



# INTEGRANDO O ENSINO SOCRÁTICO COM LLMS: UM FRAMEWORK MULTI-AGENTE PARA INOVAÇÃO NO ENSINO DE PROGRAMAÇÃO

Gabriel de Sapienza Luna

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorio Xexeo

Rio de Janeiro  
Agosto de 2025

INTEGRANDO O ENSINO SOCRÁTICO COM LLMS: UM FRAMEWORK  
MULTI-AGENTE PARA INOVAÇÃO NO ENSINO DE PROGRAMAÇÃO

Gabriel de Sapienza Luna

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO  
ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE  
ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO  
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE  
MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Geraldo Bonorio Xexeo

Aprovada por: Prof. Geraldo Bonorino Xexéo

Prof. Jano Moreira de Souza

Profa. Adriana Santarosa Vivacqua

RIO DE JANEIRO, RJ – BRASIL

AGOSTO DE 2025

Luna, Gabriel de Sapienza

Integrando o Ensino Socrático com LLMs: Um Framework Multi-Agente para Inovação no Ensino de Programação /Gabriel de Sapienza Luna. – Rio de Janeiro: UFRJ/COPPE, 2025.

XIV, 95 p.: il.; 29,7cm.

Orientador: Geraldo Bonorio Xexeo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 89 – 95.

1. Programming Education. 2. Large Language Models. 3. Socratic Method. 4. Supervised Fine-tuning. 5. Direct Preference Optimization. 6. Prompt Engineering. I. Bonorio Xexeo, Geraldo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*Aos que caminharam  
comigo quando o  
caminho ainda não  
existia. Dedico esta  
dissertação.*

# Agradecimentos

A Deus e à vida, pela oportunidade desta jornada.

À minha esposa, **Carina Felix**, meu porto seguro, cuja presença, paciência e apoio incondicional tornaram possível chegar até aqui. Ao meu filho **Kalel**, recém-nascido e já minha maior inspiração é por você que penso no futuro e busco sempre o melhor. Ao meu companheiro de quatro patas, **Bené**: embora não possa ler, estará eternamente lembrado nesta dissertação.

À minha família, base de tudo: à minha mãe **Sheila**, que mora em meu coração; ao meu pai **Carlos Eny**; ao meu irmão gêmeo **Rodrigo Luna** e ao meu irmão mais novo **Rafael Luna**; e ao meu tio **Emanuel**. Sem o apoio de vocês, ao longo de toda a minha trajetória acadêmica, não teria sido possível chegar onde cheguei. Estendo minha gratidão aos meus sogros **Vanilda Felix** e **Marcos Venicius**, e à minha cunhada **Clarice Felix**, pelo incentivo constante e pelas palavras de encorajamento nos momentos mais exigentes.

À memória da minha querida avó **Sueli**, cuja presença segue viva em mim. Seu apoio incondicional durante a graduação e seu exemplo de força e generosidade me guiaram até aqui. Que este trabalho a honre e a encha de orgulho, onde quer que esteja.

Aos amigos de vida **Luiz Felipe Martinez** e **Kauwa Muller** amigos quase irmãos, para a vida toda. Obrigado pela lealdade, pela presença nos dias difíceis e por celebrarem cada conquista comigo.

Aos amigos de faculdade **Rodrigo Paiva**, **Pedro Kleiz**, **Chris Chinedoize** e **Gustavo** companheiros desde o início da graduação em Ciência da Computação. Carrego nossa amizade como fraternidade: vocês foram abrigo, estímulo e alegria em cada etapa.

Aos professores **Geraldo Bonorino Xexéo** e **Jano Moreira de Souza**, pelas oportunidades, pela confiança e pelas experiências transformadoras no **Programa de Engenharia de Sistemas e Computação (PESC/COPPE/UFRJ)**. Sou grato também a todo o corpo docente e aos servidores do PESC, cuja dedicação viabiliza o nosso crescimento acadêmico e humano.

À melhor equipe da **CAPGOV equipe de ETL**, pela parceria, pelas conversas técnicas que viraram ideias e pelas ideias que viraram resultados. Cada contribuição

deixou marcas concretas neste trabalho.

À **Universidade Federal do Rio de Janeiro**, à **COPPE** e ao **PESC**, pelo ambiente de excelência e por todo o suporte institucional ao longo do mestrado. À **CAPES**, pelo apoio à formação acadêmica no país.

Por fim, a todos que, de alguma forma com um conselho, uma revisão, um café, um gesto de amizade contribuíram para que este projeto se concretizasse, deixo aqui o meu muito obrigado. Este título é nosso.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

## INTEGRANDO O ENSINO SOCRÁTICO COM LLMS: UM FRAMEWORK MULTI-AGENTE PARA INOVAÇÃO NO ENSINO DE PROGRAMAÇÃO

Gabriel de Sapienza Luna

Agosto/2025

Orientador: Geraldo Bonorio Xexeo

Programa: Engenharia de Sistemas e Computação

Esta dissertação apresenta um framework educacional que combina Grandes Modelos de Linguagem (LLMs) com o método socrático, buscando potencializar o ensino de programação por meio de diálogos mediados por Inteligência Artificial (IA). O objetivo central foi explorar a capacidade de LLMs compactos de internalizar e reproduzir uma abordagem pedagógica socrática eficaz e reflexiva. Para isso, foi realizada a geração de um dataset sintético com cerca de mil pares de interação aluno-professor, utilizados no processo de ajuste fino supervisionado (SFT) e na otimização direta por preferência (DPO) dos modelos LLaMA 3 e Qwen, empregando a técnica LoRA com rank fixo igual a 16. Adicionalmente, foi adotada uma estratégia criteriosa de engenharia de prompt, visando maximizar a clareza e a eficácia pedagógica das respostas geradas. A avaliação das respostas produzidas pelos modelos foi conduzida automaticamente com o auxílio do GPT-4o, considerando critérios como qualidade pedagógica, clareza e fidelidade ao método socrático. Os resultados indicaram que o método híbrido de SFT seguido de DPO proporcionou melhorias significativas no alinhamento pedagógico e na eficácia das respostas, com destaque especial ao modelo LLaMA 3.2-3B, que demonstrou excelente desempenho socrático tanto no treinamento quanto em uma validação externa com questões inéditas. Como trabalho futuro, sugere-se investigar a influência da variação dos ranks LoRA sobre a performance dos modelos, bem como avaliar a eficiência da quantização (QLoRA) quanto ao tempo de treinamento e qualidade pedagógica das respostas, visando a otimização adicional do processo educacional mediado por LLMs.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

## INTEGRANDO O ENSINO SOCRÁTICO COM LLMS: UM FRAMEWORK MULTI-AGENTE PARA INOVAÇÃO NO ENSINO DE PROGRAMAÇÃO

Gabriel de Sapienza Luna

August/2025

Advisor: Geraldo Bonorio Xexeo

Department: Systems Engineering and Computer Science

This dissertation proposes an educational framework that integrates Large Language Models (LLMs) with the Socratic method to enhance the teaching of programming through dialogues mediated by Artificial Intelligence (AI). The main objective was to investigate the capability of compact LLMs to internalize and effectively reproduce a reflective, pedagogical approach grounded in Socratic techniques. For this purpose, a synthetic dataset comprising approximately one thousand student-teacher interaction pairs was created and utilized in supervised fine-tuning (SFT) and Direct Preference Optimization (DPO) of the LLaMA 3 and Qwen models, employing the LoRA technique with a fixed rank of 16. Additionally, a deliberate prompt engineering strategy was adopted to maximize clarity and pedagogical effectiveness of the generated responses. Automatic evaluation of the generated interactions was conducted using the GPT-4o model, considering criteria such as pedagogical quality, clarity, and adherence to the Socratic method. Results indicated that the hybrid approach, combining SFT followed by DPO, led to significant improvements in the pedagogical alignment and efficacy of model-generated responses. Notably, the LLaMA 3.2-3B model demonstrated outstanding Socratic performance both during training and in external validation involving previously unseen programming questions. As future work, it is recommended to investigate the influence of varying LoRA ranks on model performance and to evaluate the efficiency of quantization methods (QLoRA) regarding training time and pedagogical quality of responses, aiming for further optimization of AI-mediated educational processes.



# Sumário

<b>Lista de Figuras</b>	<b>xii</b>
<b>Lista de Tabelas</b>	<b>xiv</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Contextualização do Tema . . . . .	1
1.2 Problema de Pesquisa e Objetivos . . . . .	2
1.2.1 Formulação do Problema . . . . .	2
1.2.2 Objetivos . . . . .	2
1.2.3 Justificativa da Proposta . . . . .	3
1.3 Relevância e Contribuições . . . . .	4
1.4 Estrutura da Dissertação . . . . .	5
<b>2 Revisão de Literatura</b>	<b>6</b>
2.1 Fundamentos do Ensino Socrático . . . . .	6
2.2 Aplicações do Ensino Socrático em Programação . . . . .	7
2.2.1 O Método Socrático na Aprendizagem de Programação . . . . .	7
2.2.2 Aplicações em Ambientes Educacionais . . . . .	7
2.2.3 Benefícios e Desafios do Ensino Socrático na Programação . . . . .	8
2.2.4 Análise do estudo: Socratic Questioning of Novice Debuggers: A Benchmark Dataset and Preliminary Evaluations . . . . .	8
2.2.5 Análise do estudo <i>Improving Socratic Question Generation using Data Augmentation and Preference Optimization</i> . . . . .	10
2.2.6 Análise do estudo <i>Instruct, Not Assist: LLM-based Multi-Turn Planning and Hierarchical Questioning for Socratic Code De- bugging</i> . . . . .	13
2.2.7 Considerações Finais sobre o Ensino Socrático . . . . .	15
2.3 LLMs e Educação . . . . .	16
2.3.1 O Papel dos LLMs na Educação . . . . .	16
2.3.2 Análise do estudo <i>SocraticLM: Exploring Socratic Personali- zed Teaching with Large Language Models</i> . . . . .	17

2.3.3	Análise do estudo <i>Boosting Large Language Models with Socratic Method for Conversational Mathematics Teaching</i> . . .	19
2.3.4	Benefícios e Desafios dos LLMs na Educação . . . . .	21
2.3.5	Perspectivas Futuras para LLMs na Educação . . . . .	22
2.4	Conclusão . . . . .	23
2.5	Técnicas de Engenharia de Prompt . . . . .	23
2.5.1	Técnicas de Engenharia de Prompt . . . . .	24
2.6	Retrieval-Augmented Generation (RAG) . . . . .	29
2.6.1	Conceitos Fundamentais de RAG . . . . .	29
2.6.2	Arquitetura Básica de um Sistema RAG . . . . .	30
2.7	Data Augmentation . . . . .	31
2.8	Fine-Tuning e LoRA Adapters . . . . .	32
2.9	Sistemas Multi-Agente em IA . . . . .	36
2.9.1	Arquiteturas . . . . .	37
2.10	Otimização Direta de Preferências (DPO) . . . . .	38
2.10.1	Formulação Matemática da DPO . . . . .	39
2.10.2	Exemplo Numérico . . . . .	39
2.10.3	Aplicação da DPO no Ensino Socrático . . . . .	41
2.11	Temperatura na Geração de Texto com LLMs: Mecanismos e Impli- cações . . . . .	41
<b>3</b>	<b>Metodologia</b>	<b>44</b>
3.1	Descrição Geral do Framework . . . . .	44
3.1.1	Motivação e Contexto . . . . .	44
3.1.2	Estrutura do Framework . . . . .	44
3.2	Detalhamento dos Componentes . . . . .	47
3.2.1	Data Augmentation . . . . .	47
3.2.2	Recomendação de Questão . . . . .	47
3.2.3	Fluxo socrático . . . . .	48
3.2.4	Criação de questões . . . . .	49
3.2.5	Correção automática de questões . . . . .	49
3.3	Procedimentos Experimentais e Ferramentas . . . . .	51
3.3.1	Metodologia Experimental . . . . .	51
3.3.2	Modelos Avaliados . . . . .	53
3.3.3	Avaliação Experimental . . . . .	54
3.3.4	Análise das Ferramentas Utilizadas . . . . .	55
3.3.5	Discussão e Escolha do Modelo . . . . .	56
<b>4</b>	<b>Avaliação Experimental do Framework</b>	<b>57</b>
4.1	Avaliação Experimental do Framework . . . . .	57

4.1.1	Distribuição de Notas por Temperatura . . . . .	64
4.1.2	Preparação do Conjunto para Fine-Tuning . . . . .	65
4.1.3	Modelos Selecionados para Fine-Tuning . . . . .	66
4.1.4	Distribuição e Análise Inicial das Notas Atribuídas pelo GPT-4o	67
4.1.5	Expansão e Qualificação do Dataset: Seleção, Melhoria e Ge- ração de Pares para DPO . . . . .	69
4.1.6	Comparação Geral dos Modelos Avaliados . . . . .	71
4.1.7	Avaliação Comparativa dos Modelos após SFT seguido de DPO	73
4.2	Validação da Generalização do Modelo em Domínios Inéditos . . . . .	75
4.2.1	Validação Externa dos Modelos . . . . .	77
4.3	Plano de Amostragem e Validação Humana . . . . .	78
4.3.1	Métrica de Avaliação: Erro Absoluto Médio e Acurácia Pon- derada . . . . .	80
<b>5</b>	<b>Conclusões e Trabalhos Futuros</b>	<b>83</b>
5.1	Síntese dos Resultados . . . . .	83
5.2	Originalidade e Contribuições . . . . .	84
5.3	Limitações . . . . .	85
5.4	Propostas para Trabalhos Futuros . . . . .	86
	<b>Referências Bibliográficas</b>	<b>89</b>

# Lista de Figuras

- 2.1 Processo de *Regular Finetuning* (Full Fine-Tuning), adaptado de RASCHKA (2023). O processo é dividido em três etapas sequenciais: (1) **Forward pass inicial** com o modelo pré-treinado original, onde os inputs ( $x$ ) são processados pelos pesos iniciais ( $W$ ), gerando embeddings intermediários ( $h$ ); (2) **Atualização dos pesos via backpropagation**, onde é calculado o vetor de atualização de pesos ( $\Delta W$ ) com o objetivo de adaptar o modelo à tarefa-alvo; (3) **Forward pass com o modelo atualizado**, utilizando os pesos ajustados ( $W' = W + \Delta W$ ) para produzir saídas mais adaptadas e precisas à tarefa específica. . . . . 33
- 2.2 Ilustração conceitual do mecanismo do LoRA, adaptado de RASCHKA (2023). O modelo pré-treinado fornece os pesos originais  $W$  (caixa cinza), e o LoRA aprende duas matrizes de baixo posto  $W_A$  e  $W_B$  (em roxo, de dimensão  $r$  muito menor) que juntas formam a atualização  $\Delta W = W_A W_B$ . Na inferência, o resultado da camada é a soma do efeito dos pesos originais congelados com a contribuição ajustada pelo LoRA. . . . . 35
- 2.3 Efeito da Temperatura ( $T$ ) na Distribuição de Probabilidade Softmax sobre Tokens Candidatos. Temperaturas baixas (ex:  $T=0.5$ ) concentram a probabilidade no token mais provável, enquanto temperaturas altas (ex:  $T=2.0$ ) tornam a distribuição mais uniforme. . . . . 42
- 3.1 Diagrama de Componentes UML representando a arquitetura funcional do sistema baseado em agentes. O Agente Professor é responsável por criar exercícios, gerar questões automaticamente e aprimorar o comportamento do agente socrático. Já o Agente Aluno contempla o agente socrático, a recomendação e a correção de questões. Ambos os agentes são acionados pelo Usuário, que identifica qual funcionalidade deseja acessar no sistema. . . . . 45

3.2	Diagrama de sequência representando o ensino socrático com erro lógico na função fatorial. O aluno recebe um exercício com enunciado e casos de teste, tenta resolver, obtém um resultado diferente do esperado, e busca auxílio com o agente professor, que o conduz por meio de perguntas reflexivas até encontrar a solução correta. . . . .	48
3.3	Diagrama de Sequência representando o fluxo do sistema RAG para recomendação de questões. . . . .	49
3.4	Fluxo socrático de interação com agentes de IA . . . . .	50
3.5	Diagrama de sequência representando os fluxos de criação de questões no módulo do professor/instrutor. No fluxo de geração automática, o professor fornece o nome, a categoria e a descrição do problema, e o sistema utiliza um LLM para gerar o enunciado refinado, o código Python e os casos de teste. No fluxo de criação manual, o professor fornece também o código, e o sistema gera automaticamente os casos de teste a partir desse código. Ambos os fluxos resultam no armazenamento da questão em um banco de dados ou arquivo JSON. . . . .	51
3.6	Diagrama de sequência da correção automática de questões com participação de um LLM avaliador. Após o envio do código Python pelo aluno, o sistema identifica a questão correspondente via embeddings e repassa os dados para o LLM, que executa o código com base nos casos de teste e compara os resultados com o gabarito. A análise de estilo é realizada com o Flake8. A nota final é composta pela execução funcional e pela conformidade de estilo, sendo então retornada ao aluno com feedback detalhado. . . . .	52
3.7	Diagrama de sequência UML simplificado da interação entre agentes Professor e Aluno. . . . .	53
3.8	Distribuição percentual das respostas para cada modelo LLM. . . . .	55
4.1	Distribuição absoluta das notas de socratismo por temperatura de geração. . . . .	65
4.2	Distribuição percentual das notas por temperatura de geração. . . . .	66
4.3	Distribuição percentual das notas por modelo e pipeline. . . . .	75

# Lista de Tabelas

3.1	Resultados das interações por modelo LLM . . . . .	54
3.2	Custos médios por 1 milhão de tokens (USD) dos modelos avaliados (valores de saída ou unificados) . . . . .	56
4.1	Escala de avaliação do nível de Socratismo em respostas de LLMs educacionais . . . . .	60
4.2	Distribuição das notas de socratismo atribuídas pelo GPT-4o-mini . .	66
4.3	Modelos selecionados para o fine-tuning supervisionado . . . . .	67
4.4	Parâmetros adicionados pelo LoRA por modelo e rank . . . . .	67
4.5	Distribuição absoluta das notas atribuídas pelo GPT-4o-mini para cada modelo após correção de anomalias, incluindo GPT-4o baseline .	68
4.6	Distribuição percentual das notas atribuídas pelo GPT-4o para cada modelo, incluindo GPT-4o-mini baseline . . . . .	68
4.7	Comparação geral dos modelos avaliados após aplicação do SFT e DPO	71
4.8	Distribuição percentual das notas atribuídas pelo GPT-4o-mini para cada modelo após SFT (avaliação automática) . . . . .	71
4.9	Métricas dos modelos LLaMA 3.2 após SFT seguido de DPO . . . . .	74
4.10	Distribuição absoluta das notas atribuídas pelo GPT-4o-mini aos mo- delos LLaMA 3.2 após SFT seguido de DPO . . . . .	74
4.11	Resultados da validação externa das respostas geradas pelo modelo LLaMA 3.2-3B SFT-DPO . . . . .	77
4.12	População original (1087) vs. amostra usada na validação (295) por classe de nota. . . . .	79
4.13	Matriz de Confusão para a classificação binária. . . . .	82
4.14	Métricas de desempenho por classe. . . . .	82

# Capítulo 1

## Introdução

### 1.1 Contextualização do Tema

A demanda por profissionais de tecnologia da informação no Brasil cresce exponencialmente: estudos indicam que o país enfrenta um déficit de até 530 mil profissionais qualificados na área GOOGLE (2022). Paralelamente, a escassez de docentes especializados em Tecnologia da Informação (TI) ameaça desacelerar a formação de novos talentos, gerando impactos negativos para o ecossistema tecnológico nacional SEPROSC (2023). Esse cenário evidencia um desafio estrutural: instituições de ensino têm dificuldade em equilibrar a demanda por apoio individualizado e a disponibilidade limitada de professores, especialmente em disciplinas práticas como programação, cujo aprendizado depende fortemente da resolução e correção de exercícios de codificação. Docentes frequentemente sobrecarregam-se ao analisar código por código, identificar erros e fornecer feedback personalizado, uma tarefa que exige tempo e expertise técnica.

Nesse contexto, a inteligência artificial (IA) surge como uma aliada promissora para automatizar tarefas educacionais repetitivas. Plataformas como Beecrowd, Codeforces e HackerRank já utilizam sistemas automatizados para correção de exercícios, mas seu foco é predominantemente avaliativo, não pedagógico. Avanços recentes em *Large Language Models* (LLMs), como o GPT-4, DeepSeek, LLAMA e entre diversos outros, ampliam esse potencial, permitindo não apenas a geração de códigos, mas também a mediação de processos de aprendizagem por meio de diálogos estruturados. Aqui, o método socrático, estratégia em que o professor guia o aluno com perguntas reflexivas, incentivando-o a superar sua zona de desenvolvimento proximal (ZDP), ganha relevância. Pesquisas exploram LLMs como tutores capazes de replicar essa abordagem em matemática, adaptando questionamentos à sequência lógica de resolução de problemas. Na programação, porém, os desafios são mais complexos: erros de sintaxe, falhas de lógica e dificuldades de abstração

exigem que os modelos compreendam não apenas o código, mas também a intenção e o raciocínio do estudante.

Embora estudos preliminares sugiram que LLMs podem atuar como tutores virtuais em programação, há lacunas críticas. A sensibilidade pedagógica necessária para personalizar diálogos socráticos, considerando o nível de conhecimento, erros recorrentes e progressão cognitiva do aluno, ainda não foi plenamente explorada.

Esta pesquisa propõe-se, portanto, a investigar como LLMs podem ser calibrados para integrar correção técnica e suporte cognitivo adaptativo, oferecendo um caminho viável para mitigar a escassez de docentes e democratizar o acesso ao aprendizado personalizado de programação no Brasil.

## 1.2 Problema de Pesquisa e Objetivos

### 1.2.1 Formulação do Problema

Apesar do potencial dos *LLMs* para automatizar tarefas educacionais, sua aplicação no ensino de programação enfrenta uma lacuna crítica: a **incapacidade de replicar a sensibilidade pedagógica de um instrutor humano** durante a resolução de exercícios. Embora plataformas como Beecrowd e HackerRank ofereçam correção automática de código, seu enfoque restrito à funcionalidade do programa e ignora dimensões essenciais do aprendizado, como:

- **Diagnóstico de erros conceituais.**
- **Adaptação ao nível cognitivo do aluno** (ex.: identificar se um erro é pontual ou revela uma lacuna de abstração).
- **Estímulo à autorreflexão** por meio de diálogos guiados, conforme proposto pelo método socrático.

Essa limitação é agravada pela escassez de docentes qualificados em TI. Assim, o problema central desta pesquisa é:

*Como desenvolver um framework pedagógico que integre LLMs ao método socrático para fornecer feedback adaptativo em programação, capacitando estudantes a superar lacunas cognitivas de forma autônoma?*

### 1.2.2 Objetivos

#### Objetivo Geral

Projetar um framework que combine LLMs e método socrático para oferecer feedback personalizado em programação, considerando erros técnicos, nível de expertise



e progressão cognitiva do estudante.

