor desempenho, com 100% de sucesso. O desempenho médio geral situou-se em 84%. Na maioria dos casos, limitou-se o número de tentativas entre 5 e 8 por prompt, considerando que, quando a LLM não apresenta uma resposta adequada inicialmente, é mais eficiente reformular o prompt do que persistir em tentativas adicionais, pois raramente há convergência para um resultado satisfatório nestas situações. O problema do caminho mínimo apresentou-se como o segundo mais desafiador. Os demais problemas demonstraram taxas de sucesso superiores a 86%.

Tabela 2. Análise de Desempenho por Problema (excluindo Gemini)

Problema	Erros		Funcionou	Tam. Prompt		Testes
	Comp.	Exec.	Correto (%)	Primeiro	Total	
Arvore Binária	0.00	0.09	100.0	98.0	112.7	30
Grafo Arvore Geradora	0.03	0.17	86.2	149.2	171.9	29
Hash	0.04	0.14	93.1	113.3	182	30
Heap	0.08	0.26	91.2	114.8	144.0	34
Jogo da Velha	0.09	0.22	88.9	104.2	144.7	36
Menor Caminho	0.11	0.21	80.0	127.0	188.5	30
Ordenação Inserção	0.06	0.21	86.7	109.8	179.1	30
Ordenação Seleção	0.09	0.20	86.7	111.6	160.0	30
Turing	0.07	0.31	56.7	109.0	164.7	30

A tabela 3 apresenta o desempenho comparativo entre 4 alunos de iniciação científica e o docente. O aluno 3 e o docente expandiram seus testes para além dos 9 problemas básicos, incluindo implementações adicionais como jogo resta-um, autômatos, máquina de estados e protocolo de comunicação. Os três primeiros alunos apresentaram desempenho similar na formulação de prompts. O aluno 4 adotou uma abordagem diferenciada, utilizando prompts mais concisos, e obteve resultados positivos. Esta estratégia de simplificação dos prompts com ajustes demonstrou potencial para melhor convergência, merecendo uma avaliação mais aprofundada em estudos futuros. O docente já possuia experiência prévia no uso de LLMs e avaliou prompts mais extensos e detalhados.

A figura 8 mostra o desempenho de gerar uma ferramenta funcional por LLM e por problema. A LLM Claude tem 100% sucesso para árvore geradora, menor caminho

Tabela 3. Análise de Desempenho por Aluno (excluindo Gemini)

Aluno	Erros		Funcionou	Tam. Prompt		Testes
	Comp.	Exec.	Correto (%)	Primeiro	Total	
Aluno 1	0.08	0.30	75.5	153.7	196.4	53
Aluno 2	0.05	0.09	79.6	102.4	167.1	54
Aluno 3	0.07	0.39	78.3	125.1	154.6	69
Aluno 4	0.00	0.17	93.0	66.4	101.5	57
Docente	0.17	0.34	90.6	123.6	173.3	64

e hash, a LLM Copilot para heap, inserção e hash e a LLM Chatgpt para jogo da velha e seleção. Todas tem sucesso para árvore binária e o pior desempenho é a máquina de Turing, onde a LLM Claude tem 80% de sucesso nas tentativas.

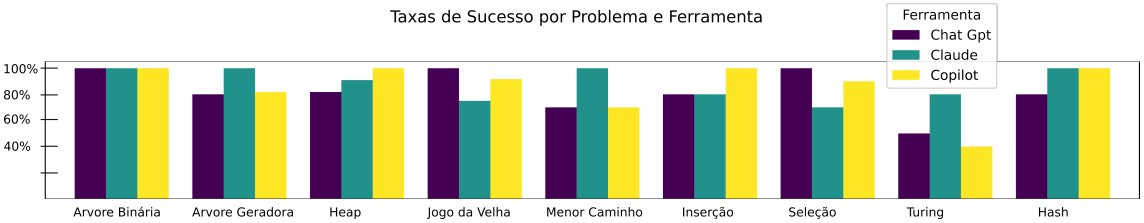

Figura 8. Desempenho das LLMs: Chatgpt, Claude e Copilot por problema

6. Material Suplementar

Este artigo disponibiliza um material complementar utilizando um Google Colab como índice [de Viçosa 2024]. O material está organizado em 4 principais seções: animações, resumos das métricas, experimentos com prompt simples e experimentos com prompts detalhados. A primeira seção do material suplementtar contém 53 animações no formato GIF, onde podemos observar de forma sucinta um exemplo ilustrativo das ferramentas gráficas produzida. A segunda seção apresenta ferramentas para exploração dos dados gerados usando recursos interativos que permitem selecionar o problema, a ferramenta, a técnica, etc. A Figura 9 ilustra os resultados obtidos para o exemplo da heap usando as quatro ferramentas com o tamanho médio dos prompts em palavras, taxa de sucesso dos testes e o número médio de linhas de código geradas.