### **Objetivos Específicos**

- Analisar criticamente a literatura sobre ensino socrático mediado por IA e os desafios na aplicação de LLMs em contextos educacionais de programação.
- Investigar estratégias de questionamento socrático adaptadas à resolução de problemas computacionais.
- Propor uma arquitetura multi-agente que simule a interação entre aluno e professor utilizando LLMs.
- Avaliar experimentalmente o framework proposto, comparando a qualidade das respostas geradas por diferentes modelos e configurações (ex: temperatura, fine-tuning).

### **1.2.3 Justificativa da Proposta**

A relevância desta dissertação é dupla, incidindo sobre desafios críticos tanto no campo social quanto no tecnológico. Socialmente, o trabalho responde diretamente à crescente demanda por profissionais de TI e à simultânea escassez de tutoria especializada em programação. Ao propor uma solução de tutoria inteligente, escalável e acessível, o projeto visa democratizar o acesso ao aprendizado de qualidade, fomentando a inclusão e a autonomia dos estudantes.

A viabilidade desta proposta reside em sua inovação técnica. Em vez de abordagens genéricas, esta pesquisa propõe um framework que integra um pipeline de técnicas de ponta da Inteligência Artificial. Por meio de data augmentation, cria-se um volume de dados pedagógicos de alta qualidade; com fine-tuning supervisionado (SFT) e adaptadores LoRA, modelos compactos são ensinados a "pensar" de forma socrática; e com a Otimização Direta por Preferências (DPO), esse comportamento é refinado para se aproximar da complexidade e sensibilidade de um tutor humano.

Ao empregar uma arquitetura multiagente e fomentar a metacognição, o framework não apenas contribui para o avanço na simulação de interações pedagógicas, mas também se alinha às diretrizes contemporâneas da educação baseada em competências. Em suma, a dissertação se justifica por sua proposta de usar engenharia de IA sofisticada como uma ponte para resolver um gargalo fundamental na educação tecnológica.

## 1.3 Relevância e Contribuições

Este trabalho se posiciona na fronteira entre Inteligência Artificial e Educação, apresentando uma contribuição original e de alta relevância para o ensino de programação. A pesquisa não apenas demonstra a viabilidade de se adaptar o método socrático a interações mediadas por LLMs, mas propõe e valida um framework completo e de baixo custo computacional para o treinamento de tutores de IA eficazes e acessíveis. As principais contribuições deste estudo são:

- **Desenvolvimento de um pipeline integrado para a geração de dados e treinamento de modelos:** Foi proposto um fluxo automatizado que utiliza técnicas de *data augmentation* para criar um dataset de alta qualidade de diálogos socráticos, empregado subsequentemente para o ajuste fino de modelos de linguagem compactos e de código aberto.
- **Análise sistemática e comparação de métodos de ajuste fino (Fine-Tuning):** O estudo implementa e compara rigorosamente diferentes estratégias de alinhamento, incluindo o ajuste fino supervisionado (SFT) com LoRA e a Otimização Direta por Preferências (DPO), identificando o pipeline sequencial SFT+DPO como o mais eficaz para maximizar a qualidade socrática das interações.
- **Metodologia de avaliação automática, escalável e objetiva:** Para superar a necessidade de anotação manual, foi implementada uma metodologia de avaliação automática utilizando um LLM de ponta (GPT-4o) como juiz, com base em uma rubrica detalhada. Esta abordagem permitiu a análise quantitativa de milhares de interações de forma consistente e escalável.
- **Validação da eficácia de modelos compactos para tarefas pedagógicas complexas:** A principal contribuição prática é a demonstração empírica de que modelos com até 3 bilhões de parâmetros, quando devidamente ajustados, podem atingir um nível de excelência em interações socráticas comparável ao de modelos muito maiores, oferecendo uma solução de alta performance e economicamente viável para plataformas educacionais.

## 1.4 Estrutura da Dissertação

Esta dissertação está organizada em cinco capítulos, descritos a seguir:

- **Capítulo 1 Introdução:** apresenta o contexto do problema, os objetivos da pesquisa, a justificativa da proposta e as contribuições esperadas, além desta descrição da estrutura do trabalho.
- **Capítulo 2 Revisão de Literatura:** discute os fundamentos do ensino socrático aplicado à programação, os avanços recentes em Large Language Models (LLMs) voltados para a educação, e técnicas relacionadas como engenharia de prompts, RAG (Retrieval-Augmented Generation), *data augmentation*, *fine-tuning* com LoRA, sistemas multi-agentes e otimização direta por preferências (DPO), com base em estudos recentes da literatura.
- **Capítulo 3 Metodologia:** descreve o *framework* proposto, detalha os componentes que integram o sistema (como geração de dados, fluxo socrático e correção automatizada), e apresenta os procedimentos experimentais adotados, incluindo os modelos avaliados e as ferramentas utilizadas.
- **Capítulo 4 Avaliação Experimental do Framework:** apresenta e analisa os resultados obtidos com os testes realizados, comparando o desempenho de diferentes LLMs no contexto educacional proposto e discutindo suas implicações pedagógicas, técnicas e econômicas.
- **Capítulo 5 Conclusões e Trabalhos Futuros:** sintetiza os principais resultados da pesquisa, discute suas limitações e propõe direções para estudos futuros que possam aprofundar e expandir o uso de LLMs no ensino socrático de programação.

# Capítulo 2

## Revisão de Literatura

Este capítulo apresenta uma revisão da literatura sobre os principais conceitos, técnicas e estudos relevantes relacionados à utilização de métodos pedagógicos socráticos, *Large Language Models* (LLMs) e técnicas avançadas associadas, como *Engenharia de Prompt*, *Retrieval-Augmented Generation* (RAG), *Data Augmentation*, *Fine-Tuning* e otimização por adaptadores *LoRA*, bem como abordagens baseadas em Sistemas *Multi-Agente* e *Otimização Direta de Preferências* (DPO). O objetivo desta revisão é contextualizar o leitor sobre as fundamentações teóricas e as aplicações práticas que embasam o desenvolvimento da presente pesquisa, destacando as principais contribuições recentes e apontando tendências e desafios emergentes nessas áreas.

Inicialmente, serão abordados os fundamentos e aplicações do ensino socrático, especialmente no contexto da programação, destacando estudos recentes que exploram sua eficácia em ambientes educacionais assistidos por IA. Em seguida, será explorada a integração de *Large Language Models* em contextos educacionais, enfatizando os benefícios, desafios e perspectivas futuras desse tipo de aplicação. A seguir, serão detalhadas técnicas fundamentais para otimizar a interação e o desempenho desses modelos, como *Engenharia de Prompt* e *RAG*, seguidas por uma discussão sobre a importância do *Data Augmentation* para superar limitações relacionadas à escassez de dados rotulados. Por fim, serão analisadas técnicas avançadas de ajuste fino dos LLMs, incluindo os adaptadores *LoRA* e estratégias inovadoras como a *Otimização Direta de Preferências* (DPO), além de uma discussão sobre sistemas *multi-agente* em IA, que representam uma abordagem emergente para ampliar ainda mais as capacidades desses modelos em tarefas complexas e interativas.

### 2.1 Fundamentos do Ensino Socrático

O ensino socrático, inspirado no método dialógico desenvolvido por Sócrates (470-399 a.C.), caracteriza-se pelo uso estratégico de perguntas para estimular a re-

flexão crítica e a construção autônoma de conhecimento PAUL e ELDER (2006). Diferentemente de abordagens expositivas, essa metodologia coloca o aluno como protagonista, incentivando-o a elaborar hipóteses, justificar ideias e revisar conclusões. Na educação em programação, essa abordagem revela-se particularmente relevante para desenvolver habilidades de depuração e pensamento algorítmico.

## 2.2 Aplicações do Ensino Socrático em Programação

O ensino de programação apresenta desafios pedagógicos significativos, uma vez que os estudantes frequentemente enfrentam dificuldades na compreensão dos conceitos abstratos e na resolução de problemas computacionais AL-HOSSAMI *et al.* (2024). Nesse contexto, o **método socrático** tem se mostrado uma abordagem eficaz para promover a aprendizagem ativa e o desenvolvimento do pensamento crítico em estudantes de computação AL-HOSSAMI *et al.* (2023a).

### 2.2.1 O Método Socrático na Aprendizagem de Programação

A pedagogia socrática baseia-se na interação dialógica entre o professor e o aluno, onde o instrutor não fornece respostas diretas, mas utiliza perguntas estratégicas para guiar o estudante na construção do conhecimento AL-HOSSAMI *et al.* (2023b). Essa abordagem tem sido empregada em cursos introdutórios de programação, especialmente para auxiliar estudantes a identificar e corrigir erros em seus códigos. Segundo pesquisas recentes, o uso de tutoria socrática para depuração de código melhora a retenção de conceitos fundamentais e estimula a autonomia na resolução de problemas AL-HOSSAMI *et al.* (2023b).

Em estudos conduzidos por AL-HOSSAMI *et al.* (2023a), foi demonstrado que a interação socrática melhora a compreensão da estrutura lógica dos programas. Por exemplo, ao invés de apontar diretamente um erro em um código, um instrutor pode perguntar: *O que acontece se você executar essa função para um valor menor que zero?* ou *Quais são as condições de parada desse laço?* Essas perguntas incentivam o estudante a refletir sobre a implementação e desenvolver uma abordagem analítica para solucionar erros AL-HOSSAMI *et al.* (2023a).

### 2.2.2 Aplicações em Ambientes Educacionais

A integração do método socrático em plataformas de ensino de programação tem sido explorada em diversos contextos, incluindo cursos online, tutores virtuais e

sistemas de aprendizado automático DING *et al.* (2024a). Estudos apontam que sistemas baseados em modelos de linguagem podem ser programados para atuar como instrutores socráticos, fazendo perguntas que levam os estudantes a compreenderem melhor seus erros e a refinarem suas soluções KUMAR e LAN (2024a).

Uma aplicação interessante desse método ocorre na depuração de código. Em uma abordagem tradicional, um aluno que comete um erro recebe uma mensagem direta do compilador ou do ambiente de desenvolvimento integrado (*IDE*) informando sobre a falha. No entanto, ao utilizar uma abordagem socrática, um sistema pode gerar perguntas guiadas para ajudar o estudante a encontrar e corrigir o problema por conta própria. Essa técnica foi testada em um conjunto de 38 exercícios de programação, nos quais os alunos corrigiram seus próprios códigos baseando-se exclusivamente nas perguntas feitas pelo instrutor virtual AL-HOSSAMI *et al.* (2023a).

### 2.2.3 Benefícios e Desafios do Ensino Socrático na Programação

Os benefícios do método socrático no ensino de programação incluem:

- **Maior engajamento:** A necessidade de responder perguntas desafiadoras incentiva os estudantes a participarem ativamente do processo de aprendizado AL-HOSSAMI *et al.* (2023b).
- **Melhoria na capacidade de depuração:** Estudantes que utilizam esse método demonstram maior habilidade na identificação e correção de erros (AL-HOSSAMI *et al.*, 2023a).
- **Aprimoramento do raciocínio lógico:** O questionamento constante leva à construção progressiva do conhecimento e ao desenvolvimento de um pensamento computacional mais estruturado (VYGOTSKY, 1978).

Entretanto, existem desafios na implementação desse modelo, como a curva de aprendizado para os instrutores, que precisam reformular seu papel na sala de aula, além da necessidade de adaptação dos sistemas de ensino para suportar interações mais abertas e baseadas em diálogo (KUMAR e LAN, 2024a).

### 2.2.4 Análise do estudo: Socratic Questioning of Novice Debuggers: A Benchmark Dataset and Preliminary Evaluations

No artigo Socratic Questioning of Novice Debuggers: A Benchmark Dataset and Preliminary Evaluations, os autores investigam a aplicação do questionamento

socrático como estratégia pedagógica para auxiliar programadores iniciantes na depuração de código. Diante da crescente demanda por ensino de programação e da escassez de instrutores qualificados, o estudo propõe a utilização de modelos de *Large Language Models* (LLMs) para atuar como agentes instrutores, conduzindo estudantes por meio de diálogos estruturados que os levam a identificar e corrigir erros de forma independente (AL-HOSSAMI *et al.*, 2023b).

Os autores introduzem um conjunto de dados anotado manualmente contendo diálogos socráticos entre um instrutor e um estudante que tenta corrigir erros em seu código. O objetivo principal é maximizar a retenção do conhecimento e promover uma aprendizagem mais autônoma e reflexiva, em oposição a abordagens tradicionais que oferecem respostas diretas. No experimento, os pesquisadores avaliaram a capacidade de modelos de linguagem da família GPT de gerar intervenções socráticas eficazes para depuração de código. Os resultados demonstraram que o GPT-4 superou significativamente o GPT-3.5 em termos de precisão e recall, mas ainda ficou abaixo da performance humana, evidenciando a necessidade de aprimoramentos na modelagem dessas interações (AL-HOSSAMI *et al.*, 2023b).

A pesquisa se fundamenta na teoria da zona de desenvolvimento proximal (ZDP) de Vygotsky, que sugere que a aprendizagem é mais eficaz quando o aluno recebe suporte para alcançar um nível de compreensão ligeiramente além de sua capacidade atual (QUINTANA *et al.*, 2004). O estudo também reforça a importância do scaffolding educacional, conceito no qual o ensino é estruturado para guiar gradualmente o estudante até a autonomia no aprendizado (WOOD *et al.*, 1976). O trabalho se diferencia de abordagens anteriores ao apresentar um conjunto de dados específico para questionamento socrático na depuração de código, além de avaliar a viabilidade da automação desse processo com LLMs.

Apesar das contribuições relevantes, os autores destacam que persistem desafios, como a geração de respostas pouco pertinentes ou excessivamente objetivas, que podem minimizar a capacidade de reflexão e reduzir o envolvimento ativo do aprendiz no processo de depuração. Nesse sentido, enfatiza-se a urgência de avanços no ajuste de modelos para torná-los mais coerentes com práticas pedagógicas eficazes. Essa calibragem adequada inclui a personalização do nível de orientação oferecida ao aprendiz, de modo a evitar que as intervenções se tornem superficiais ou sobrecarreguem o estudante com instruções imediatas. Dessa forma, o artigo contribui para o debate acerca do uso de LLMs como auxiliares educacionais, reforçando a importância de unir fundamentos teóricos e práticas instrucionais sólidas para, de fato, potencializar a aprendizagem em programação.

### 2.2.5 Análise do estudo *Improving Socratic Question Generation using Data Augmentation and Preference Optimization*

O estudo KUMAR e LAN (2024b) investiga de LLMs para aprimorar a geração automática de questões socráticas, um método amplamente reconhecido na educação por estimular o pensamento crítico e a aprendizagem ativa. Embora os LLMs tenham demonstrado potencial na criação de perguntas instrucionais, abordagens tradicionais baseadas em *prompting* frequentemente produzem perguntas inadequadas, como questões irrelevantes, repetitivas, muito diretas ou prematuras. Essas limitações comprometem a eficácia do método socrático, que se baseia na construção progressiva do raciocínio do aluno, guiando-o na resolução de problemas sem fornecer respostas diretas.

Os autores de KUMAR e LAN (2024b) identificaram que os modelos proprietários de ponta, como GPT-3.5 e GPT-4, embora eficazes, podem gerar perguntas que desviam o aluno do problema real ou oferecem soluções de maneira implícita, o que reduz a capacidade do estudante de desenvolver seu próprio raciocínio. Assim, o objetivo central do estudo é desenvolver um método de otimização que direcione os LLMs para gerar questões socráticas mais alinhadas com os princípios pedagógicos desse método. Para isso, os autores propõem um modelo baseado em aprendizado por reforço com feedback de IA (*Reinforcement Learning with AI Feedback* - RLAIIF), estruturado em duas fases: *Data Augmentation* e *Preference Optimization*.

#### Abordagem proposta

A abordagem apresentada no estudo combina duas técnicas principais para treinar modelos de linguagem a evitar perguntas inválidas e priorizar questões pedagogicamente adequadas.

##### 1. Data Augmentation (Aumento de Dados)

Para melhorar a capacidade dos LLMs na distinção entre perguntas válidas e inválidas, os autores introduzem um método de aumento de dados que gera exemplos sintéticos de perguntas inadequadas. Essas perguntas são divididas em quatro categorias específicas:

- **Perguntas irrelevantes:** desviam a atenção do estudante do problema central, podendo causar confusão.
- **Perguntas repetidas:** já foram levantadas anteriormente no diálogo, não acrescentando valor adicional ao aprendizado.



- **Perguntas diretas:** revelam o erro no código do estudante de forma explícita, reduzindo o esforço cognitivo necessário para a descoberta do problema.
- **Perguntas prematuras:** sugerem uma correção antes que o estudante compreenda a origem do erro, prejudicando a construção lógica do conhecimento.

Para gerar tais perguntas, o GPT-4 é utilizado em um esquema de *few-shot prompting*, em que exemplos de perguntas inválidas são fornecidos ao modelo para guiá-lo na criação de novas perguntas com as mesmas características. Em seguida, uma segunda instância do GPT-4 realiza uma verificação de consistência, classificando as perguntas geradas em seis categorias: as quatro categorias inválidas mencionadas, além de duas adicionais *bom* (quando a pergunta é válida) e *incorreto* (quando a pergunta não se relaciona com o contexto da conversa). Perguntas classificadas como *boas* ou *incorretas* são removidas do conjunto de dados aumentado, garantindo a qualidade dos exemplos utilizados para treinar o modelo.

O resultado desse processo é a construção de um *dataset* de preferências contendo 2.500 pares de perguntas válidas e inválidas, que servirá como base para a etapa seguinte de otimização.

## 2. Otimização por Preferência (*Preference Optimization*)

Após a criação do *dataset* de preferências, os autores utilizam a técnica de *Direct Preference Optimization* (DPO) para ajustar (*fine-tune*) o modelo LLama 2-7B. Essa abordagem permite que o modelo aprenda a diferenciar perguntas pedagogicamente adequadas daquelas que devem ser evitadas. Diferente de métodos tradicionais de aprendizado por reforço, como *Proximal Policy Optimization* (PPO), o DPO não requer um modelo de recompensa separado, tornando o treinamento mais eficiente e computacionalmente menos custoso.

O processo de otimização ocorre por meio de um ajuste supervisionado inicial (*Supervised Fine-Tuning* - SFT), no qual o modelo é treinado com as perguntas válidas do conjunto de dados original. Em seguida, a etapa de DPO aprimora o modelo ajustando suas preferências para favorecer perguntas que seguem os princípios do método socrático, reduzindo a probabilidade de geração de perguntas inválidas. Esse refinamento permite ao modelo preservar a coerência aprendida no SFT, ao mesmo tempo em que melhora sua capacidade de gerar questões educacionalmente apropriadas.

## Contribuições e resultados

A principal contribuição do estudo está na introdução de um método de aumento de dados baseado na geração de perguntas inválidas, permitindo a criação de um *dataset* de preferência para treinar modelos de linguagem de forma mais eficaz. Além disso, os autores demonstram que a otimização via DPO permite que modelos de código aberto, como o LLama 2-7B, alcancem desempenho competitivo com modelos proprietários muito maiores, como GPT-3.5 e GPT-4, em métricas de similaridade textual, como ROUGE-L e BERTScore.

Os resultados experimentais indicam que:

- O modelo treinado via DPO reduz significativamente a geração de perguntas inválidas, alinhando melhor a produção do LLM com o método socrático.
- O LLama 2-7B otimizado alcança desempenho superior ou equivalente ao GPT-3.5 e GPT-4 na métrica ROUGE-L, destacando-se como uma alternativa mais acessível e eficiente.
- O método proposto oferece uma solução mais acessível e privativa para a geração de perguntas educacionais, visto que o LLama 2 pode ser executado localmente, sem depender de APIs proprietárias.

Apesar dos avanços apresentados, os autores destacam algumas limitações e direções para trabalhos futuros. Primeiramente, a análise foi conduzida exclusivamente no LLama 2-7B, deixando em aberto a possibilidade de aplicação da técnica a modelos de maior escala ou outras arquiteturas de código aberto, como o Mistral. Além disso, a efetividade pedagógica do modelo ainda precisa ser validada em cenários educacionais reais, com estudantes interagindo diretamente com o sistema. Outro aspecto relevante para investigações futuras é a introdução de mecanismos que atribuam pesos diferentes às categorias de perguntas inválidas, uma vez que algumas podem ser mais prejudiciais do que outras para o aprendizado do estudante.

Em suma, o estudo de KUMAR e LAN (2024b) apresenta uma solução inovadora para aprimorar a geração de perguntas socráticas utilizando LLMs de código aberto. A combinação de aumento de dados e otimização por preferência não apenas melhora a qualidade das perguntas geradas, mas também demonstra que modelos menores e acessíveis podem competir com modelos proprietários de grande porte. Os resultados sugerem que a aplicação dessa metodologia pode viabilizar o uso de LLMs em ambientes educacionais, oferecendo suporte pedagógico personalizado de forma eficiente e econômica. À medida que novas pesquisas exploram melhorias adicionais e avaliações empíricas com alunos reais, essa abordagem pode se tornar uma ferramenta valiosa para o ensino assistido por IA.

### 2.2.6 Análise do estudo *Instruct, Not Assist: LLM-based Multi-Turn Planning and Hierarchical Questioning for Socratic Code Debugging*

O estudo de KARGUPTA *et al.* (2024) propõe uma nova abordagem para a instrução socrática em depuração de código utilizando modelos de linguagem de grande porte (LLMs). Os autores argumentam que, embora os LLMs possuam grande potencial para auxiliar estudantes, as abordagens convencionais muitas vezes os configuram como assistentes que fornecem respostas diretas, em vez de incentivar a reflexão e o raciocínio crítico. Para abordar essa limitação, os autores desenvolvem o ***TreeInstruct***, um agente instrutor que emprega planejamento baseado em espaço de estado e questionamento hierárquico para guiar os alunos na identificação e correção de erros de programação de forma independente.

A proposta de KARGUPTA *et al.* (2024) insere-se em um contexto de crescente interesse pelo uso de LLMs no suporte educacional, buscando ampliar a acessibilidade a feedback personalizado. No entanto, os modelos atualmente utilizados são frequentemente treinados para interações de assistência direta, o que pode ser inadequado para o ensino de habilidades de resolução de problemas. O ***TreeInstruct*** resolve esse problema ao reorientar o LLM para agir como um instrutor, estruturando a interação com o aluno por meio de perguntas socráticas que estimulam o aprendizado ativo. O objetivo central da abordagem é garantir que os alunos compreendam e corrijam seus erros sem receber respostas explícitas, promovendo um processo de aprendizado mais profundo.

#### Abordagem proposta

O ***TreeInstruct*** é fundamentado em três princípios-chave:

1. **Estimativa do espaço de estado:** O instrutor modela a interação com base na distância entre a resposta inicial do aluno e a resposta ideal, rastreando seu progresso no espaço de conhecimento ao longo da conversa.
2. **Questionamento socrático baseado em árvore:** O instrutor gera perguntas sequenciais que dependem do conhecimento atual do aluno e de seus erros, construindo dinamicamente uma árvore de questionamento socrático.
3. **Reestruturação adaptativa da conversa:** O instrutor ajusta dinamicamente o plano de ensino com base no progresso do aluno, garantindo que todas as falhas de compreensão sejam abordadas antes da finalização do processo.

Diferentemente de abordagens tradicionais que se baseiam em interações de turno único, o ***TreeInstruct*** adota um modelo de planejamento contínuo. O método

emprega um *Verificador* que acompanha a interação, avaliando as respostas do aluno para atualizar o estado de conhecimento e determinar os próximos passos do diálogo. Essa estrutura permite que o instrutor adapte suas perguntas conforme necessário, evitando abordagens genéricas ou excessivamente diretas.

Para avaliar sua eficácia, os autores utilizam um conjunto de dados já estabelecido para depuração de código, além de um novo *dataset* chamado **MULTI-DEBUG**, que inclui 150 problemas de programação com múltiplos erros, criados e anotados por especialistas. Esse novo conjunto de dados representa um desafio mais complexo, exigindo que os alunos corrijam múltiplos erros interdependentes em suas soluções.

## Contribuições e resultados

O estudo apresenta diversas contribuições significativas para a aplicação de LLMs na educação, destacando-se:

- **Introdução de um modelo instrutor adaptativo:** O *TreeInstruct* é o primeiro sistema a utilizar estimativa de espaço de estado e questionamento baseado em árvore para planejamento multi-turno na depuração de código.
- **Criação de um conjunto de dados desafiador:** O **MULTI-DEBUG** fornece um benchmark mais complexo, permitindo avaliar o desempenho de diferentes estratégias de ensino assistido por IA em cenários realistas.
- **Resultados superiores em comparação a abordagens convencionais:** O *TreeInstruct* superou modelos baseline, como *Vanilla LLM* e *BRIDGE*, tanto em sucesso na depuração quanto na estruturação das interações.

Os experimentos demonstram que o *TreeInstruct* melhora a eficácia das interações ao reduzir a quantidade de respostas diretas e aumentar a coerência do questionamento socrático. A tabela de resultados indica que o método:

- **Aprimora a taxa de sucesso na depuração**, especialmente para múltiplos erros interdependentes.
- **Diminui a necessidade de interações excessivas**, conduzindo os alunos à solução correta com menos perguntas.
- **Garante maior coerência lógica**, assegurando que as perguntas reflitam corretamente os erros cometidos pelo aluno.

Além da avaliação quantitativa, os autores realizam um estudo qualitativo com cinco alunos de diferentes níveis de habilidade em programação. Os resultados mostram que o *TreeInstruct* consegue guiar os estudantes na depuração de seus códigos de maneira mais eficiente, adaptando as interações ao nível de conhecimento de cada um. Isso sugere que a abordagem pode ser generalizada para outros contextos educacionais, proporcionando um suporte instrucional mais eficaz.

O estudo de KARGUPTA *et al.* (2024) introduz um avanço significativo na utilização de LLMs para ensino de depuração de código, enfatizando a importância de estruturar a interação instrucional de forma hierárquica e adaptativa. Ao invés de apenas fornecer correções diretas, o *TreeInstruct* emprega planejamento baseado em espaço de estado para incentivar a reflexão e o raciocínio crítico dos alunos. Essa abordagem se mostra especialmente promissora para contextos de aprendizado autônomo, permitindo que estudantes desenvolvam suas habilidades de programação de maneira mais independente e eficaz.

Dada a robustez do método, os autores sugerem algumas direções futuras para expandir suas aplicações. Entre elas, destacam-se a adaptação para outros domínios além da programação, a incorporação de modelos mais avançados para o agente verificador e a exploração de métodos multimodais, combinando questionamento textual com recursos visuais para facilitar a compreensão de conceitos abstratos. Esses avanços podem consolidar ainda mais o uso de LLMs como ferramentas instrucionais, transformando o ensino assistido por IA em diversas áreas do conhecimento.

## 2.2.7 Considerações Finais sobre o Ensino Socrático

O ensino socrático aplicado à programação representa uma abordagem pedagógica inovadora que fomenta a aprendizagem ativa e a construção do conhecimento por meio do questionamento estruturado. A partir da revisão de literatura, observou-se que essa metodologia não apenas promove um melhor entendimento dos conceitos computacionais, mas também aprimora o pensamento crítico e a resolução de problemas (AL-HOSSAMI *et al.*, 2023b).

Os estudos recentes reforçam que o uso do ensino socrático em programação tem um impacto significativo no desempenho dos alunos. Segundo pesquisas, esse método auxilia no desenvolvimento de habilidades metacognitivas, pois os estudantes são incentivados a refletir sobre sua lógica e depuração de código, levando a uma compreensão mais profunda dos conceitos fundamentais (DING *et al.*, 2024a). Além disso, estudos que analisaram a aplicação de modelos de linguagem na geração de perguntas socráticas demonstram que a interação dialógica baseada nesse método melhora a retenção de conhecimento e torna o aprendizado mais dinâmico (AL-HOSSAMI *et al.*, 2023a).

Apesar de suas vantagens, a implementação do ensino socrático em programação apresenta desafios, como a necessidade de professores altamente qualificados para formular perguntas eficazes e guiar o aprendizado dos alunos sem fornecer respostas diretas (KUMAR e LAN, 2024a). A integração de assistentes baseados em inteligência artificial pode mitigar essa dificuldade, oferecendo suporte automatizado na formulação de perguntas e na análise das respostas dos estudantes (AL-HOSSAMI *et al.*, 2023a).

A partir das análises realizadas, destaca-se a necessidade de mais pesquisas empíricas sobre a eficácia do ensino socrático no contexto da programação, especialmente em diferentes níveis de ensino e com diversos perfis de estudantes. Além disso, a exploração de abordagens híbridas, combinando questionamento socrático com modelos de aprendizado baseados em inteligência artificial, pode ampliar as possibilidades de personalização do ensino e aprimorar a experiência educacional (KUMAR e LAN, 2024a).

Por fim, a aplicação do ensino socrático na programação não se limita ao aprimoramento técnico dos estudantes, mas também fortalece suas habilidades cognitivas essenciais para a solução de problemas complexos. O resgate dessa abordagem filosófica em um contexto tecnológico reafirma a importância do pensamento crítico e do aprendizado ativo na formação de futuros profissionais da computação.

## 2.3 LLMs e Educação

A ascensão dos (LLMs) *Large Language Models* tem promovido avanços significativos na educação, oferecendo novas possibilidades de ensino personalizado, suporte ao aprendizado e automação de processos pedagógicos (DU *et al.*, 2024a; LIU *et al.*, 2024a). Esses modelos, treinados em vastos conjuntos de dados textuais, demonstram capacidades notáveis em compreender e gerar linguagem natural, tornando-se ferramentas promissoras para diversas aplicações educacionais.

### 2.3.1 O Papel dos LLMs na Educação

Os LLMs possuem potencial para transformar a educação por meio da personalização do aprendizado, geração de materiais didáticos, assistência na resolução de problemas e suporte à escrita acadêmica (DU *et al.*, 2024a). Esses modelos são capazes de adaptar-se ao nível de conhecimento do aluno, fornecendo explicações e exemplos sob medida para diferentes contextos de ensino.

Segundo LIU *et al.* (2024a), a aplicação de LLMs no ensino pode seguir dois paradigmas distintos: (i) um modelo tradicional de Pergunta-Resposta (Q&A), onde os alunos apresentam questões e o modelo fornece respostas diretas; e (ii) um modelo

baseado no Ensino Socrático, no qual o LLM conduz os alunos por um processo de reflexão crítica, incentivando-os a formular hipóteses e estruturar seu próprio entendimento. Esse último modelo tem se mostrado mais eficaz para promover habilidades cognitivas avançadas.

Além disso, LLMs podem ser utilizados para gerar materiais didáticos adaptados ao contexto de ensino. Como discutido em DU *et al.* (2024a), esses modelos podem criar exercícios personalizados, fornecer feedback automatizado e gerar explicações detalhadas para conceitos complexos, promovendo um aprendizado mais eficaz. Estudos recentes mostram que o uso de LLMs no ensino de química tem permitido a criação de cenários de aprendizado interativo, em que os estudantes podem explorar conteúdos de maneira adaptativa GAO *et al.* (2024).

### 2.3.2 Análise do estudo *SocraticLM: Exploring Socratic Personalized Teaching with Large Language Models*

O estudo de LIU *et al.* (2024b) propõe o SocraticLM, um novo paradigma para ensino socrático baseado em LLMs, buscando transcender abordagens convencionais de perguntas e respostas. Em vez de fornecer apenas respostas diretas, o modelo desempenha um papel ativo na instrução, estimulando o pensamento crítico e a autonomia na resolução de problemas.

#### Framework e Construção do SocraticLM

Para alcançar esse objetivo, foi desenvolvido um pipeline multiagente denominado Reitor-Professor-Aluno, responsável por estruturar interações de ensino no estilo socrático. Esse pipeline permitiu a criação do conjunto de dados SocraTeach, que contém mais de 35.000 diálogos de ensino multi-rodadas elaborados meticulosamente para simular interações autênticas entre professor e aluno. O conjunto de dados foi construído para representar seis perfis distintos de alunos, com diferentes estados cognitivos, permitindo que o modelo se adapte dinamicamente ao nível de compreensão do aprendiz.

Os agentes do pipeline são estruturados da seguinte forma:

- **Reitor:** Supervisiona e valida as respostas do professor, garantindo que sigam o ensino socrático.
- **Professor:** Responsável por formular perguntas e estimular o raciocínio do aluno sem fornecer respostas diretas.
- **Aluno:** Simula diferentes perfis de estudantes, representando distintos graus de compreensão e dificuldades na resolução de problemas.

Além disso, foi implementado um sistema de estados cognitivos para simular a diversidade de compreensão dos alunos, cobrindo aspectos como interpretação de problemas, cálculo, domínio do conhecimento e sede de aprendizado.

## Treinamento e Ajuste do Modelo

O *fine-tuning* do SocraticLM foi realizado em três estágios:

1. **Treinamento separado:** Primeiramente, o modelo foi ajustado em dados de diálogos socráticos antes de ser treinado em um conjunto menor de resolução de problemas.
2. **Ajuste de instruções:** Foram empregadas instruções específicas para diferenciar diálogos socráticos e resolução de problemas, garantindo que o modelo não simplificasse indevidamente o processo de ensino.
3. **Configuração de *prompt* misto:** Para aprimorar a capacidade de raciocínio do modelo, foi utilizado treinamento em configurações *zero-shot* e *one-shot*.

## Resultados e Avaliação

Para medir o impacto do SocraticLM, foi desenvolvida uma métrica de avaliação composta por cinco dimensões pedagógicas:

- **Precisão no reconhecimento de respostas incorretas (*IARA*):** Avalia a capacidade do modelo de identificar respostas erradas dos alunos.
- **Precisão no reconhecimento de respostas corretas (*CARA*):** Mede a eficácia do modelo em validar respostas corretas.
- **Taxa de explicação bem-sucedida (*SER*):** Analisa se o modelo fornece justificativas adequadas para suas perguntas.
- **Taxa de rejeição bem-sucedida (*SRR*):** Mede a habilidade do modelo em recusar perguntas irrelevantes e redirecionar a conversa de maneira produtiva.
- **Qualidade geral:** Um índice holístico que avalia o alinhamento com o ensino socrático e a experiência do aluno.

Os experimentos demonstraram que o SocraticLM superou o GPT-4 em mais de 12% em qualidade de ensino socrático, além de apresentar melhorias significativas na identificação de erros e na formulação de instruções que incentivam o pensamento crítico.

O *SocraticLM* representa um avanço significativo no uso de LLMs para ensino personalizado, ao introduzir um mecanismo estruturado para interações pedagógicas



mais eficazes. Seu diferencial reside na capacidade de adaptar o estilo de ensino conforme o perfil do aluno, tornando a aprendizagem mais interativa e reflexiva. Os resultados indicam que esse modelo pode ser uma alternativa promissora para aplicações educacionais voltadas ao ensino socrático.

O código e o conjunto de dados SocraTeach estão disponíveis publicamente para futuras pesquisas em <https://github.com/Ljyustc/SocraticLM>.

### 2.3.3 Análise do estudo *Boosting Large Language Models with Socratic Method for Conversational Mathematics Teaching*

O estudo de DING *et al.* (2024b) propõe o **SocraticLLM**, um modelo de ensino socrático para a matemática, projetado para auxiliar estudantes a desenvolver um raciocínio mais estruturado e autônomo por meio de conversação. A pesquisa destaca que, embora os *Large Language Models* (LLMs) tenham alcançado grande sucesso no raciocínio matemático, os métodos existentes focam principalmente na resolução direta de problemas, fornecendo soluções ou utilizando abordagens como *Chain-of-Thought* (**CoT**) para melhorar a precisão das respostas. No entanto, essas abordagens não incentivam ativamente o aprendizado e a compreensão profunda dos conceitos matemáticos.

Para superar essa limitação, os autores propõem um método baseado no ensino socrático, em que o LLM guia os alunos através de questionamentos estratégicos que promovem a reflexão e a descoberta do conhecimento. Essa abordagem difere dos modelos tradicionais, pois não fornece soluções imediatas, mas conduz os alunos a desenvolverem suas próprias estratégias para a resolução de problemas matemáticos.

#### Construção do Conjunto de Dados SocraticMATH

Um dos principais desafios para o desenvolvimento do SocraticLLM foi a ausência de conjuntos de dados específicos para o ensino socrático da matemática. Para preencher essa lacuna, os autores criaram o **SocraticMATH**, um conjunto de dados que contém diálogos estruturados no estilo socrático, cobrindo mais de 500 tópicos do currículo de matemática do ensino fundamental.

O conjunto de dados foi construído em três fases:

1. **Coleta de Dados:** Questões matemáticas foram extraídas de exames reais aplicados no ensino fundamental na China, abrangendo uma ampla diversidade de tópicos.
2. **Pré-Anotação Automática:** Para reduzir o custo de anotação humana,

diálogos iniciais foram gerados automaticamente pelo GPT-4, utilizando um *prompt* cuidadosamente projetado para imitar um professor no estilo socrático.

3. **Anotação Humana:** Como os LLMs nem sempre geram interações matemáticas corretas, três especialistas revisaram manualmente os diálogos, corrigindo erros e aprimorando a qualidade pedagógica das interações. Durante esse processo, 23% dos diálogos foram removidos por inconsistências, e 18% foram modificados para melhorar a lógica e a coerência das respostas.

A análise estatística do SocraticMATH revelou que cada diálogo tem, em média, cinco turnos de interação, garantindo um nível adequado de engajamento e reflexão por parte do aluno. Além disso, a diversidade dos exemplos permite que o modelo aprenda diferentes estratégias pedagógicas para guiar o estudante.

### Método Proposto: SocraticLLM

O modelo **SocraticLLM** foi desenvolvido para atuar como um tutor matemático, utilizando uma estrutura conversacional baseada em quatro estratégias fundamentais:

- **Revisão:** O modelo verifica e analisa as respostas do aluno, garantindo que estejam corretas antes de prosseguir para a próxima etapa do diálogo.
- **Heurística:** Em vez de fornecer respostas diretas, o LLM formula perguntas que incentivam o aluno a explorar diferentes caminhos de solução.
- **Retificação:** Caso o aluno cometa erros, o modelo identifica e corrige as falhas, explicando os equívocos de maneira instrutiva.
- **Sumarização:** Ao final do diálogo, o LLM fornece um resumo dos conceitos abordados, reforçando os aprendizados adquiridos.

Para garantir que o SocraticLLM produza respostas confiáveis e pedagogicamente eficazes, os autores desenvolveram uma abordagem de aprimoramento baseada no uso de conhecimento extra. Esse conhecimento inclui soluções detalhadas dos problemas, permitindo que o modelo oriente os alunos sem fornecer respostas explícitas.

### Resultados e Avaliação

A eficácia do SocraticLLM foi avaliada por meio de métricas automáticas e avaliações humanas. Os experimentos compararam o modelo com diversas abordagens de última geração, incluindo GPT-4 e modelos abertos como LLaMA2-7B e Qwen1.5-7B. Os principais resultados indicam que:

- O SocraticLLM superou o GPT-4 em métricas de confiabilidade e alinhamento com o método socrático.
- Em avaliações humanas, obteve as maiores pontuações em qualidade pedagógica e capacidade de orientação.
- Quando removidos os componentes de *prompt* e conhecimento extra, o desempenho do modelo caiu significativamente, demonstrando a importância dessas técnicas para a melhoria das respostas.

Além das avaliações quantitativas, os autores realizaram um estudo qualitativo que destacou como o SocraticLLM reduz a tendência dos LLMs de aceitar respostas incorretas dos alunos sem questionamento. O modelo se mostrou mais crítico e interativo, promovendo um aprendizado mais ativo e engajado.

O estudo de DING *et al.* (2024b) apresenta uma abordagem inovadora para o ensino de matemática baseado no método socrático, utilizando um modelo de IA que promove a reflexão e o pensamento crítico. O desenvolvimento do SocraticMATH como um conjunto de dados de diálogos matemáticos estruturados fornece uma base sólida para futuras pesquisas na área de ensino assistido por IA.

Os autores destacam algumas direções para pesquisas futuras, incluindo:

- Incorporar grafos de conhecimento para melhorar a contextualização das perguntas.
- Personalizar o ensino com base em informações individuais dos alunos.
- Explorar a aplicação do método em outras disciplinas além da matemática.

Os códigos e o conjunto de dados SocraticMATH estão disponíveis publicamente em <https://github.com/ECNU-ICALK/SocraticMath>, possibilitando o avanço das pesquisas sobre ensino socrático com LLMs.

### 2.3.4 Benefícios e Desafios dos LLMs na Educação

Os principais benefícios dos LLMs na educação incluem:

- **Personalização do Ensino:** LLMs podem adaptar-se ao ritmo e estilo de aprendizado de cada aluno, fornecendo conteúdo customizado LIU *et al.* (2024a).
- **Aprimoramento do Pensamento Crítico:** Modelos que utilizam abordagens socráticas incentivam a participação ativa dos alunos na construção do conhecimento LIU *et al.* (2024a).

- **Suporte à Escrita Acadêmica:** LLMs podem auxiliar na estruturação de textos, revisão de escrita e formulação de argumentos DU *et al.* (2024a).
- **Automação de Feedback:** Com a capacidade de analisar respostas abertas, os LLMs podem fornecer avaliações detalhadas e sugestões de melhoria DU *et al.* (2024a).
- **Auxílio na Preparação de Exames:** Alguns estudos apontam que LLMs podem gerar automaticamente perguntas de avaliação e até mesmo fornecer soluções explicativas GAO *et al.* (2024).

Apesar dos benefícios, existem desafios importantes a serem considerados:

- **Tendência à Geração de Respostas Errôneas:** LLMs podem gerar respostas imprecisas ou alucinações, exigindo mecanismos de validação DU *et al.* (2024a).
- **Falta de Contexto Pedagógico:** Muitas vezes, esses modelos não possuem conhecimento sobre metodologias educacionais específicas, o que pode limitar sua aplicabilidade LIU *et al.* (2024a).
- **Questões Éticas e de Privacidade:** O uso de LLMs na educação levanta preocupações sobre proteção de dados e viés algorítmico DU *et al.* (2024a).
- **Dependência Tecnológica:** O uso excessivo de LLMs pode levar a uma diminuição do pensamento crítico dos alunos, tornando-os excessivamente dependentes dessas ferramentas para resolver problemas acadêmicos GAO *et al.* (2024).

### 2.3.5 Perspectivas Futuras para LLMs na Educação

O futuro dos LLMs na educação dependerá de estratégias para mitigar seus desafios e maximizar seus benefícios. Algumas direções promissoras incluem:

- **Aprimoramento da Interação Socrática:** Avanços no design de prompts e no fine-tuning dos modelos podem aumentar sua capacidade de conduzir diálogos educativos eficazes LIU *et al.* (2024a).
- **Integração com Sistemas de Gestão de Aprendizado (LMS):** Incorporar LLMs em plataformas educacionais pode tornar seu uso mais acessível e eficiente DU *et al.* (2024a).
- **Combinação com Técnicas de Recuperação de Informação:** Utilizar métodos de busca documental pode aumentar a precisão das respostas dos LLMs, reduzindo erros e desinformação DU *et al.* (2024a).

- **Desenvolvimento de Modelos Específicos para Educação:** Alguns pesquisadores propõem o desenvolvimento de LLMs treinados exclusivamente com conteúdos educacionais, reduzindo a tendência a erros e vieses GAO *et al.* (2024).

## 2.4 Conclusão

Os *Large Language Models* representam uma revolução no ensino, possibilitando novas formas de aprendizado interativo e personalizado. Enquanto algumas pesquisas indicam que LLMs podem potencialmente substituir métodos tradicionais de ensino, a visão predominante é que esses modelos devem ser utilizados como ferramentas complementares, auxiliando professores e alunos a expandirem suas capacidades cognitivas e pedagógicas GAO *et al.* (2024).

Embora desafios como a verificação de precisão e preocupações éticas ainda precisem ser enfrentados, os avanços contínuos na área indicam um futuro promissor para sua implementação na educação. Pesquisas recentes DU *et al.* (2024a); LIU *et al.* (2024a) destacam que, ao serem projetados com abordagens adequadas, os LLMs podem se tornar aliados poderosos na formação acadêmica, promovendo um ensino mais acessível, interativo e eficaz. Como apontado em GAO *et al.* (2024), o desenvolvimento de modelos mais especializados e adaptados ao contexto educacional pode ser um passo essencial para garantir que a tecnologia beneficie amplamente o aprendizado, sem comprometer a autonomia cognitiva dos alunos.

## 2.5 Técnicas de Engenharia de Prompt

A engenharia de prompt surgiu como uma estratégia fundamental para maximizar o desempenho dos LLMs em atividades específicas, sem a necessidade de alterar os pesos de seus parâmetros, processo este que costuma ser altamente custoso. Essa técnica consiste na otimização das instruções em linguagem natural, os prompts, projetadas para ativar e orientar de forma eficiente o conhecimento implícito adquirido pelos modelos durante a fase de pré-treinamento. Ao contrário das abordagens baseadas em *fine-tuning*, que exigem a modificação dos parâmetros do modelo, a *engenharia de prompt* atua exclusivamente na interface semântica entre o usuário e o sistema. Seu principal objetivo é estruturar entradas textuais que funcionem como gatilhos cognitivos, permitindo ao modelo acessar padrões relevantes em sua base de conhecimento e gerar saídas alinhadas com objetivos específicos. Além de reduzir significativamente os custos computacionais, essa abordagem favorece uma interação mais eficaz entre humanos e máquinas, na qual a qualidade das respostas do LLM

está diretamente relacionada à estratégia utilizada pelo usuário na formulação dos *prompts*, assegurando o melhor aproveitamento do modelo em contextos diversos.