Tabela para o Problema: Heap

Ferramenta	Número de Ocorrências	Funcionou Corretamente	Média de Palavras nos Prompts	Média de Linhas de Código
Chat Gpt	11	9	150.2	85.545455
Claude	11	10	157.5	121.272727
Copilot	12	12	125.9	70.250000
Gemini	7	5	133.0	82.428571

Figura 9. Tabelas Interativas para Exploração dos Dados dos Experimentos

As duas últimas seções são organizadas por problema e por usuário. Para cada problema, existe uma sub-seção para usuário, onde uma tabela resumida dos experimentos com número de tentativas ou *shots*(T), erros de compilação e execução, tamanho inicial

e total em palavras dos prompts, linhas de código geradas para a última implementação, funcionalidade e qual técnica de prompt que foi utilizada. Toda tabela tem um link para um Google Colab que contém o experimento que foi executado e pode ser reproduzido, com os prompts usados e os resultados obtidos com todos os seus detalhes. São 106 sub-seções, onde cada uma delas contém uma tabela e o detalhamento dos resultados.

Tabela 4. Exemplo de Tabela para o Hash e programador 1 com a comparação entre ferramentas

LMM	Erros Comp.	T	Erros Exec.	1^0 Prompt	Todos os Prompts	Linhas Código	Funcionalidade	Técnica de Prompt
Chat	0	2	1	30	63	153	Correta	Simples
Claude	0	1	0	30	30	205	Correta	Simples
Copilot	0	1	0	30	30	84	Correta	Simples
Gemini	0	1	0	30	30	97	Correta	Simples

7. Conclusões

As LLMs estabeleceram-se como uma tecnologia permanente, mesmo considerando o elevado custo de construção e manutenção dos Datacenters [Khowaja et al. 2024] e as incertezas quanto à disponibilidade futura das opções gratuitas a curto prazo. Este trabalho avaliou o potencial das LLMs para criação de recursos educacionais com visualização gráfica e interatividade. Resultados mostraram que é possível, em questão de minutos, gerar interfaces interativas. Demonstramos a robustez do sistema frente a variações nos prompts, alcançando taxas de sucesso superiores a 80% com descrições de aproximadamente 100 palavras ou 6-8 linhas para cada ferramenta de ensino gerada. Os códigos gerados são concisos, contendo em média 100 linhas, incluindo comentários. Uma contribuição significativa foi a documentação abrangente de todos os prompts e códigos, que podem servir como fonte de dados para análises posteriores, estabelecendo um dataset inicial na área de ferramentas didáticas computacionais com interface. O repositório completo está disponível em [de Viçosa 2024]. Embora a construção incremental das ferramentas com prompts mais simples tenha se mostrado eficaz, observamos algumas instabilidades nas LLMs, o que é esperado devido à natureza emergente da tecnologia. No entanto, Com uma amostragem substancial de mais de 350 ensaios, os resultados evidenciam o potencial desta tecnologia, gerando mais de 250 ferramentas com interfaces didáticas. Trabalhos futuros explorarão, a partir da coleta de feedback de alunos e professores, a avaliação da qualidade de uso das ferramentas, com foco na usabilidade e na experiência do aprendiz, além do seu impacto no aprendizado dos estudantes, por meio de métricas como engajamento e compreensão dos conceitos, dentre outras. Pretende-se investigar também a similaridade entre os códigos gerados das ferramentas. As LLMs também pode ser usadas para melhorar a produção de material didática em domínios específicos juntamente com o Google Colab, como por exemplo a área de arquitetura de computadores [Canesche et al. 2021, Ferreira et al. 2024b, Ferreira et al. 2024a, de Figueiredo et al. 2024].