A adaptação de LLMs por meio de *fine-tuning* requer vastos conjuntos de dados rotulados, resultando em elevado consumo de energia, tempo e recursos financeiros. Esse processo pode ser dispendioso ao ajustar os modelos para funções específicas. Conforme destacado por CHANG *et al.* (2024), a *engenharia de prompt* possibilita o controle do comportamento do modelo através da manipulação do texto de entrada. Os autores observam que "o prompting tem dominado a forma de interação entre humanos e sistemas de IA, com grande potencial para abrir caminho para a AGI" CHANG *et al.* (2024). Além disso, esse paradigma promoveu uma mudança significativa na utilização dos modelos: sistemas generalistas podem ser empregados para executar diversas operações distintas por meio da formulação adequada dos prompts de entrada, eliminando a necessidade de atualizar seus parâmetros internos e, assim, economizando recursos computacionais significativos.

### 2.5.1 Técnicas de Engenharia de Prompt

Diversas abordagens têm sido desenvolvidas para estruturar *prompts* de modo mais efetivo. Essas estratégias englobam desde métodos simples, sem exemplos, como o *Zero-Shot*, passando por procedimentos com poucos exemplos, como o *Few-Shot*, até processos avançados que induzem raciocínio passo a passo ou realizam otimização automática do *prompt*, como o CoT (*Chain-of-Thought*) VATSAL e DUBBEY (2024).

#### Zero-Shot

No zero-shot, o modelo é instigado a realizar uma tarefa sem receber exemplos específicos como entrada. Essa técnica explora exclusivamente o conhecimento adquirido durante o pré-treinamento do modelo, exigindo apenas uma formulação clara no prompt que explicita o objetivo da tarefa a ser realizada. Segundo DEBNATH *et al.* (2025), a engenharia de prompt zero-shot permite que um LLM execute uma tarefa sem ter sido previamente exposto a exemplos semelhantes, seja durante o treinamento ou no próprio prompt. Nessa abordagem, confia-se que o modelo utilize seu extenso repertório de conhecimento pré-treinado para gerar a resposta de forma autônoma.

No entanto, o desempenho em cenários zero-shot tende a ser limitado em tarefas mais complexas. Sem exemplos que ilustrem o formato ou a lógica esperada, o modelo pode interpretar a instrução de maneira ambígua ou falhar em capturar nuances relevantes. Assim, o zero-shot é mais apropriado para atividades diretas ou objetivas, em que uma pergunta bem formulada pode ativar conhecimentos já presentes

no modelo. Por outro lado, mostra-se frequentemente insuficiente quando a tarefa exige passos de raciocínio não triviais ou o reconhecimento de padrões específicos que não estejam explícitos na instrução.

## Few-Shot

Em contraste, o *few-shot* fornece ao modelo alguns exemplos concretos do objetivo da tarefa que o LLM deve desempenhar, geralmente algo entre 1 a 5 exemplos de entrada e saída desejada, antes de solicitar a resolução de um novo caso. Introduzido e amplamente difundido no trabalho de BROWN *et al.* (2020), este método simples demonstrou-se eficiente em aumentar significativamente o desempenho do LLM em tarefas complexas, mesmo com poucos exemplos de qualidade no prompt, especialmente quando comparado ao cenário zero-shot.

Conforme discutido por DEBNATH *et al.* (2025), o *zero-shot prompting* consiste em fornecer ao modelo apenas uma instrução direta, sem exemplos prévios, confiando em sua capacidade de generalização adquirida durante o treinamento anterior: *o zero-shot prompting permite que um modelo de linguagem execute uma tarefa sem ter visto quaisquer exemplos dessa tarefa durante o treinamento ou no prompt*. No entanto, uma abordagem frequentemente mais eficaz é o *few-shot prompting*, que consiste em fornecer ao modelo alguns exemplos representativos da tarefa antes de solicitar a geração de uma nova saída. Essa técnica apresentou resultados expressivos no estudo inovador de BROWN *et al.* (2020), que foi realizado no GPT-3: O modelo é condicionado por alguns pares de entrada e saída fornecidos no prompt, permitindo-lhe inferir o padrão e generalizá-lo para novos casos.

Na prática da engenharia de prompts, o *few-shot prompting* pode ser utilizado para ensinar ao modelo a estrutura desejada da resposta, especialmente útil em tarefas complexas como simular um diálogo de ensino socrático. A seguir, um exemplo ilustrativo:

### Prompt:

*Aluno: Por que os planetas giram ao redor do Sol?*

*Professor: O que você acha que faz os objetos se moverem em linha reta ou em curvas?*

*Aluno: Talvez uma força?*

*Professor: Muito bem. Que força você conhece que age entre objetos com massa?*

*Aluno: A gravidade.*

Neste exemplo, o modelo é guiado por três pares de interação aluno-professor antes de gerar uma nova resposta. Esses exemplos permitem que o modelo aprenda

a forma socrática de responder, ou seja, utilizando perguntas reflexivas ao invés de afirmações diretas. Trata-se de uma aplicação explícita de *few-shot prompting* para controle do estilo e da função pedagógica da resposta.

Em DEBNATH *et al.* (2025) reforçam essa abordagem ao afirmarem que *o desempenho depende significativamente da qualidade, relevância e diversidade dos exemplos fornecidos*, destacando a importância da curadoria cuidadosa dos exemplos dentro do prompt.

De acordo com BROWN *et al.* (2020) também comprovam que modelos como o GPT-3 apresentaram *melhoria notável no desempenho em uma série de tarefas quando fornecido com apenas um punhado de exemplos*, o que valida a eficácia dessa técnica mesmo em tarefas linguísticas sofisticadas e abertas como simulações educacionais.

A vantagem do *few-shot prompting* está em aproveitar melhor o conhecimento do modelo, guiando-o por meio de padrões fornecidos nos exemplos. Entretanto, há limitações claras de acordo com VATSAL e DUBEY (2024) são encontradas como:

- **Custo de contexto** Incluir exemplos consome tokens do limite de contexto do modelo, o que pode ser proibitivo para entradas ou prompts muito longos.
- **Sensibilidade e viés** A escolha e formulação dos exemplos podem influenciar significativamente o comportamento do modelo. Por exemplo, o modelo pode se enviesar a repetir estilos ou preferir respostas semelhantes às dos exemplos fornecidos, em vez de refletir a distribuição real da tarefa.

Com isso, VATSAL e DUBEY (2024) ressaltam que uma seleção cuidadosa dos exemplos de prompt é crítica para desempenho ótimo e para mitigar vieses indesejados no *few-shot*.

Em resumo, o *prompting* few-shot geralmente supera o zero-shot em consistência e acurácia (especialmente em tarefas de maior complexidade ou com saída estruturada), mas exige curadoria e conhecimento do domínio para escolher bons exemplos. Seu contexto ideal de aplicação é quando dispomos de alguns exemplos representativos da tarefa e queremos rapidamente adaptar o modelo a um novo domínio ou formato de resposta, sem realizar *fine-tuning* completo.

## Chain-of-Thought (CoT) Prompting

LLMs frequentemente falham ao lidar com problemas que exigem raciocínio lógico, matemático de múltiplas etapas e capacidade de generalização. Para contornar essa limitação, WEI *et al.* (2022a) introduziram a técnica de *Chain-of-Thought prompting*, na qual o prompt estimula o modelo a pensar passo a passo ao longo da geração da resposta. Ao invés de responder diretamente com a solução final, o



modelo é instruído (geralmente através de uma dica explícita como “Vamos pensar passo a passo”) a produzir uma cadeia de raciocínio intermediária, isto é, explicitar em texto os passos lógicos ou cálculos realizados, antes de chegar à conclusão final. Esse procedimento guia o LLM por um encadeamento lógico de ideias, levando-o a respostas mais estruturadas e coerentes SAHOO *et al.* (2024); WEI *et al.* (2022a).

Por exemplo, diante de um problema matemático de raciocínio, o prompt CoT forneceria ao modelo um exemplo de solução detalhando cada etapa de cálculo e, em seguida, pediria que o modelo fizesse o mesmo para um novo problema. Estudos mostraram ganhos expressivos com CoT: ao habilitar o “raciocínio em cadeia”, modelos gigantes como o PaLM 540B atingiram estado da arte em benchmarks de matemática e raciocínio lógico, obtendo, por exemplo, 90% de acurácia em um conjunto de problemas aritméticos complexos, desempenho muito superior ao obtido com prompts diretos tradicionais (SAHOO *et al.*, 2024; WEI *et al.*, 2022a).

De fato, CoT revelou-se uma capacidade emergente que se manifesta plenamente apenas em LLMs de grande porte, modelos menores não mostraram a mesma melhoria, ao passo que modelos com dezenas ou centenas de bilhões de parâmetros passaram a resolver problemas antes considerados inabordáveis via prompting simples WEI *et al.* (2022a). Em benchmarks de matemática e senso comum, a adoção de cadeias de raciocínio levou a resultados de ponta: por exemplo, com um modelo de 540 bilhões de parâmetros (PaLM), alcançou-se acurácia em torno de 90% em perguntas de raciocínio lógico e aritmético, superando abordagens sem CoT e aproximando-se do desempenho de modelos especificamente treinados para essas tarefas WEI *et al.* (2022a). Essas evidências confirmam que CoT maximiza a capacidade de raciocínio dos LLMs, extraíndo respostas mais fundamentadas e coerentes.

Além do desempenho bruto, uma vantagem colateral do CoT é fornecer maior transparência ou interpretabilidade ao resultado, já que expõe a sequência de passos que levou à conclusão, o que pode ser útil para análise ou verificação do raciocínio. Por outro lado, o CoT Prompting tem limitações a considerar. Primeiramente, a efetividade do método depende do tamanho e qualidade do modelo: somente os modelos mais avançados conseguem seguir instruções do tipo pense passo a passo de maneira confiável, enquanto modelos menores tendem a produzir cadeias de raciocínio confusas ou incorretas (WEI *et al.*, 2022a). Segundo, há um custo de token e tempo, pois a cadeia de pensamento torna a resposta mais longa e o processo de geração mais demorado, o que pode ser inviável em aplicações de tempo real ou com limite estrito de tokens. Terceiro, embora o CoT geralmente melhore a exatidão, se um passo do raciocínio estiver equivocado, a resposta final também estará, isto é, o CoT pode às vezes propagar erros em vez de corrigi-los.

Em síntese, o CoT Prompting é mais apropriado para tarefas complexas que demandam decomposição lógica, como problemas matemáticos, questões de raciocínio

comum e perguntas que envolvem múltiplas etapas de dedução. Nessas situações, ele tende a superar tanto o zero-shot quanto o few-shot convencional em termos de qualidade da resposta (SAHOO *et al.*, 2024; WEI *et al.*, 2022a). Entretanto, para tarefas mais simples ou de resposta direta, o CoT pode ser desnecessário e até contraproducente, dada sua complexidade.

## Comparação

Comparando-se as três abordagens, fica evidente que cada técnica atende melhor a determinados cenários. O Zero-Shot Prompting destaca-se pela aplicabilidade imediata, é útil quando não se dispõe de exemplos ou quando a tarefa é trivial o bastante para não requerer demonstrações. Já o Few-Shot Prompting oferece um equilíbrio: requer um pequeno esforço de curadoria de exemplos, mas em troca entrega melhorias notáveis de desempenho, sendo ideal quando se têm alguns dados de exemplo e deseja-se adaptar rapidamente o modelo a um novo contexto ou estilo de resposta (BROWN *et al.*, 2020). Por fim, o Chain-of-Thought Prompting representa um avanço significativo para problemas de alta complexidade. Estudos recentes enfatizam que técnicas como CoT ampliaram os limites do que LLMs conseguem resolver apenas com prompts, tornando possível atingir ou chegar perto do estado da arte em tarefas antes inatingíveis sem treino específico (CHANG *et al.*, 2024; WEI *et al.*, 2022b). Em contrapartida, essa técnica exige modelos potentes e maior custo computacional, razão pela qual seu uso costuma ser reservado a aplicações em que a qualidade do raciocínio é prioritária. Em síntese, não há uma técnica universalmente superior, mas sim contextos ideais para cada abordagem: o zero-shot é preferível em consultas simples e cenários com ausência de exemplos; o few-shot é recomendado quando há alguns exemplos disponíveis e busca-se melhor performance sem treinar o modelo; e o CoT é indicado quando se necessita extrair do modelo um raciocínio estruturado e aprofundado, maximizando a precisão em tarefas desafiadoras de múltiplas etapas. As evidências empíricas na literatura corroboram essa complementaridade: cada técnica, dentro de seu contexto ótimo, contribui para explorar o potencial dos modelos de linguagem de forma eficaz SAHOO *et al.* (2024); VATSAL e DUBEY (2024). Essa análise comparativa ressalta, portanto, que o sucesso na utilização de LLMs muitas vezes depende de escolher a estratégia de prompting adequada à tarefa, equilibrando-se vantagens e limitações de zero-shot, few-shot e CoT conforme o caso de uso.

## 2.6 Retrieval-Augmented Generation (RAG)

Os LLMs têm revolucionado o processamento de linguagem natural, demonstrando capacidades notáveis. Entretanto, esses modelos enfrentam limitações conhecidas, como alucinações (respostas incorretas ou inventadas) e conhecimento desatualizado devido ao corte temporal dos dados de treinamento LEWIS *et al.* (2020). *Retrieval-Augmented Generation* (RAG) surgiu como uma solução promissora para mitigar esses problemas, incorporando conhecimento externo no processo de geração de texto. Em essência, RAG combina um mecanismo de recuperação de informações (busca por dados relevantes em bases de conhecimento) com um modelo gerador (LLM) para produzir respostas mais acuradas e atualizadas.

### 2.6.1 Conceitos Fundamentais de RAG

O RAG é uma técnica na qual um modelo de linguagem amplia seu conhecimento consultando fontes externas confiáveis antes de gerar uma resposta. Ao invés de confiar apenas na memória estática de seus parâmetros, o modelo busca informações adicionais em bases de dados ou documentos relevantes e as utiliza para fundamentar a geração do texto. O objetivo principal do RAG é otimizar as saídas de um LLM em termos de factualidade e relevância, reduzindo a incidência de informações falsas ou defasadas BORGEAUD *et al.* (2022); LEWIS *et al.* (2020). Isso é particularmente importante em tarefas intensivas em conhecimento e em domínios onde os dados mudam constantemente (por exemplo, notícias recentes, pesquisas científicas em evolução, informações corporativas privadas), uma vez que o conhecimento paramétrico de um LLM tradicional é estático e pode não cobrir eventos ou dados posteriores ao seu treinamento.

Em outras palavras, RAG aumenta as capacidades de um LLM com dados frescos e específicos do domínio, sem necessidade de retreinamento completo do modelo para cada atualização de conhecimento.

Essa técnica apresenta as seguintes vantagens:

- possibilitar que o modelo acesse informações atualizadas ou de nicho no momento da inferência, mantendo as respostas relevantes e corretas;
- melhorar a confiabilidade e a transparência das respostas, pois as informações recuperadas podem servir de referência ou evidência para o que é gerado.

De fato, incorporar evidências recuperadas como contexto pode aumentar a exatidão, controlabilidade e pertinência das respostas de um LLM, atacando de frente

problemas de alucinação e falta de conhecimento específico. Em resumo, RAG representa uma mudança de paradigma onde a geração de linguagem é guiada por recuperação de conhecimento, unindo o melhor dos sistemas de busca com a fluência dos modelos gerativos.

## 2.6.2 Arquitetura Básica de um Sistema RAG

A arquitetura de um sistema RAG combina dois componentes principais: o recuperador de informações (retriever) e o gerador de linguagem (generator). O processo funciona da seguinte forma:

- **Indexação de Conhecimento:** Documentos relevantes são segmentados em trechos e convertidos em representações vetoriais armazenadas em um banco de dados vetorial para busca eficiente.
- **Consulta e Recuperação:** Dada uma entrada do usuário, o sistema recupera documentos relevantes ao criar representações vetoriais da consulta e realizar uma busca por similaridade.
- **Augmentação do Prompt:** O conteúdo recuperado é integrado ao prompt original, formando um prompt enriquecido.
- **Geração da Resposta:** O modelo gerador (LLM) utiliza o contexto fornecido para produzir respostas fundamentadas.

A arquitetura RAG traz consigo uma mudança significativa na forma como os modelos de linguagem abordam problemas de geração de conteúdo. Ao incorporar a recuperação dinâmica de informações, essa técnica supera limitações importantes dos LLMs tradicionais, especialmente em termos de atualização contínua e precisão factual. Na prática, isso possibilita a criação de sistemas robustos, capazes de lidar com a complexidade e a variabilidade inerentes aos ambientes reais, incluindo aplicações críticas na área educacional.

Contudo, ainda existem desafios importantes que precisam ser abordados, como a eficiência computacional em grandes bases de dados, a garantia de qualidade das fontes recuperadas e a integração perfeita entre recuperação e geração. Futuras pesquisas e desenvolvimentos nessa área têm o potencial de ampliar ainda mais as aplicações e a eficácia dos sistemas RAG, garantindo respostas cada vez mais confiáveis, atualizadas e relevantes em diversos domínios.

## 2.7 Data Augmentation

Os processos de rotulagem e preparação de dados para o treinamento de modelos de IA demandam alto investimento em mão de obra humana e recursos financeiros significativos TAN *et al.* (2024). Nesse cenário, os LLMs, como o GPT-4, representam uma possibilidade inédita de automatização e otimização substancial dessas etapas. Em contraste com a dependência tradicional da anotação manual, pesquisas recentes têm explorado esses LLMs instruídos por linguagem natural para duas funções críticas: a anotação automática de dados (rotulagem de dados brutos) e a geração de dados sintéticos (criação artificial de exemplos anotados). Esses resultados são posteriormente aplicados no refinamento (*fine-tuning*) de modelos menores. Essa estratégia viabiliza a transferência indireta do conhecimento implícito em LLMs avançados para arquiteturas compactas, por meio de um mecanismo comparável à destilação de conhecimento ou ao auto-treinamento, em que as previsões do LLM maior atuam como rótulos indiretos para o treinamento de LLMs menores WANG *et al.* (2021).

Consequentemente, estudos recentes evidenciam avanços significativos na utilização desses LLMs, com bilhões de parâmetros, como geradores de dados para treinamento. No estudo de WANG *et al.* (2021), por exemplo, o GPT-3 foi implementado para anotação automática em múltiplos cenários de tarefas de processamento de linguagem natural, alcançando reduções entre 50% e 96% nos custos operacionais em comparação à rotulagem manual, sem comprometer a acurácia dos modelos treinados com esses dados. Notavelmente, os autores observaram que modelos menores treinados com dados sintéticos conseguiram igualar ou até superar o desempenho do próprio GPT-3, utilizando técnicas de engenharia de prompt, como few-shots. TÖRNBERG (2023), ao avaliarem o GPT-4, relataram que o modelo superou consistentemente tanto anotadores não especializados quanto especialistas humanos em qualidade de classificação, apresentando um custo por rótulo de magnitude inferior ao da anotação humana tradicional.

Outras investigações, como a de GOEL *et al.* (2023), demonstraram que, em uma tarefa de extração de informações médicas, o uso de um LLM acelerou o processo de rotulagem em 58% do tempo em comparação aos métodos tradicionais, mantendo um desempenho equivalente ao de anotadores humanos especializados na área médica. De modo semelhante, ALIZADEH *et al.* (2025) demonstraram que as anotações produzidas automaticamente pelo ChatGPT apresentaram qualidade superior às produzidas por anotadores humanos, sendo cerca de 30 vezes mais custo-efetivas. Esses trabalhos ressaltam que os LLMs podem gerar rapidamente grandes volumes de dados de treinamento de alta qualidade e baixo custo, algo particularmente valioso em cenários de poucos recursos ou em domínios nos quais faltam

dados anotados suficientes. Em resumo, a utilização de LLMs como ferramenta de *data augmentation* e anotação automática desponta como uma estratégia promissora para atenuar o gargalo de dados anotados em aprendizado supervisionado. Os ganhos em custo, velocidade e escala proporcionados por LLMs, aliados ao fato de que os modelos menores ajustados com esses dados frequentemente mantêm ou até melhoram o desempenho em relação a dados totalmente humanos, tornam essa abordagem extremamente atrativa do ponto de vista prático.

## 2.8 Fine-Tuning e LoRA Adapters

O treinamento de um LLM demanda, em sua etapa inicial, extensos conjuntos de dados textuais não rotulados para aprendizagem de padrões linguísticos genéricos. Entretanto, quando se almeja especializar tais modelos na geração de respostas específicas para domínios de conhecimento particulares ou em tarefas especializadas, torna-se necessário adaptar o modelo pré-treinado mediante um processo de refinamento. Esta adaptação ocorre através do ajuste fino (*fine-tuning*) dos parâmetros do modelo, utilizando para isso conjuntos de dados menores e devidamente etiquetados correspondentes ao domínio-alvo. Esse paradigma de pré-treino seguido de fine-tuning revolucionou o campo a partir de trabalhos como o BERT DEVLIN *et al.* (2019), que demonstrou desempenho de ponta em diversas tarefas após o ajuste fino em conjuntos de dados específicos. Com o aumento significativo no número de parâmetros dos modelos, a realização do fine-tuning tornou-se um desafio computacional e de armazenamento BROWN *et al.* (2020). O propósito fundamental dessa técnica mantém-se inalterado: especializar o modelo para alcançar alta performance em uma tarefa específica, aproveitando o conhecimento generalista internalizado durante o pré-treinamento. Essa abordagem dispensa a necessidade de treinar um novo modelo a partir do zero, otimizando recursos computacionais e preservando a base linguística já consolidada.

Os benefícios do *fine-tuning* são evidentes. Primeiro, ele geralmente requer uma quantidade menor de dados do que treinar um modelo do zero, já que ele reaproveita os conhecimentos já incorporados no pré-treinamento DEVLIN *et al.* (2019). Segundo, é mais rápido convergir para uma boa solução, dado que os parâmetros inicializam próximos de uma configuração já competente. Terceiro, o modelo ajustado tende a manter conhecimentos prévios e adaptar-se à nova tarefa, obtendo desempenhos superiores ao uso de modelos genéricos sem *fine-tuning* HOWARD e RUDER (2018). No entanto, juntamente com essas vantagens, emergem desafios relevantes. Um dos principais problemas é a ineficiência em termos de armazenamento: para cada nova tarefa, um novo conjunto completo de pesos ajustados deve ser armazenado, o que se torna inviável em modelos de grande escala HOULSBY *et al.* (2019). Além disso,

atualizar todos os bilhões de parâmetros de um LLM demanda recursos computacionais massivos, tanto em tempo de processamento quanto em memória de GPU, o que frequentemente inviabiliza a aplicação do *fine-tuning* tradicional em cenários de hardware limitado BROWN *et al.* (2020). Há também o risco da chamada catástrofe do esquecimento (*catastrophic forgetting*), fenômeno em que o ajuste do modelo a uma tarefa específica pode degradar sua performance em conhecimentos ou habilidades gerais adquiridas no pré-treinamento. Por fim, garantir que o modelo generalize adequadamente, evitando overfitting aos dados de fine-tuning que frequentemente são de pequeno porte constitui um desafio adicional, demandando técnicas específicas de treinamento e regularização LUO *et al.* (2023).

O método de ajuste fino completo (*full fine-tuning*) é realizado em três etapas sequenciais: a passagem inicial pelo modelo original, o cálculo dos gradientes via backpropagation, e a passagem pelo modelo atualizado, conforme ilustrado por 2.1.

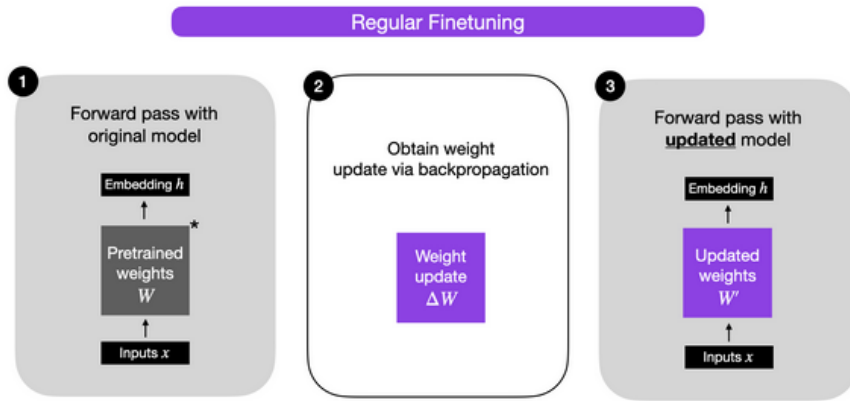


Figura 2.1: Processo de *Regular Finetuning* (Full Fine-Tuning), adaptado de RASCHKA (2023). O processo é dividido em três etapas sequenciais: (1) **Forward pass inicial** com o modelo pré-treinado original, onde os inputs ( $x$ ) são processados pelos pesos iniciais ( $W$ ), gerando embeddings intermediários ( $h$ ); (2) **Atualização dos pesos via backpropagation**, onde é calculado o vetor de atualização de pesos ( $\Delta W$ ) com o objetivo de adaptar o modelo à tarefa-alvo; (3) **Forward pass com o modelo atualizado**, utilizando os pesos ajustados ( $W' = W + \Delta W$ ) para produzir saídas mais adaptadas e precisas à tarefa específica.

Com esses desafios, surgiram técnicas eficientes para realizar o *fine-tuning* desses grandes modelos de linguagens, como a Adaptação LoRA, que reduz drasticamente o custo computacional e o armazenamento necessário ao ajustar apenas subconjuntos de parâmetros do modelo..

## LoRA

Uma das inovações mais influentes em técnicas de *fine-tuning* de LLMs é o LoRA (*Low-Rank Adaptation*), proposto por HU *et al.* (2022). O objetivo do LoRA está

em explorar a redundância linear presente nas atualizações de pesos necessárias para a adaptação de uma nova tarefa.

Ao invés de atualizar diretamente as grandes matrizes de pesos de um Transformer, o LoRA propõe aprender duas matrizes de dimensões reduzidas que, quando multiplicadas, aproximam a variação desejada nesses pesos. Matematicamente, seja  $W \in \mathbb{R}^{m \times n}$  uma matriz de pesos pré-treinada de uma camada do modelo. No *fine-tuning* tradicional, a atualização seria realizada por meio de uma matriz  $\Delta W$  de mesma dimensão, tal que:

$$W_{\text{adaptado}} = W + \Delta W. \quad (2.1)$$

No entanto, o LoRA assume que  $\Delta W$  possui *low-rank*, ou seja, pode ser decomposta como o produto de duas matrizes de baixa dimensionalidade:

$$\Delta W = B \cdot A, \quad (2.2)$$

onde  $B \in \mathbb{R}^{m \times r}$  e  $A \in \mathbb{R}^{r \times n}$ , com  $r \ll m, n$ . Assim, apenas as matrizes de adaptação  $A$  e  $B$  são efetivamente treinadas durante o *fine-tuning*, enquanto os pesos originais  $W$  permanecem congelados. Ao final do processo, a matriz adaptada do modelo resulta em:

$$W_{\text{adaptado}} = W + B \cdot A. \quad (2.3)$$

A hipótese subjacente ao LoRA é que as mudanças necessárias em um LLM pré-treinado para adaptá-lo a uma tarefa específica residem em um *subespaço de dimensionalidade muito menor* do que o espaço completo dos parâmetros HU *et al.* (2022). Isso se alinha com observações empíricas de que os gradientes de *fine-tuning* frequentemente possuem redundâncias lineares significativas AGHAJANYAN *et al.* (2021).

Ao restringir  $\Delta W$  a um posto  $r$  fixo, o LoRA força a atualização a ter essa baixa dimensionalidade intrínseca. Quando  $r$  é escolhido adequadamente, o modelo consegue capturar as adaptações necessárias com muito menos graus de liberdade, sem perda notável de desempenho. Por outro lado, se  $r$  for muito pequeno, a capacidade de adaptação pode ficar insuficiente; se for grande demais, o LoRA aproxima-se do ajuste tradicional (quando  $r = \min(m, n)$ ), perdendo os ganhos de eficiência.

Na prática, valores de  $r$  bastante pequenos (por exemplo, 8, 16 ou 32) têm se mostrado suficientes para muitas tarefas, correspondendo a apenas 0,1% a 0,5% do número de parâmetros original de cada matriz HU *et al.* (2022).

O estudo HU *et al.* (2022) investigou a *deficiência de posto* (*rank deficiency*) em adaptações de modelos de linguagem, mostrando evidências de que há de fato uma



LoRA weights,  $W_A$  and  $W_B$ , represent  $\Delta W$

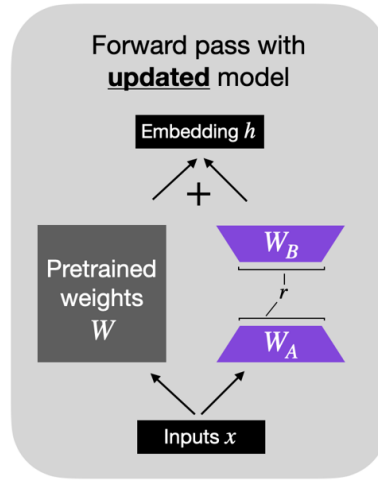


Figura 2.2: Ilustração conceitual do mecanismo do LoRA, adaptado de RASCHKA (2023). O modelo pré-treinado fornece os pesos originais  $W$  (caixa cinza), e o LoRA aprende duas matrizes de baixo posto  $W_A$  e  $W_B$  (em roxo, de dimensão  $r$  muito menor) que juntas formam a atualização  $\Delta W = W_A W_B$ . Na inferência, o resultado da camada é a soma do efeito dos pesos originais congelados com a contribuição ajustada pelo LoRA.

estrutura de baixo posto nas matrizes de atualização, o que explica a eficácia do LoRA. Foi relatado no mesmo estudo, altíssima eficiência de parâmetros ao reduzir drasticamente o número de parâmetros treináveis, o LoRA minimiza os requisitos de memória e armazenamento.

Diante do exposto, o método LoRA consolida-se como uma das estratégias mais eficazes e viáveis para o *fine-tuning* eficiente dos LLMs, especialmente em um cenário marcado pelo crescimento exponencial da escala e da complexidade desses modelos. Seu principal diferencial reside na capacidade de realizar adaptações relevantes utilizando uma fração extremamente reduzida dos parâmetros originais do modelo, o que resulta em benefícios diretos relacionados à redução de memória, economia de armazenamento e viabilização de treinamento em ambientes computacionais com recursos limitados.

Além de sua eficiência paramétrica, o LoRA mantém desempenho competitivo quando comparado ao *fine-tuning* tradicional, preservando a acurácia em tarefas variadas e com diferentes arquiteturas de modelos. Em um de seus experimentos mais notáveis, os autores demonstraram que, ao aplicar LoRA em um modelo GPT-3 com 175 bilhões de parâmetros, bastava ajustar apenas 10.000 pesos treináveis para alcançar resultados praticamente equivalentes aos obtidos com o ajuste completo de todos os parâmetros. Essa descoberta reforça a hipótese de que as atualizações necessárias para adaptação a novas tarefas residem em um subespaço de

baixa dimensionalidade, e que o modelo pode ser eficientemente especializado sem reconfigurar toda a sua estrutura interna.

Outro aspecto relevante é o fato de que, após o treinamento, as atualizações realizadas por meio das matrizes de baixo posto podem ser incorporadas diretamente aos pesos originais do modelo, garantindo que a velocidade de inferência e o consumo de memória permaneçam inalterados. Ademais, a técnica favorece a modularidade, permitindo o reaproveitamento de um mesmo modelo base para múltiplas tarefas, com o armazenamento e o carregamento de conjuntos distintos de matrizes de adaptação. Por fim, destaca-se que o LoRA apresenta simplicidade de implementação, o que contribui para sua ampla adoção na prática, seja em ambientes acadêmicos ou no setor industrial.

Dessa forma, o LoRA se consolida como uma das principais estratégias contemporâneas de fine-tuning eficiente, oferecendo um equilíbrio notável entre simplicidade, baixo custo computacional e desempenho competitivo, contribuindo de maneira significativa para a democratização do uso de LLMs em diversos contextos e aplicações, conforme discutido por HU *et al.* (2022).

## 2.9 Sistemas Multi-Agente em IA

Os LLMs estão revolucionando diversas tarefas de PNL ao demonstrar as capacidades notáveis de compreensão e geração de texto em nível humano. Nos últimos anos, pesquisadores têm explorado a possibilidade de ir além de um único agente LLM, investigando sistemas multi-agente nos quais múltiplos LLMs ou instâncias de LLMs interagem e colaboram para resolver problemas complexos. A motivação subjacente é que múltiplos agentes, comunicando-se entre si, podem trazer perspectivas diversas, verificação mútua e especialização em pequenas tarefas, superando limitações de um agente isolado. Estudos recentes indicam que agentes LLM cooperativos podem melhorar a precisão e o raciocínio, estimular o pensamento divergente e facilitar a conclusão de tarefas de forma mais eficaz BO *et al.* (2024).

Para aproveitar essas potencialidades, várias arquiteturas de sistemas multi-agente com LLMs foram propostas. Essas arquiteturas definem como os agentes são organizados, quais papéis desempenham e como se comunicam (por exemplo, via linguagem natural em prompt). Em paralelo, aplicações práticas emergiram para avaliar se múltiplos LLMs colaborando realmente superam abordagens tradicionais de agente único em termos de desempenho, precisão ou capacidade de resolver tarefas em conjunto. Adicionalmente, trabalhos investigam os fundamentos teóricos da coordenação e comunicação entre agentes baseados em LLMs por exemplo, como estabelecer protocolos de diálogo, como garantir consistência de informação entre agentes, e como dividir recompensas ou tarefas entre eles.

### 2.9.1 Arquiteturas

Diversas arquiteturas têm sido propostas para estruturar sistemas onde múltiplos LLMs atuam como agentes. Em geral, cada agente é instanciado por um modelo de linguagem (seja o mesmo LLM com prompts diferenciados ou até modelos distintos) e se comunica por meio de linguagem natural, trocando mensagens em formato de texto. A seguir, destacam-se algumas abordagens encontradas na literatura recente:

- **Arquitetura Baseada em Papéis (Role-Playing)** No trabalho de LI *et al.* (2023), propõe-se a definição de papéis específicos para cada agente, utilizando prompts para atribuir personalidades ou funções distintas, inspiradas no conceito de "sociedade da mente" MINSKY (1988), o método sugere que múltiplas instâncias de um LLM (agentes) gerem respostas individuais para uma mesma pergunta, debatendo essas respostas em rodadas sucessivas. O debate permite aos agentes criticar e revisar suas próprias respostas com base nas respostas dos outros, convergindo para uma resposta comum e mais precisa.. No framework CAMEL (*Communicative Agents for Mind Exploration of Large Language Model Society*), por exemplo, dois ou mais agentes com papéis complementares (como usuário e assistente, ou especialistas de áreas distintas) engajam-se em diálogos simulados para resolver tarefas. A estratégia emprega *inception prompting*, prompts iniciais que definem contexto e objetivos, para guiar as interações dos agentes, mantendo alinhamento com as intenções humanas. As mensagens trocadas geram dados conversacionais, úteis tanto para análise de comportamentos coletivos quanto para fine-tuning de modelos. Essa arquitetura prioriza a cooperação autônoma, minimizando a necessidade de intervenção humana direta. Variações incluem agentes representando perspectivas técnicas divergentes ou especialistas em domínios variados colaborando em tarefas integradoras.
- **Arquitetura Hierárquica** Em ZHANG *et al.* (2024b), os agentes são organizados em uma estrutura hierárquica, com um agente coordenador (mestre) responsável por decompor tarefas e integrar resultados, enquanto agentes trabalhadores (especialistas) focam em subproblemas. O framework Chain-of-Agents (CoA) proposto utiliza uma cadeia sequencial: agentes trabalhadores processam segmentos de texto longo, comunicam conclusões parcialmente, e o coordenador sintetiza as contribuições em uma saída coerente. Essa abordagem intercala etapas de leitura e raciocínio: cada agente lida com trechos curtos, contornando limitações de janela de contexto dos LLMs, e repassa resumos essenciais. A hierarquia mostrou-se eficaz para gerenciar contextos extensos, melhorando a atenção a detalhes críticos.

- **Arquitetura de Debate Multi-Agente** No trabalho de DU *et al.* (2024b), múltiplos agentes interagem de forma colaborativa ou competitiva para alcançar consenso. Nesse paradigma, cada LLM propõe soluções, critica argumentos alheios e refina suas posições iterativamente, simulando debates humanos. No estudo citado, diferentes instâncias de um LLM discutem até convergir em uma resposta consensual. Em cada rodada, os agentes geram respostas, avaliam propostas dos demais e ajustam suas contribuições, repetindo o ciclo até um resultado final (por voto ou consenso). Diferentemente da hierarquia fixa, não há um líder definido; todos contribuem igualmente, privilegiando a dinâmica coletiva para aprimorar a qualidade da resposta.

Sistemas multi-agente baseados em LLMs representam uma fronteira promissora na inteligência artificial, integrando avanços recentes em modelos de linguagem com princípios clássicos de coordenação entre agentes. Como demonstrado por frameworks como *CAMEL*, *Chain-of-Agents* (CoA) e debates multi-agente, a diversidade arquitetural, de hierarquias com agentes mestres e trabalhadores a debates simétricos entre pares, permite adaptação a contextos variados. Esses sistemas não apenas superam as limitações de agentes únicos, como maior precisão e resolução de problemas robusta, mas também validam a motivação inicial desta pesquisa: a colaboração entre LLMs, mediada por comunicação em linguagem natural, potencializa verificação mútua, síntese de perspectivas e especialização.

Os desafios inerentes a essa abordagem, como a manutenção de contexto e o gerenciamento de conflitos, são mitigados por mecanismos teórico-práticos, de prompts estruturados a memórias compartilhadas, cuja eficácia é corroborada por comparativos quantitativos em benchmarks. Em síntese, a inteligência coletiva emergente dessas arquiteturas não só resolve tarefas complexas com eficiência, mas também inaugura um paradigma onde adaptabilidade e interpretabilidade redefinem o potencial da IA em cenários do mundo real.

## 2.10 Otimização Direta de Preferências (DPO)

A Otimização Direta de Preferências (DPO) é uma técnica recente desenvolvida pela universidade de Stanford RAFAILOV *et al.* (2023a), para alinhar os LLMs às preferências humanas de forma mais direta e eficiente. Diferentemente de abordagens tradicionais, como o Aprendizado por Reforço a partir de Feedback Humano (RLHF), que requerem a construção de modelos de recompensa separados, a DPO simplifica o processo ao ajustar diretamente o modelo principal com base em dados de preferências humanas RAFAILOV *et al.* (2023a). Neste capítulo, exploraremos detalhadamente a formulação matemática da DPO e forneceremos exemplos numé-

ricos para ilustrar sua aplicação no contexto do ensino socrático.

### 2.10.1 Formulação Matemática da DPO

A DPO busca ajustar um modelo de linguagem  $\pi_\theta$  para que ele gere respostas alinhadas às preferências humanas. Consideramos um conjunto de dados de preferências composto por trios  $(x, y^+, y^-)$ , onde:

- $x$ : *prompt* ou entrada fornecida ao modelo.
- $y^+$ : resposta preferida pelos humanos.
- $y^-$ : resposta menos preferida ou rejeitada.

O objetivo é maximizar a probabilidade de  $y^+$  em relação a  $y^-$ . A função de perda (*loss function*) utilizada na DPO é definida como:

$$L(\theta) = \mathbb{E}_{(x, y^+, y^-)} \left[ \log \sigma \left( \beta \left( \log \frac{\pi_\theta(y^+|x)}{\pi_{\text{ref}}(y^+|x)} - \log \frac{\pi_\theta(y^-|x)}{\pi_{\text{ref}}(y^-|x)} \right) \right) \right] \quad (2.4)$$

Onde:

- $\pi_\theta(y|x)$ : é a probabilidade da resposta  $y$  dada a entrada  $x$  segundo o modelo ajustado.
- $\pi_{\text{ref}}(y|x)$ : é a probabilidade da resposta  $y$  dada a entrada  $x$  segundo um modelo de referência (geralmente o modelo antes do ajuste fino).
- $\sigma(z) = \frac{1}{1+e^{-z}}$ : é a função sigmoide, que mapeia valores reais para o intervalo  $(0, 1)$ .
- $\beta$ : é um hiperparâmetro que controla a escala do aprendizado.

Essa formulação busca aumentar a probabilidade das respostas preferidas em relação às não preferidas, alinhando o modelo às preferências humanas de maneira direta e eficiente RAFAILOV *et al.* (2023a).

### 2.10.2 Exemplo Numérico

Para ilustrar a aplicação da DPO, consideremos um exemplo no contexto do ensino socrático. Suponha que temos a seguinte entrada e duas possíveis respostas:

- **Entrada ( $x$ ):** "Explique o teorema de Pitágoras."
- **Resposta preferida ( $y^+$ ):** "O teorema de Pitágoras afirma que, em um triângulo retângulo, o quadrado da hipotenusa é igual à soma dos quadrados dos catetos."

- **Resposta não preferida ( $y^-$ ):** "Pitágoras era um filósofo grego antigo."

Suponhamos que as probabilidades atribuídas pelo modelo de referência  $\pi_{\text{ref}}$  sejam:

- $\pi_{\text{ref}}(y^+ | x) = 0,6$
- $\pi_{\text{ref}}(y^- | x) = 0,4$

Após o ajuste utilizando a DPO, o modelo ajustado  $\pi_\theta$  atribui as seguintes probabilidades:

- $\pi_\theta(y^+ | x) = 0,8$
- $\pi_\theta(y^- | x) = 0,2$

Aplicando a função de perda da DPO:

$$\begin{aligned}
 L(\theta) &= \log \sigma \left( \beta \left( \log \frac{0,8}{0,6} - \log \frac{0,2}{0,4} \right) \right) \\
 &= \log \sigma \left( \beta \left( \log \frac{4}{3} - \log \frac{1}{2} \right) \right) \\
 &= \log \sigma \left( \beta \left( \log \frac{4}{3} + \log 2 \right) \right) \\
 &= \log \sigma \left( \beta \left( \log \frac{4 \times 2}{3} \right) \right) \\
 &= \log \sigma \left( \beta \left( \log \frac{8}{3} \right) \right)
 \end{aligned}$$

Supondo  $\beta = 1$ :

$$\begin{aligned}
 L(\theta) &= \log \sigma \left( \log \frac{8}{3} \right) \\
 &= \log \sigma (\log 2,6667) \\
 &= \log \sigma (0,9808) \\
 &= \log \left( \frac{1}{1 + e^{-0,9808}} \right) \\
 &\approx \log(0,7273) \\
 &\approx -0,3185
 \end{aligned}$$

Portanto, a função de perda da DPO,  $L(\theta)$ , é aproximadamente  $-0,3185$  quando  $\beta = 1$ .

O valor negativo indica que o modelo está penalizando a resposta não preferida, alinhando-se às preferências humanas.

### 2.10.3 Aplicação da DPO no Ensino Socrático

No contexto do ensino socrático, a DPO pode ser utilizada para treinar LLMs a gerarem perguntas que incentivem o pensamento crítico dos alunos. Por exemplo, ao invés de fornecer respostas diretas, o modelo pode ser ajustado para preferir perguntas que guiem o aluno a descobrir a resposta por si mesmo. Isso é alcançado ao utilizar dados de preferências onde perguntas que promovem a reflexão são marcadas como preferidas, enquanto perguntas que fornecem respostas diretas são marcadas como não preferidas.

A Otimização Direta de Preferências oferece uma abordagem eficiente para alinhar Grandes Modelos de Linguagem às preferências humanas, eliminando a necessidade de modelos de recompensa separados. Sua aplicação no ensino socrático pode potencializar a capacidade dos LLMs em promover o aprendizado ativo e o pensamento crítico, alinhando-se aos princípios dessa metodologia pedagógica.

## 2.11 Temperatura na Geração de Texto com LLMs: Mecanismos e Implicações

A temperatura é um hiperparâmetro crucial no processo de geração de texto por LLMs, atuando como um modulador da aleatoriedade durante a amostragem de tokens. Essencialmente, a temperatura ajusta a distribuição de probabilidade sobre o vocabulário de possíveis próximos tokens, influenciando diretamente o equilíbrio entre determinismo e diversidade nas saídas geradas pelo modelo.

### Mecanismo de Ação e Formulação Matemática

O mecanismo pelo qual a temperatura exerce sua influência ocorre antes da aplicação da função softmax, que converte os logits (valores brutos de pontuação para cada token) em probabilidades. A temperatura  $T$  re-escala esses logits: um valor de  $T > 1$  “achata” a distribuição, tornando as probabilidades entre tokens mais uniformes, enquanto um valor de  $0 < T < 1$  a torna mais “aguda”, concentrando a probabilidade nos tokens de maior logit inicial.