8. Agradecimentos

Apoio financeiro da Bolsa de Iniciação Científica da FAPEMIG e do CNPq programa Institucional, Bolsa PIBEN Funarbe/Pro-Reitoria de Ensino UFV, FAPEMIG APQ-01577-22, CNPq e Centro de Ciência Exatas e Tecnológicas da UFV.

Referências

[Al-Shetairy et al. 2024] Al-Shetairy, M., Hindy, H., Khattab, D., and Aref, M. M. (2024). Transformers utilization in chart understanding: A review of recent advances & future trends. *arXiv preprint arXiv:2410.13883*.

[Canesche et al. 2021] Canesche, M., Bragança, L., Neto, O. P. V., Nacif, J. A., and Ferreira, R. (2021). Google colab cad4u: Hands-on cloud laboratories for digital design. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE.

[Chen et al. 2023] Chen, B., Zhang, Z., Langrené, N., and Zhu, S. (2023). Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*.

[de Figueiredo et al. 2024] de Figueiredo, G. A., de Souza, E. S., Rodrigues, J. H., Nacif, J. A., and Ferreira, R. (2024). Desenvolvendo ferramentas para ensino de risc-v com python, verilog, matplotlib, svg e chatgpt. *International Journal of Computer Architecture Education*, 13(1):43–52.

[de Viçosa 2024] de Viçosa, U. F. (2024). Material Complementar. `https://colab.research.google.com/drive/1DshdOS61plOIr6ow1jFI6W4EhZzNFBH8?usp=sharing`. [Online].

[Del and Fishel 2022] Del, M. and Fishel, M. (2022). True detective: a deep abductive reasoning benchmark undoable for gpt-3 and challenging for gpt-4. *arXiv preprint arXiv:2212.10114*.

[Ferreira et al. 2024a] Ferreira, R., Canesche, M., Jamieson, P., Neto, O. P. V., and Nacif, J. A. (2024a). Examples and tutorials on using google colab and gradio to create online interactive student-learning modules. *Computer Applications in Engineering Education*, page e22729.

[Ferreira et al. 2024b] Ferreira, R., Sabino, C., Canesche, M., Neto, O. P. V., and Nacif, J. A. (2024b). Aiot tool integration for enriching teaching resources and monitoring student engagement. *Internet of Things*, 26:101045.

[Khowaja et al. 2024] Khowaja, S. A., Khuwaja, P., Dev, K., Wang, W., and Nkenyereye, L. (2024). Chatgpt needs spade (sustainability, privacy, digital divide, and ethics) evaluation: A review. *Cognitive Computation*, pages 1–23.

[Kiesler and Schiffner 2023] Kiesler, N. and Schiffner, D. (2023). Large language models in introductory programming education: Chatgpt's performance and implications for assessments. *arXiv preprint arXiv:2308.08572*.

[Liu et al. 2021] Liu, J., Liu, A., Lu, X., Welleck, S., West, P., Bras, R. L., Choi, Y., and Hajishirzi, H. (2021). Generated knowledge prompting for commonsense reasoning. *arXiv preprint arXiv:2110.08387*.

[Logan IV et al. 2021] Logan IV, R. L., Balažević, I., Wallace, E., Petroni, F., Singh, S., and Riedel, S. (2021). Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*.

[Sato et al. 2024] Sato, Y., Suzuki, A., and Mineshima, K. (2024). Building a large dataset of human-generated captions for science diagrams. In *International Conference on Theory and Application of Diagrams*, pages 393–401. Springer.

[Wei et al. 2022] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

[White et al. 2023] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

[Xu et al. 2023] Xu, X., Tao, C., Shen, T., Xu, C., Xu, H., Long, G., and Lou, J.-g. (2023). Re-reading improves reasoning in language models. *arXiv preprint arXiv:2309.06275*.

[Yang et al. 2023] Yang, Z., Li, L., Wang, J., Lin, K., Azarnasab, E., Ahmed, F., Liu, Z., Liu, C., Zeng, M., and Wang, L. (2023). Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*.

[Yang and Zhu 2024] Yang, Z. and Zhu, Z. (2024). Heuristic question sequence generation based on retrieval augmentation. *Education and Lifelong Development Research*.

[Yao et al. 2024] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. (2024). Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

[Yao et al. 2022] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

[Zala et al. 2023] Zala, A., Lin, H., Cho, J., and Bansal, M. (2023). Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. *arXiv preprint arXiv:2310.12128*.

[Zhou et al. 2022] Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al. (2022). Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.