Seja  $z_i$  o logit atribuído pelo modelo ao token  $i$ , a probabilidade final  $P(i)$  para a escolha do próximo token, após a aplicação da temperatura  $T$ , é calculada pela função softmax modificada:

$$P(i) = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \quad (2.5)$$

Onde:

- $T = 1$  resulta na distribuição de probabilidade original aprendida pelo modelo.

- $T \rightarrow 0$  faz com que a distribuição se aproxime de uma função determinística (argmax), selecionando quase sempre o token com o maior logit.
- $T > 1$  aumenta a entropia da distribuição, permitindo que tokens menos prováveis sejam amostrados com maior frequência, incentivando a exploração e a diversidade.

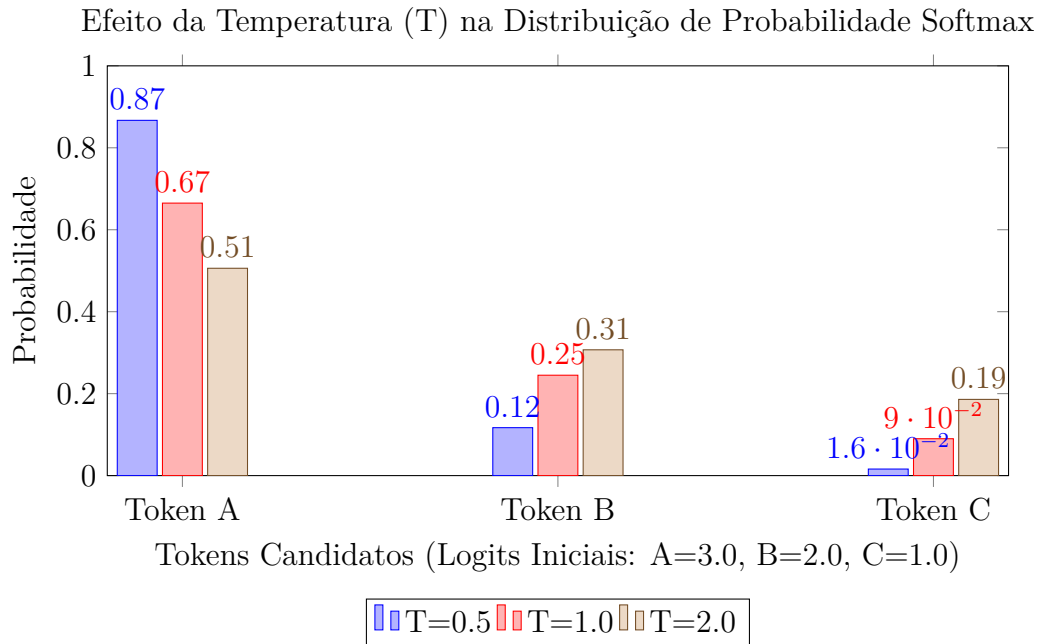


Figura 2.3: Efeito da Temperatura (T) na Distribuição de Probabilidade Softmax sobre Tokens Candidatos. Temperaturas baixas (ex:  $T=0.5$ ) concentram a probabilidade no token mais provável, enquanto temperaturas altas (ex:  $T=2.0$ ) tornam a distribuição mais uniforme.

## Impacto Prático e Relação com Outras Técnicas de Amostragem

A escolha da temperatura tem implicações diretas no comportamento do LLM:

- **Temperaturas Baixas (e.g.,  $T < 0.5$ ):** Geram texto mais focado, previsível, consistente e, por vezes, repetitivo. São úteis em tarefas que exigem precisão factual, respostas diretas ou aderência estrita a um formato (ex: resumo, tradução).
- **Temperaturas Médias (e.g.,  $0.7 \leq T \leq 1.0$ ):** Frequentemente oferecem um bom equilíbrio entre coerência e criatividade controlada, adequadas para muitas tarefas conversacionais e de escrita criativa.
- **Temperaturas Altas (e.g.,  $T > 1.0$ ):** Maximizam a diversidade, a “surpresa” e a exploração de novas ideias. No entanto, aumentam significativamente o risco de gerar texto incoerente, semanticamente falho, ou que se desvia do tópico (“off-topic”).



É importante notar que a temperatura não é a única técnica para controlar a aleatoriedade. Métodos como *Top-K sampling* (que considera apenas os K tokens mais prováveis) e *Nucleus sampling* (Top-P, que considera o menor conjunto de tokens cuja soma de probabilidade ultrapassa P) também são amplamente utilizados, muitas vezes em combinação com a temperatura, para refinar o processo de amostragem e evitar a inclusão de tokens muito improváveis, mesmo com temperaturas mais altas.

### **Aplicação e Relevância para o Ensino Socrático nesta Pesquisa**

Nesta dissertação, a temperatura foi um parâmetro chave explorado durante a geração de dados sintéticos para o ensino socrático (Capítulo 4). A hipótese era que a temperatura ideal para gerar perguntas socráticas eficazes, aquelas que são abertas, relevantes, estimulantes e que guiam o aluno sem dar a resposta, poderia residir em uma faixa intermediária.

- Uma temperatura muito baixa ( $T \rightarrow 0$ ) poderia levar o agente “professor” a gerar perguntas muito óbvias, repetitivas ou fechadas, limitando a reflexão do aluno.
- Uma temperatura muito alta ( $T > 1.0$ ) poderia resultar em perguntas desconexas, confusas ou irrelevantes para o erro do aluno, prejudicando o fluxo dialógico.

A análise experimental buscou identificar empiricamente a faixa de temperatura que otimiza a qualidade pedagógica das interações geradas, conforme avaliado pelo GPT-4. Embora estudos como o de Renze e Guven (2024) RENZE e GUVEN (2024) sugiram que variações de temperatura entre 0.0 e 1.0 podem não ter impacto *estatístico* significativo no desempenho de LLMs em tarefas de *resolução de problemas*, esta pesquisa investiga seu impacto *qualitativo* na adequação pedagógica e no estilo da interação gerada, um aspecto crucial para a simulação eficaz do método socrático. Os resultados apresentados no Capítulo 4 fornecerão insights sobre como a aleatoriedade controlada pela temperatura afeta a utilidade educacional das saídas do LLM neste contexto específico.

# Capítulo 3

## Metodologia

### 3.1 Descrição Geral do Framework

Este capítulo apresenta o desenvolvimento de um framework destinado a apoiar professores e alunos no ensino de programação em ambientes de ensino. O objetivo principal é aliviar a carga dos docentes ao automatizar os processos-chave do ensino que variam desde a criação de exercícios até a correção automática, com o uso de técnicas avançadas de Inteligência Artificial (IA). Este framework busca integrar o método socrático ao ensino de programação, promovendo uma aprendizagem mais interativa e eficaz do que os métodos tradicionais conhecidos.

#### 3.1.1 Motivação e Contexto

Embora existam diversas ferramentas voltadas para a correção automática de códigos como: LeetCode, CodeChef, Becrowd, Exercism, e o ***Machine Teaching*** que foi desenvolvida e é utilizada pela Universidade Federal do Rio de Janeiro (UFRJ) para o ensino de python nas disciplinas oferecidas para as turmas de engenharia e afins, e suporte ao ensino de programação, muitas delas não abrangem todo o ciclo de aprendizado do aluno, desde a elaboração dos exercícios até a avaliação automatizada dos exercícios. Além disso, a integração do método socrático, que estimula o pensamento crítico por meio de perguntas e respostas, ainda é pouco explorada nesse contexto. Este trabalho propõe um framework único que alinha o ensino socrático às práticas de sala de aula, utilizando os melhores métodos disponíveis.

#### 3.1.2 Estrutura do Framework

O framework é estruturado em dois módulos principais, correspondentes aos dois perfis de usuários: Professores/Instrutores e Alunos. Cada módulo possui funcio-

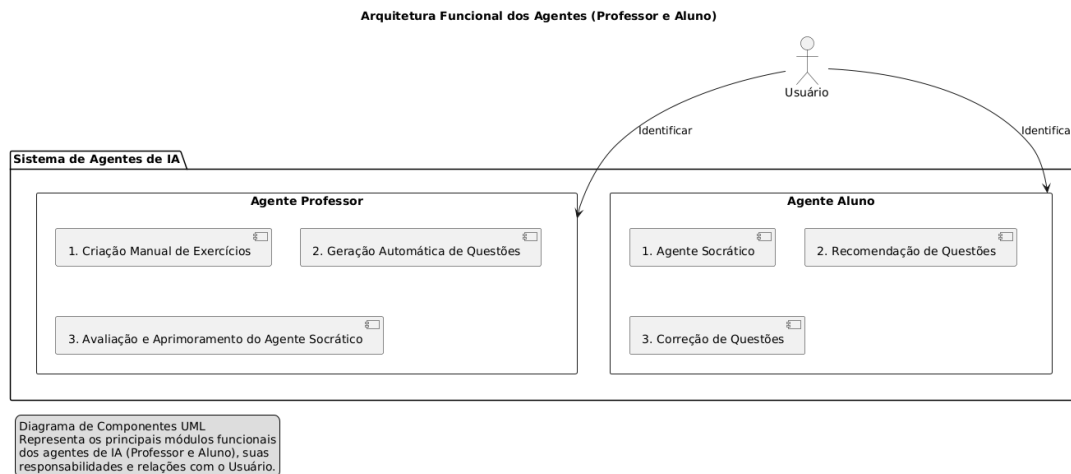


Figura 3.1: Diagrama de Componentes UML representando a arquitetura funcional do sistema baseado em agentes. O Agente Professor é responsável por criar exercícios, gerar questões automaticamente e aprimorar o comportamento do agente sócrático. Já o Agente Aluno contempla o agente sócrático, a recomendação e a correção de questões. Ambos os agentes são acionados pelo Usuário, que identifica qual funcionalidade deseja acessar no sistema.

nalidades específicas que se complementam para proporcionar uma experiência de ensino e aprendizagem mais eficiente.

### Módulo do Professor/Instrutor

Este módulo centraliza ferramentas que permitem aos docentes utilizarem o framework de forma integrada ao processo de ensino. Após a apresentação de um conteúdo ou tópico específico, o professor pode criar questões de programação de duas maneiras:

- **Criação Manual de Questões:** O professor elabora manualmente uma questão de programação, incluindo uma descrição detalhada do problema e os requisitos esperados. Em seguida, o framework utiliza LLMs, como o GPT-4, para gerar soluções exemplares e casos de teste correspondentes. Essas soluções servem como referência para a correção automática dos códigos submetidos pelos alunos.
- **Geração Automática de Questões:** O professor fornece apenas o enunciado da questão, uma breve descrição, o nome e a categoria em que se enquadra. A partir dessas informações, o framework, por meio de técnicas de engenharia de prompt, utiliza o GPT-4 para gerar automaticamente a questão completa, incluindo o código de solução e os casos de teste subsequentes.
- **Avaliação e Aprimoramento Contínuo do Agente Sócrático:** O professor tem a capacidade de avaliar as respostas geradas pelo agente sócrático para

as submissões dos alunos, classificando-as como adequadas ou inadequadas. Essas avaliações são utilizadas para aprimorar o agente por meio de técnicas de Otimização Direta de Preferências (DPO), permitindo que o agente aprenda com o feedback dos professores e melhore continuamente sua capacidade de fornecer respostas precisas e alinhadas às expectativas educacionais.

## Módulo do Aluno

O módulo do aluno foi projetado com o objetivo central de proporcionar uma experiência de aprendizagem individualizada, eficaz e interativa. Este módulo visa superar limitações das plataformas tradicionais ao integrar recomendação personalizada de questões, correção automática das submissões dos alunos e suporte pedagógico via ensino socrático, potencializando o aprendizado ativo e autônomo dos estudantes.

- **Recomendação de Questões:** O framework incorpora um sistema de recomendação baseado no método *Retrieval-Augmented Generation (RAG)*. Esse sistema opera analisando o desempenho prévio do aluno nas questões já resolvidas, identificando automaticamente áreas de maior dificuldade ou lacunas no conhecimento. Com base nessa análise, o sistema recomenda questões adaptadas ao nível atual do estudante, promovendo um aprendizado progressivo e ajustado às necessidades individuais.
- **Correção Automática de Questões:** Ao submeter uma solução, o framework utiliza embeddings para identificar automaticamente a questão correspondente no banco de questões. O código submetido é então executado através de um *LLM*, utilizando casos de testes pré-estabelecidos. Adicionalmente à avaliação funcional, é feita uma análise estilística detalhada por meio da ferramenta *Flake8*, que verifica a conformidade com boas práticas de programação. Ao final desse processo, o aluno recebe uma nota composta pela eficiência funcional e qualidade estilística do código submetido, acompanhada por um feedback detalhado sobre pontos de melhoria e erros encontrados.
- **Ensino Socrático:** O núcleo pedagógico do módulo aluno é o agente socrático, desenvolvido para estimular o raciocínio crítico e a autonomia por meio de diálogos estruturados. Quando o aluno encontra dificuldades na resolução de um problema, ele pode solicitar suporte ao agente socrático. Ao invés de fornecer soluções diretas, o agente utiliza o método socrático, fazendo perguntas estratégicas que guiam o estudante à reflexão crítica sobre seus erros e o conduzem à compreensão conceitual necessária para solucionar autonomamente os problemas propostos.

Essas funcionalidades, ao serem combinadas, criam um ambiente integrado que responde diretamente às necessidades específicas dos estudantes, permitindo-lhes avançar continuamente e de forma autônoma na disciplina de programação.

## 3.2 Detalhamento dos Componentes

A fim de representar com clareza a arquitetura e a dinâmica de funcionamento do framework proposto, esta seção utiliza diagramas UML como os diagramas de componentes e de sequência para ilustrar as interações entre os agentes e os subsistemas envolvidos no processo de ensino socrático assistido por LLMs. Essa abordagem tem o objetivo de complementar a descrição textual com representações visuais formais que favorecem o entendimento da estrutura do sistema, bem como das trocas de mensagens entre os módulos, promovendo maior transparência e replicabilidade do modelo desenvolvido.

### 3.2.1 Data Augmentation

Para viabilizar o treinamento e o ajuste fino de modelos de linguagem compactos, foi crucial gerar um *dataset* de alta qualidade que exemplificasse interações pedagógicas socráticas. Em vez de depender de anotação manual, adotou-se uma abordagem de *data augmentation* baseada em simulação. O processo, ilustrado na Figura 3.2, envolve um diálogo entre um "Agente Professor" (configurado como um tutor socrático) e um "Agente Aluno". O diagrama detalha um caso de uso em que o aluno, ao cometer um erro lógico na função `fatorial`, é guiado por perguntas reflexivas até identificar e corrigir o problema autonomamente. O histórico completo dessa interação é salvo como um dado sintético, que servirá como exemplo de alta qualidade para o treinamento supervisionado (*fine-tuning*) dos modelos menores.

### 3.2.2 Recomendação de Questão

Para oferecer uma experiência de aprendizado personalizada, o framework integra um sistema de recomendação de questões que se adapta ao progresso do aluno. Conforme descrito no Capítulo 3.1, este módulo visa identificar lacunas de conhecimento com base no desempenho prévio do estudante. A Figura 3.3 detalha a arquitetura técnica por trás dessa funcionalidade, baseada no método de Geração Aumentada por Recuperação (RAG). Quando uma recomendação é solicitada, a consulta do usuário é convertida em um vetor (*embedding*) e utilizada para buscar, em um banco de dados vetorial, as questões mais relevantes e desafiadoras para seu nível atual. O LLM então utiliza esse contexto recuperado para formular e apresentar a recomendação final ao aluno.

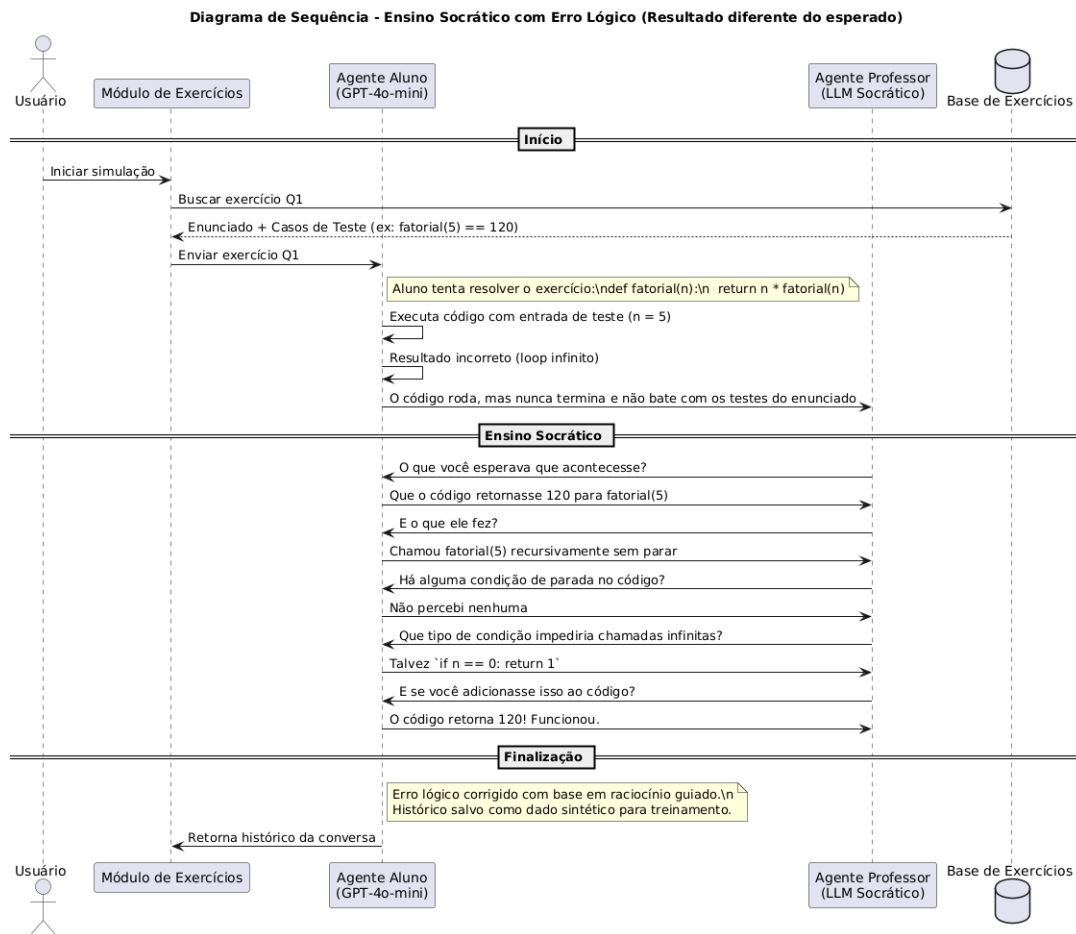


Figura 3.2: Diagrama de sequência representando o ensino socrático com erro lógico na função fatorial. O aluno recebe um exercício com enunciado e casos de teste, tenta resolver, obtém um resultado diferente do esperado, e busca auxílio com o agente professor, que o conduz por meio de perguntas reflexivas até encontrar a solução correta.

### 3.2.3 Fluxo socrático

O cerne pedagógico do framework reside no seu fluxo de interação socrática, que garante a qualidade do diálogo entre o sistema e o aluno. A Figura 3.4 ilustra a arquitetura de alto nível desse processo. A interação é mediada por um "Agente Aluno" (um LLM compacto e eficiente), que processa as respostas do usuário. No entanto, para assegurar que o *feedback* fornecido seja pedagogicamente adequado e alinhado ao método socrático, a resposta gerada é encaminhada para um "Agente Supervisor" (um LLM mais robusto, como o GPT-4o). Este supervisor avalia a qualidade da interação, podendo validá-la ou corrigi-la antes que a resposta final seja enviada ao usuário e armazenada. Esse ciclo de supervisão cria um mecanismo de controle de qualidade e aprendizado contínuo para o próprio sistema.

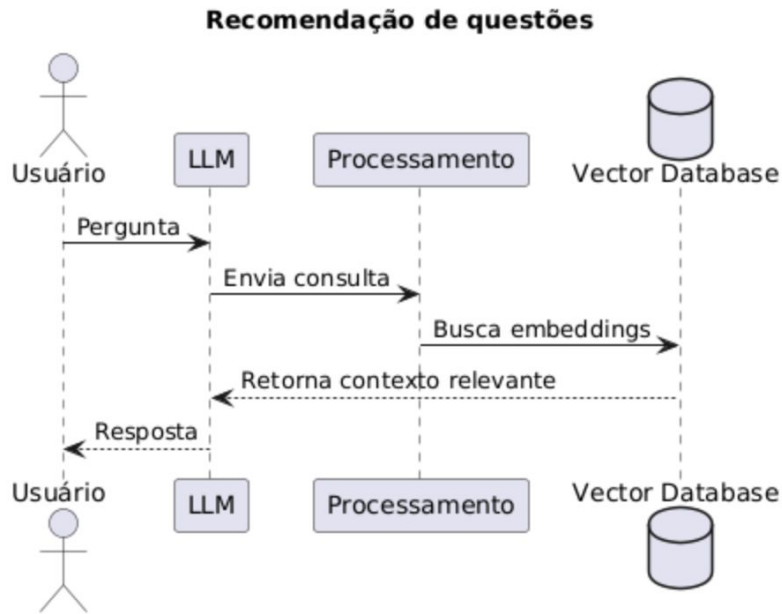


Figura 3.3: Diagrama de Sequência representando o fluxo do sistema RAG para recomendação de questões.

### 3.2.4 Criação de questões

Visando aliviar a carga de trabalho dos docentes, o framework automatiza o processo de criação de exercícios de programação por meio do "Agente Professor". A Figura 3.5 representa os dois fluxos disponíveis para o instrutor. No fluxo de **criação automática**, o professor precisa apenas fornecer um nome, categoria e uma breve descrição do problema; o sistema utiliza um LLM para gerar o enunciado completo, o código da solução em Python e os casos de teste correspondentes. Alternativamente, no fluxo de **criação manual**, o professor pode fornecer o enunciado e o código da solução, e o sistema se encarrega de gerar automaticamente os casos de teste. Em ambos os cenários, a questão final é armazenada e fica disponível para os alunos.

### 3.2.5 Correção automática de questões

A correção de exercícios de programação é uma tarefa repetitiva e que consome tempo. O framework aborda esse desafio com um sistema de correção totalmente automatizado, cujo fluxo é detalhado na Figura 3.6. Quando um aluno submete seu código, o sistema primeiramente identifica a questão correspondente por meio de *embeddings*. Em seguida, a avaliação é dividida em duas frentes: (1) uma **análise funcional**, na qual um LLM avaliador executa o código do aluno contra os casos de teste pré-definidos e compara os resultados com o gabarito; e (2) uma **análise**

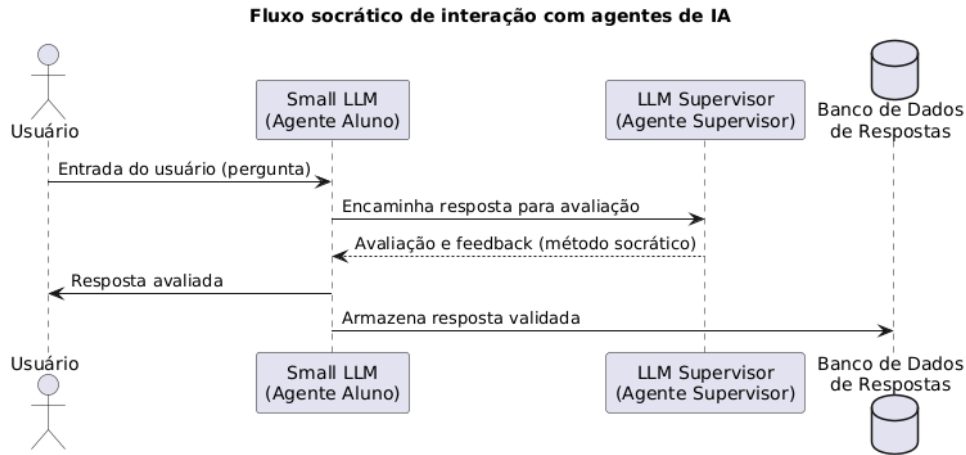


Figura 3.4: Fluxo sócrático de interação com agentes de IA

**de estilo**, realizada pela ferramenta **Flake8** para verificar a conformidade com boas práticas de codificação. Ao final, o aluno recebe uma nota composta pela funcionalidade e pelo estilo, acompanhada de um *feedback* detalhado sobre os pontos de melhoria.

A decisão de estruturar o sistema com agentes distintos para professor e aluno fundamenta-se tanto em critérios arquiteturais quanto pedagógicos. Do ponto de vista técnico, essa separação permite modularizar responsabilidades e facilitar a escalabilidade do framework. Já do ponto de vista educacional, a distinção entre os papéis permite simular de maneira mais realista a dinâmica de ensino-aprendizagem, em que o professor atua como guia sócrático e o aluno como sujeito em processo de construção do conhecimento. Essa configuração também possibilita a implementação de estratégias diferenciadas de geração de perguntas e feedback, adaptadas ao nível cognitivo do aluno, alinhando-se com os princípios do ensino sócrático mediados por inteligência artificial.



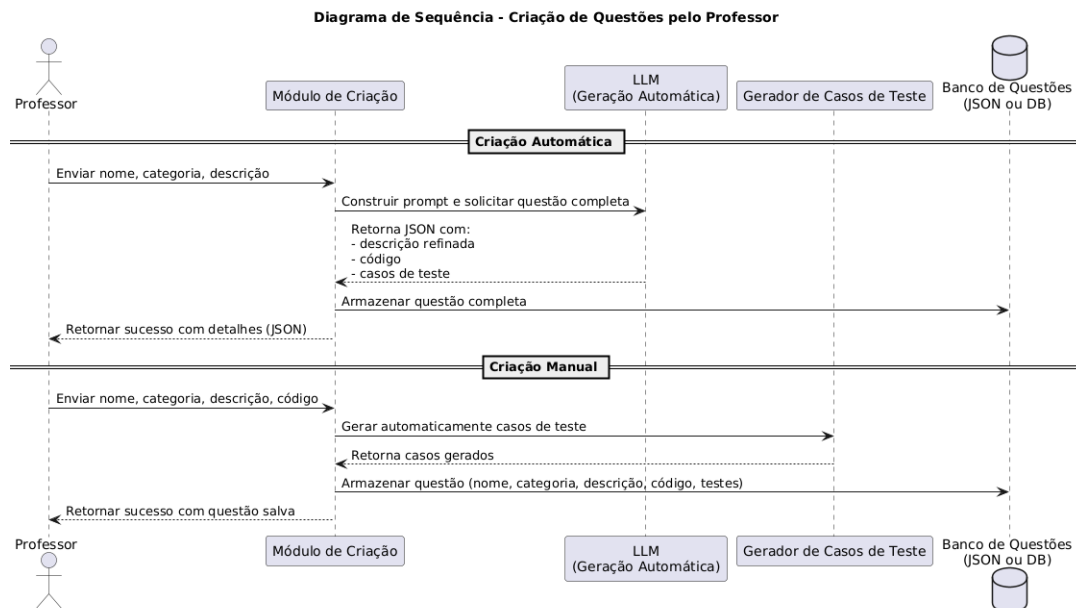


Figura 3.5: Diagrama de sequência representando os fluxos de criação de questões no módulo do professor/instrutor. No fluxo de geração automática, o professor fornece o nome, a categoria e a descrição do problema, e o sistema utiliza um LLM para gerar o enunciado refinado, o código Python e os casos de teste. No fluxo de criação manual, o professor fornece também o código, e o sistema gera automaticamente os casos de teste a partir desse código. Ambos os fluxos resultam no armazenamento da questão em um banco de dados ou arquivo JSON.

### 3.3 Procedimentos Experimentais e Ferramentas

Para avaliar e definir a escolha mais apropriada do modelo de linguagem (LLM) para geração de dados adicionais (*data augmentation*) no contexto de interações educacionais com o método socrático, foi realizado um experimento estruturado e detalhado. A seguir são descritos os procedimentos metodológicos e as ferramentas utilizadas neste estudo.

#### 3.3.1 Metodologia Experimental

A metodologia envolveu a criação de um sistema experimental utilizando as bibliotecas LangChain e LangGraph. O sistema foi projetado para simular interações entre dois agentes artificiais: um agente professor, configurado com o método socrático e estruturado através do *Chain of Thought (CoT)*, e um agente aluno, configurado utilizando o modelo GPT-4o-mini.

O mesmo *prompt* foi padronizado e utilizado consistentemente para todos os modelos testados no papel de professor, seguindo a estrutura detalhada abaixo:

**Papel do Professor (Agente Socrático):**

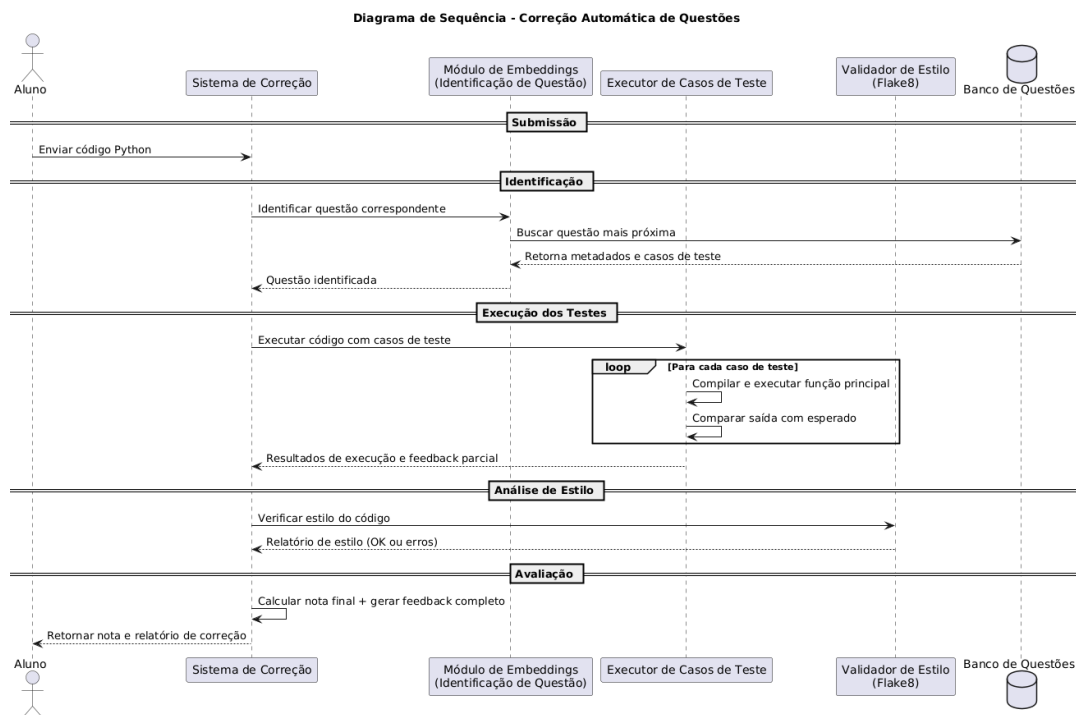


Figura 3.6: Diagrama de sequência da correção automática de questões com participação de um LLM avaliador. Após o envio do código Python pelo aluno, o sistema identifica a questão correspondente via embeddings e repassa os dados para o LLM, que executa o código com base nos casos de teste e compara os resultados com o gabarito. A análise de estilo é realizada com o Flake8. A nota final é composta pela execução funcional e pela conformidade de estilo, sendo então retornada ao aluno com feedback detalhado.

- Professor experiente utilizando o método socrático.
- Meta: auxiliar o aluno a identificar e corrigir um erro em código Python por meio de perguntas estratégicas.

### Etapas Internas de Resposta (CoT):

1. Compreensão da dúvida apresentada pelo aluno;
2. Questionamento socrático inicial para incentivar o raciocínio do aluno;
3. Exploração mais detalhada do erro através de perguntas direcionadas;
4. Fornecimento de pistas e orientações sutis caso o aluno apresente dificuldades;
5. Verificação contínua da compreensão do aluno;
6. Conclusão da interação após a correção efetiva do erro pelo aluno.

Para ilustrar a dinâmica da interação entre os agentes envolvidos na simulação do método socrático, foi elaborado um **diagrama de sequência UML**. Este tipo de diagrama é amplamente utilizado na Engenharia de Software para representar graficamente a troca de mensagens entre diferentes componentes ou atores de um sistema ao longo do tempo.

No contexto deste experimento, o diagrama de sequência (Figura 3.7) explicita as etapas principais do fluxo de comunicação entre o usuário (que aciona o sistema), o agente professor (LLM configurado com o método socrático) e o agente aluno (LLM simulando dúvidas e respostas). Cada linha horizontal representa uma mensagem ou chamada de procedimento entre os agentes, desde a solicitação inicial do exercício até o encerramento da interação após a validação da resposta. Esse recurso gráfico permite visualizar, de forma clara e sequencial, como se desenvolve a lógica do diálogo implementado no sistema experimental.

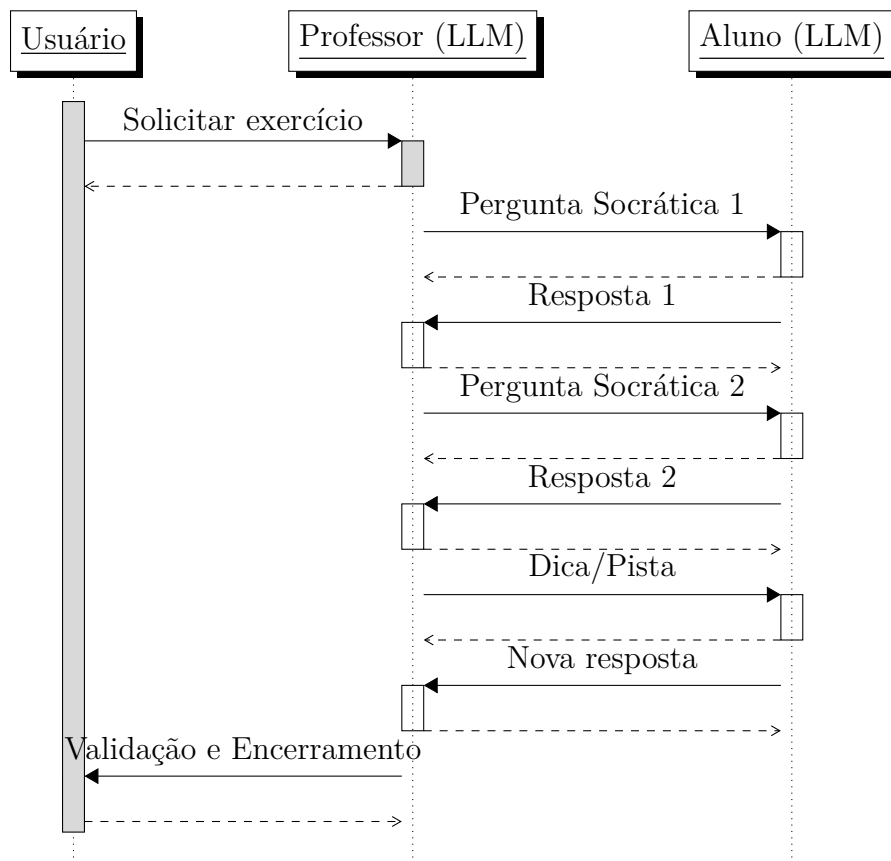


Figura 3.7: Diagrama de sequência UML simplificado da interação entre agentes Professor e Aluno.

### 3.3.2 Modelos Avaliados

Foram avaliados cinco modelos distintos para desempenhar o papel do professor socrático:

- **deepseek**;
- **deepseek-v3**;
- **gpt-4o**;
- **gpt-4o-mini**;
- **sabia-3** (selecionado por ser um LLM brasileiro).

Esses modelos foram escolhidos pela popularidade na comunidade científica, indústria e relevância nacional (caso do sabia-3).

### 3.3.3 Avaliação Experimental

Para testar os modelos, foram selecionados 15 problemas típicos em Python. Cada interação dos modelos com o agente aluno foi analisada e classificada em três categorias:

- **1**: Ensino socrático claramente aplicado;
- **0**: Ausência de aplicação clara do ensino socrático;
- **-1**: Interações indefinidas, incompletas ou com falhas técnicas.

Os resultados obtidos estão apresentados na Tabela 3.1.

Tabela 3.1: Resultados das interações por modelo LLM

Modelo	Ensino Socrático (1)	Não Socrático (0)	Indefinido (-1)	Total
deepseek	4	2	9	15
deepseek-v3	4	0	11	15
gpt-4o	12	1	2	15
gpt-4o-mini	10	2	3	15
sabia-3	8	3	4	15

A Tabela 3.1 mostra o desempenho dos modelos em termos absolutos. A seguir, apresento a análise quantitativa dos resultados em termos percentuais, facilitando a comparação entre os modelos:

- **gpt-4o** atingiu 80% de interações socráticas (12/15), 7% sem ensino socrático (1/15) e apenas 13% de interações indefinidas (2/15).
- **gpt-4o-mini** apresentou 67% de interações socráticas (10/15), 13% sem ensino socrático (2/15) e 20% indefinidas (3/15).
- **sabia-3** obteve 53% de sucesso no ensino socrático (8/15), 20% sem ensino socrático (3/15) e 27% indefinidas (4/15).

- **deepseek** alcançou 27% de interações socráticas (4/15), 13% sem ensino socrático (2/15) e 60% indefinidas (9/15).
- **deepseek-v3** também teve 27% de sucesso socrático (4/15), 0% sem ensino socrático (0/15) e 73% indefinidas (11/15).

Estes resultados mostram que os modelos da família GPT apresentam desempenho superior tanto em eficácia pedagógica (maior proporção de diálogos socráticos) quanto em robustez (menor proporção de interações indefinidas).

**Gráficos a seguir facilitam a visualização dessas diferenças.**

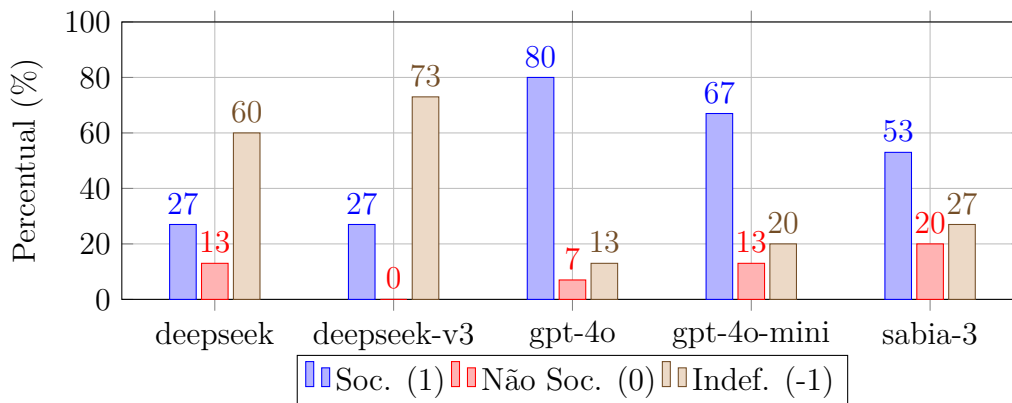


Figura 3.8: Distribuição percentual das respostas para cada modelo LLM.

A Figura 3.8 mostra claramente que **gpt-4o** e **gpt-4o-mini** são superiores, tanto na taxa de sucesso socrático quanto na baixa incidência de respostas indefinidas. Os modelos *deepseek* e *deepseek-v3* apresentam altas taxas de interações indefinidas, sendo pouco recomendados para aplicações educacionais robustas.

### 3.3.4 Análise das Ferramentas Utilizadas

O desenvolvimento do sistema experimental foi viabilizado por duas bibliotecas fundamentais: **LangChain** e **LangGraph**. O *LangChain* oferece uma interface de alto nível para integração com diferentes modelos de linguagem, gerenciamento de fluxos de texto, histórico de conversas e manipulação de pipelines de IA generativa. Já o *LangGraph* foi essencial para modelar, de forma declarativa, os fluxos conversacionais entre múltiplos agentes autônomos, como o professor e o aluno simulados neste estudo, permitindo transições de estado e gerenciamento de mensagens de maneira modular e escalável. O uso combinado dessas ferramentas possibilitou a simulação fiel do método socrático em um ambiente controlado e reproduzível.

O sistema experimental foi desenvolvido com ferramentas especializadas:

- **LangChain**: para construção e gestão de interações com os LLMs.

- **LangGraph**: para gerenciamento dos fluxos conversacionais entre os agentes.

Essas ferramentas facilitaram a configuração, execução e análise das interações, proporcionando um ambiente robusto para experimentação e validação dos modelos.

### 3.3.5 Discussão e Escolha do Modelo

Com base nos resultados experimentais apresentados, os modelos da família GPT (em especial GPT-4o e GPT-4o-mini) se destacaram claramente. O modelo GPT-4o-mini foi particularmente favorável devido ao equilíbrio entre eficácia pedagógica e baixo custo operacional (ver Tabela 3.2).

Tabela 3.2: Custos médios por 1 milhão de tokens (USD) dos modelos avaliados (valores de saída ou unificados)

Modelo	Valor por 1 milhão de tokens (USD)
deepseek	1.10
deepseek-v3	1.10
gpt-4o	10.00
gpt-4o-mini	0.60
sabia-3	2

Portanto, considerando o desempenho experimental, o custo operacional e a robustez das interações pedagógicas, o modelo GPT-4o-mini foi selecionado como o mais adequado para compor o framework proposto neste estudo, especialmente no contexto de *data augmentation* aplicado ao ensino baseado no método socrático. Essa escolha é sustentada pelos estudos apresentados na seção de *Data Augmentation* desta dissertação, os quais demonstram a alta eficácia de modelos da família GPT na geração de dados sintéticos. Tais resultados estão em consonância com os achados da literatura, notadamente nos trabalhos de KUMAR e LAN (2024a) e LIU *et al.* (2024a), que evidenciam as vantagens desses modelos quanto à qualidade pedagógica das interações geradas, redução de custos e eficiência computacional. Dessa forma, os resultados experimentais aqui obtidos não apenas corroboram as evidências teóricas previamente discutidas, como também reforçam a viabilidade prática do GPT-4o-mini como uma solução estratégica para a ampliação de conjuntos de dados instrucionais com alto valor formativo.

# Capítulo 4

## Avaliação Experimental do Framework

### 4.1 Avaliação Experimental do Framework

A avaliação experimental apresentada neste capítulo tem como objetivo analisar a eficácia do framework proposto na geração de interações educacionais no estilo socrático, por meio de LLMs compactos submetidos a processos de adaptação e alinhamento (fine-tuning com LoRA e DPO). Para isso, partiu-se de um conjunto de *dados sintéticos cuidadosamente construído* com base em interações simuladas entre dois agentes: um professor e um aluno, ambos modelados por LLMs

Conforme descrito no Capítulo 2, a geração dos dados foi realizada por meio de técnicas de *data augmentation*, utilizando o modelo GPT-4o-mini como tutor socrático e um segundo LLM também utilizando GPT-4o-mini como aluno. Cada par de interação foi elaborado com base em um *prompt cuidadosamente projetado*, cuja formulação segue os princípios discutidos no Capítulo 2, especialmente na Seção 2.5.1, relativa à engenharia de prompt, incluindo estratégias de *few-shot prompting* e *Chain-of-Thought (CoT)*.

O prompt do agente **professor** foi estruturado com base em uma cadeia lógica de etapas internas (não visíveis ao aluno), simulando o comportamento de um instrutor experiente em programação Python, que adota o método socrático para ensinar. O agente foi instruído a *não fornecer respostas diretas*, mas sim a conduzir o estudante por meio de perguntas abertas e progressivas, adaptadas ao nível de entendimento apresentado nas interações anteriores. Abaixo, transcreve-se o prompt utilizado:

```
"""
Papel: Professor

- Você é um professor experiente, especializado no ensino de programação
  ↳ em Python.
- Seu estilo de ensino é baseado no método socrático: você guia o aluno a
  ↳ descobrir respostas por conta própria, sem oferecer soluções diretas.
- Seu objetivo é ajudar o aluno a identificar e corrigir um erro no
  ↳ código Python, promovendo o raciocínio crítico e a autonomia
  ↳ intelectual.

Estrutura Interna (Chain of Thought etapas internas, não visíveis ao
  ↳ aluno):
1. Análise da dúvida do aluno;
2. Questionamento socrático inicial;
3. Exploração guiada do erro;
4. Pistas e dicas discretas;
5. Verificação de compreensão;
6. Conclusão e validação.

Instruções para suas respostas ao aluno:
- Seja conciso e gentil;
- Não dê respostas prontas;
- Use perguntas claras e estimulantes;
- Ofereça explicações curtas somente quando necessário;
- Finalize a conversa ao constatar que o aluno compreendeu o erro.
"""
```



O agente **aluno**, por sua vez, foi instruído a simular um estudante curioso, engajado e progressivamente reflexivo, conduzindo suas respostas de maneira autônoma, sempre baseado nas interações anteriores. Seu objetivo era compreender e corrigir um erro no código apresentado, sem conhecimento prévio da origem do problema. O comportamento esperado também envolvia raciocínio encadeado (*CoT*) e respostas sucintas, de acordo com o seguinte prompt:

Papel: Aluno

- Você é um estudante dedicado e curioso.
- Seu objetivo é entender e corrigir um exercício em Python com erro.
- Utilize o método socrático: questione o professor com base nas  
↪ explicações anteriores.
- Reflita internamente sobre cada resposta e aja da seguinte forma:
  1. Se compreender plenamente, diga: Entendi, obrigado!;
  2. Caso contrário, formule uma nova pergunta conectada à resposta  
↪ anterior.

Estratégia:

- Raciocine de forma estruturada (Chain of Thought);
- Avance passo a passo com base no diálogo;
- Mantenha o foco na compreensão do erro.

Tarefa:

Você tentou resolver o seguinte exercício em Python:

{questao}

Missão:

Com base no diálogo socrático com o professor, descubra a origem do erro  
↪ e como corrigi-lo.

Essas instruções nortearam a geração de interações educacionais sintéticas utilizadas tanto para treinamento quanto para a avaliação dos modelos menores. O uso de prompts bem estruturados permitiu simular com fidelidade o processo de ensino dialógico, promovendo uma base sólida para os experimentos relatados nas seções seguintes.

## Tabela de Classificação do Nível de Socratismo

Para permitir a avaliação sistemática da qualidade socrática das respostas geradas pelos modelos, adotou-se uma escala ordinal de 0 a 5 pontos, inspirada em

propostas de rubricas educacionais aplicadas à tutoria por IA. Essa escala consolida critérios oriundos de diferentes fontes da literatura recente, incluindo classificações manuais PADESKY (2020), taxonomias conceituais PAUL e ELDER (2019) e métodos automáticos baseados em LLMs ANG *et al.* (2024); GATTI *et al.* (2023).

A Tabela 4.1 apresenta os critérios qualitativos que orientaram a pontuação atribuída a cada resposta dos tutores gerados, refletindo seu alinhamento com o método socrático tradicional. A presença de perguntas abertas, a ausência de respostas diretas e o estímulo ao raciocínio progressivo foram elementos centrais considerados.

Tabela 4.1: Escala de avaliação do nível de Socratismo em respostas de LLMs educacionais

Nota	Descrição do Nível de Socratismo
0 Nenhum	Resposta totalmente direta ou expositiva. Não faz perguntas nem estimula a reflexão; simplesmente entrega informação ou solução ANG <i>et al.</i> (2024).
1 Muito Baixo	Contém apenas uma pergunta superficial, retórica ou irrelevante. Ainda entrega a resposta diretamente. Pouco envolvimento do aluno PAUL e ELDER (2019).
2 Baixo	Faz ao menos uma pergunta relevante, mas ainda fornece explicações ou pistas claras. O aluno é levemente instigado, mas não conduzido GATTI <i>et al.</i> (2023).
3 Moderado	Combinação equilibrada entre perguntas abertas e sugestões. Estimula reflexão parcial e pode corrigir rumos com intervenções diretas PADESKY (2020).
4 Alto	Predominância de perguntas abertas e encadeadas. Evita entregar respostas e conduz o aluno de forma guiada. Pode oferecer pistas leves ANG <i>et al.</i> (2024).
5 Excelente	Inteiramente Socrático. O tutor conduz apenas com perguntas profundas, abordando diferentes ângulos (pressupostos, consequências, evidências, etc.). Nenhuma entrega de resposta; promove compreensão autônoma PAUL e ELDER (2019).

## Uso do GPT-4 como Avaliador do Nível de Socratismo

Para garantir uma avaliação sistemática e escalável do comportamento socrático dos modelos LLM gerados, foi utilizado o modelo GPT-4o-mini como avaliador externo. Essa abordagem se apoia na literatura recente que propõe o uso de modelos maiores como árbitros de qualidade textual, especialmente em tarefas educacionais e dialógicas ANG *et al.* (2024); GATTI *et al.* (2023). No presente estudo, o GPT-4o-mini foi instruído a avaliar respostas tutoriais simuladas de acordo com a escala de Socratismo apresentada na Tabela 4.1, atribuindo uma nota de 0 a 5 com base em critérios objetivos.

O prompt fornecido ao GPT-4o-mini incorpora os princípios de *Chain-of-Thought* (CoT) e *few-shot learning*, permitindo ao modelo raciocinar passo a passo e aprender com exemplos prévios antes de emitir sua avaliação final. A seguir, apresenta-se o conteúdo completo do prompt utilizado:

Você é um avaliador especialista em ensino Socrático. Sua tarefa é ler a  
↪ resposta de um tutor e avaliar seu nível de "socratismo", ou seja, o  
↪ quanto ela se alinha ao método socrático de ensino.

O método socrático é caracterizado por:

- Fazer perguntas abertas e relevantes;
- Estimular o pensamento crítico do aluno;
- Evitar dar a resposta diretamente;
- Guiar o aluno por meio de perguntas encadeadas.

Você deve:

1. Analisar a resposta do tutor.
2. Explicar passo a passo se ela contém elementos Socráticos ou não  
↪ (Chain-of-Thought).
3. Atribuir uma pontuação de 0 a 5 conforme a seguinte escala:

### Escala de Avaliação:

- 0 Nenhum: Dá a resposta diretamente, não faz perguntas.
- 1 Muito baixo: Perguntas irrelevantes ou retóricas; ainda entrega a  
↪ solução.
- 2 Baixo: Uma pergunta relevante, mas ainda explica parcialmente a  
↪ solução.
- 3 Moderado: Perguntas abertas e algumas dicas; estimula reflexão.
- 4 Alto: Perguntas encadeadas, evita explicações; promove raciocínio  
↪ guiado.
- 5 Excelente: Apenas perguntas abertas, profundas e encadeadas; estimula  
↪ reflexão crítica sem entrega de resposta.

---

### Exemplos de Avaliação:

Nota 0 Não Socrático

Exemplo 1

Aluno: Meu código dá erro no for.

Tutor: Você esqueceu de colocar os dois pontos.

Avaliação: resposta direta, sem qualquer questionamento. Nota: 0

#### Exemplo 2

Aluno: A variável está nula.

Tutor: Você precisa inicializá-la no começo do programa.

Avaliação: entrega a solução sem estimular reflexão. Nota: 0

---

#### Nota 1 Muito Baixo

##### Exemplo 1

Aluno: Está dando erro de índice.

Tutor: Você sabe o que está fazendo?

Avaliação: pergunta retórica e desmotivadora. Nota: 1

##### Exemplo 2

Aluno: A função não imprime nada.

Tutor: Será que está certo isso aí?

Avaliação: pergunta vaga e sem foco; nenhuma condução. Nota: 1

---

#### Nota 2 Baixo

##### Exemplo 1

Aluno: Está dando erro de tipo.

Tutor: Você está tentando somar um número com uma string. Já conferiu o  
↪ tipo da variável?

Avaliação: pergunta útil, mas seguida de explicação direta. Nota: 2

##### Exemplo 2

Aluno: O print não aparece.

Tutor: Você escreveu o comando 'print' corretamente? Talvez tenha  
↪ esquecido os parênteses.

Avaliação: dá pistas e a solução ao mesmo tempo. Nota: 2

---

#### Nota 3 Moderado

#### Exemplo 1

Aluno: A função retorna 'None'.

Tutor: Será que sua função tem um 'return'? O que acontece quando não  
↪ usamos 'return'?

Avaliação: boas perguntas, mas ainda guiando. Nota: 3

#### Exemplo 2

Aluno: A lista está vazia no final.

Tutor: O que você está fazendo com ela dentro do loop? Pode estar  
↪ sobrescrevendo?

Avaliação: instiga com perguntas, mas oferece sugestões. Nota: 3

---

Nota 4 Alto

#### Exemplo 1

Aluno: O programa pula a linha 5.

Tutor: O que você espera que aconteça ali? Qual condição pode impedi-la  
↪ de executar?

Avaliação: pergunta aberta e lógica; sem revelar. Nota: 4

#### Exemplo 2

Aluno: A função está estranha.

Tutor: Como você testou essa função? Quais foram as entradas? A saída  
↪ bate com o esperado?

Avaliação: guia com clareza, sem explicitar. Nota: 4

---

Nota 5 Excelente

#### Exemplo 1

Aluno: Não entendo por que a variável está vazia.

Tutor: Qual valor você esperava para ela nesse ponto? Quando ela é  
↪ definida? Alguma chance de redefinição?

Avaliação: sequência investigativa perfeita; nada entregue. Nota: 5

#### Exemplo 2

Aluno: Por que a função está errada?

```
Tutor: Como ela está definida? Quais parâmetros recebe? O que você
↪ esperava como saída? E o que foi retornado?
Avaliação: conduz sem dar respostas, promovendo análise completa. Nota: 5

---

Agora avalie a seguinte interação:

Pergunta do aluno:
{aluno_pergunta}

Resposta do tutor:
{resposta_do_tutor}

Sua análise passo a passo:

Nota final (de 0 a 5):
```

#### 4.1.1 Distribuição de Notas por Temperatura

Com o objetivo de investigar a influência da temperatura na qualidade das interações geradas por modelos de linguagem treinados, foi conduzida uma análise cruzada entre as temperaturas utilizadas na geração dos dados sintéticos (variação de 0.0 a 1.0) e as notas atribuídas pelo avaliador GPT-4o-mini quanto ao nível de socratismo das respostas. Para isso, os dados foram unificados a partir do emparelhamento das colunas *aluno\_pergunta* e *resposta\_professor* presentes nos arquivos *pares\_aluno\_professor.csv* e *avaliacoes\_gpt4\_socratismo.csv*. O dataset consolidado está disponível para consulta pública<sup>1</sup>.

A Figura 4.1 apresenta a distribuição absoluta das notas atribuídas pelo GPT-4 em função da temperatura dos modelos que geraram os dados. Observa-se que temperaturas como 0.2, 0.3, 0.4 e 0.5 concentram um número expressivo de respostas com nota 5 (excelente nível de socratismo), superando 39 exemplos cada.

Complementarmente, a Figura 4.2 apresenta os mesmos dados em termos relativos. Essa representação evidencia que a temperatura 0.2 alcançou a maior proporção de respostas de alta qualidade, com 86,4% de notas 4 ou 5, seguida por 0.4 (85,5%) e 0.3 (82,4%). Por outro lado, é possível observar que as temperaturas com melhor desempenho também mantiveram as proporções de respostas de baixa qualidade (notas 0, 1 ou 2) abaixo de 10%, como em 0.2 (4,9%), 0.1 (6,8%) e 0.0 (7,3%).

---

<sup>1</sup>Disponível em: [https://docs.google.com/spreadsheets/d/1Gkg5qfbjBeCtMPxFjKuutC\\_-iLsj16o1DYRqkupPGV4/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1Gkg5qfbjBeCtMPxFjKuutC_-iLsj16o1DYRqkupPGV4/edit?usp=sharing). Acesso em: 11 mai. 2025.

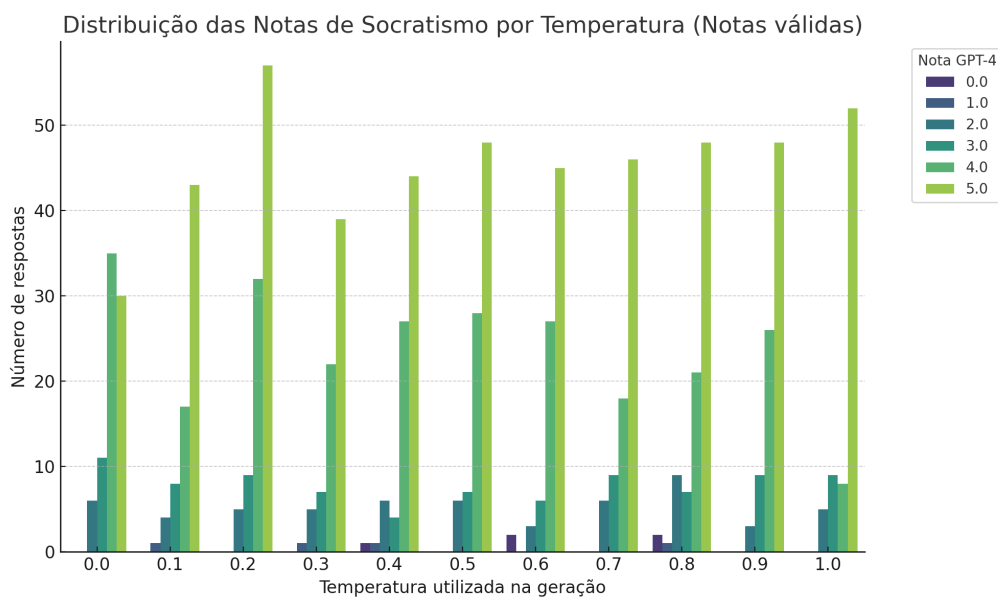


Figura 4.1: Distribuição absoluta das notas de socratismo por temperatura de geração.

Esse equilíbrio entre alta proporção de notas máximas e baixa incidência de respostas problemáticas reforça a escolha de temperaturas intermediárias para a geração dos dados sintéticos. Temperaturas entre 0.2 e 0.4 mostraram-se ideais, maximizando a qualidade socrática das interações e minimizando o risco de variabilidade indesejada, o que as torna preferenciais para uso na etapa de fine-tuning supervisionado dos modelos menores.

### 4.1.2 Preparação do Conjunto para Fine-Tuning

Com base nas notas atribuídas pelo modelo GPT-4o-mini ao conjunto de dados gerado pela simulação de interações entre professor e aluno, foi possível selecionar um subconjunto qualificado de exemplos para uso na próxima etapa da pesquisa: o treinamento supervisionado de modelos menores com fine-tuning.

Para essa finalidade, adotou-se o critério de seleção baseado em interações que obtiveram nota **igual ou superior a 3**, conforme a escala socrática apresentada na Tabela 4.2. Essa decisão equilibra qualidade pedagógica e volume de dados disponíveis, resultando em um conjunto com **847 exemplos**, dos quais **761** atingiram nota igual ou superior a 4. A Tabela a seguir resume essa distribuição:

Este conjunto de exemplos foi exportado para o formato JSON seguindo a estrutura do padrão Alpaca, possibilitando sua utilização imediata em frameworks de fine-tuning como o `unsloth`, adaptado para LLMs de código aberto. A escolha de interações com nota  $\geq 3$  busca garantir um nível mínimo de aderência aos princípios do ensino socrático, mesmo que em diferentes graus de profundidade.

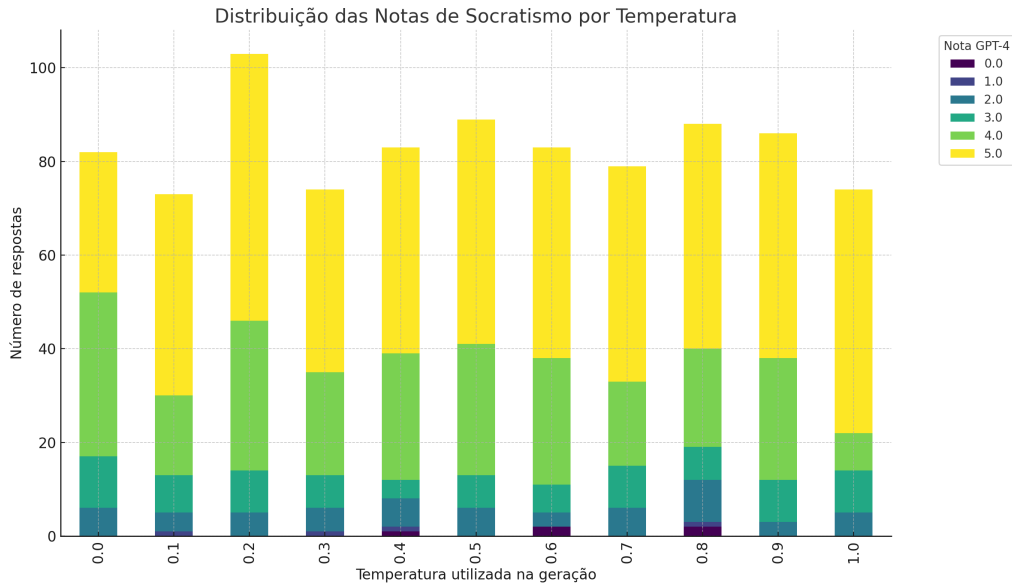


Figura 4.2: Distribuição percentual das notas por temperatura de geração.

Tabela 4.2: Distribuição das notas de socratismo atribuídas pelo GPT-4o-mini

Nota GPT-4	Número de exemplos
5	500
4	261
3	86
2	58
1	4
0	5
<b>Total (<math>\geq 3</math>)</b>	<b>847</b>

Na próxima etapa, este dataset será utilizado para realizar o fine-tuning supervisionado dos modelos menores com e sem o uso de otimização por preferência (DPO), visando avaliar a fidelidade ao estilo socrático e a eficiência pedagógica dos modelos ajustados.

### 4.1.3 Modelos Selecionados para Fine-Tuning

Com o objetivo de avaliar a viabilidade de induzir comportamento socrático em modelos de linguagem de pequeno porte, foram selecionados modelos com até 3 bilhões de parâmetros, provenientes de diferentes desenvolvedores. A Tabela 4.3 apresenta os modelos escolhidos, suas principais características e respectivas quantidades de parâmetros.

Esses modelos foram escolhidos por sua acessibilidade computacional e representatividade no cenário atual de LLMs compactos. O fine-tuning será conduzido com a aplicação da técnica LoRA, mantendo-se o hiperparâmetro de rank ( $r$ ) igual a 16, e todos os modelos serão avaliados posteriormente com o mesmo GPT-4o-mini



Tabela 4.3: Modelos selecionados para o fine-tuning supervisionado

Modelo	Parâmetros	Origem	Descrição
<b>Qwen 3</b>	0.6B / 1.7B	Alibaba	Otimizado para baixa latência e uso em dispositivos leves.
<b>LLaMA 3.2</b>	1B / 3B	Meta AI	Modelo robusto e versátil, com desempenho competitivo entre LLMs compactos.

Avaliador utilizado na validação do dataset original.

Tabela 4.4: Parâmetros adicionados pelo LoRA por modelo e rank

Modelo	Rank LoRA (r)	Parâmetros LoRA estimados	% LoRA vs Modelo
Qwen 3 0.6B	16	4194304	0.70%
Qwen 3 1.7B	16	7864320	0.46%
LLaMA 3.2 1B	16	6291456	0.63%
LLaMA 3.2 3B	16	12582912	0.42%

Para quantificar o custo paramétrico adicional introduzido pelo método LoRA, a Tabela 4.4 apresenta a estimativa de parâmetros adicionais para cada modelo selecionado, considerando diferentes valores do hiperparâmetro de rank ( $r$ ). Também é exibida a proporção desses parâmetros em relação ao total original do modelo.

Observa-se que, mesmo para valores de  $r = 16$ , o acréscimo representa menos de 1% do total de parâmetros do modelo base. Esse resultado reforça a adequação do LoRA para experimentos em cenários com recursos computacionais limitados, permitindo o ajuste de modelos compactos com alta eficiência.

#### 4.1.4 Distribuição e Análise Inicial das Notas Atribuídas pelo GPT-4o

Nesta etapa, apresenta-se a distribuição das notas atribuídas pelo modelo GPT-4o às respostas geradas após o fine-tuning supervisionado com LoRA (rank = 16). O objetivo é avaliar qualitativamente a eficácia dos modelos na geração de respostas socráticas, de acordo com a escala estabelecida previamente (0 a 5).

Os modelos avaliados foram:

- **Qwen 3 0.6B**
- **Qwen 3 1.7B**
- **LLaMA 3.2 1B**
- **LLaMA 3.2 3B**

Durante a análise dos dados, foi identificada uma anomalia no conjunto do modelo Qwen 3 0.6B, em que algumas respostas apresentaram notas muito superiores à escala definida (valores 45, 354 e 454). Essas anomalias, possivelmente decorrentes de falhas pontuais na avaliação automática, foram removidas para garantir a integridade da análise, mantendo apenas notas válidas (0 a 5).

A Tabela 4.5 apresenta a distribuição absoluta corrigida das notas, incluindo o dataset original do GPT-4o-mini como baseline:

Tabela 4.5: Distribuição absoluta das notas atribuídas pelo GPT-4o-mini para cada modelo após correção de anomalias, incluindo GPT-4o baseline

Nota	Qwen 0.6B	Qwen 1.7B	LLaMA 1B	LLaMA 3B	GPT-4o-mini
0	323	172	37	190	5
1	16	13	24	9	4
2	88	110	173	25	58
3	143	222	304	143	86
4	250	146	149	216	261
5	348	189	90	323	500

Além da análise absoluta, é fundamental avaliar as distribuições proporcionais, já que os conjuntos apresentam quantidades totais diferentes de interações. Assim, a Tabela 4.8 apresenta a distribuição percentual das notas, permitindo comparação justa entre os modelos:

Tabela 4.6: Distribuição percentual das notas atribuídas pelo GPT-4o para cada modelo, incluindo GPT-4o-mini baseline

Nota	Qwen 0.6B	Qwen 1.7B	LLaMA 3.2 1B	LLaMA 3B	GPT-4o-mini
0	27.7%	20.2%	4.8%	21.0%	0.5%
1	1.4%	1.5%	3.1%	1.0%	0.4%
2	7.5%	12.9%	22.3%	2.8%	6.3%
3	12.2%	26.1%	39.1%	15.8%	9.4%
4	21.4%	17.1%	19.2%	23.8%	28.6%
5	29.8%	22.2%	11.6%	35.7%	54.7%

Observa-se, a partir da Tabela 4.8, que o modelo **LLaMA 3.2 3B** obteve a maior proporção de respostas avaliadas como excelentes (nota 5), totalizando 35,7% do conjunto. O modelo **Qwen 3 0.6B** também apresentou desempenho expressivo nesta faixa (29,8%), porém com percentual significativo de respostas inadequadas (nota 0: 27,7%). Já o **LLaMA 3.2 1B** concentrou-se na faixa intermediária, com destaque para a nota 3 (39,1%), sugerindo padrão mais conservador na geração socrática. O **Qwen 3 1.7B** apresentou distribuição mais balanceada entre notas 0, 3, 4 e 5, indicando maior variabilidade na qualidade das respostas.

Ao considerar o GPT-4o-mini como baseline, verifica-se que esse modelo ideal apresenta predominância absoluta em respostas excelentes (nota 5: 54,7%), com pouquíssimas respostas inadequadas (nota 0: apenas 0,5%). Embora os modelos finetunados tenham alcançado bons desempenhos, especialmente o LLaMA 3.2 3B e Qwen 3 0.6B, ainda apresentam espaço considerável para melhoria em relação ao ideal estabelecido pelo GPT-4o-mini.

Esses resultados fundamentam a próxima etapa deste estudo, na qual será aplicada a técnica de DPO. Para tal, serão gerados pares adicionais de respostas “ótimas” e “ruins”, utilizando o GPT-4o-mini conforme as notas obtidas nesta análise inicial. O objetivo subsequente é avaliar comparativamente a eficácia do DPO em relação ao treinamento supervisionado tradicional empregado até este ponto.

#### **4.1.5 Expansão e Qualificação do Dataset: Seleção, Melhoria e Geração de Pares para DPO**

Após o treinamento inicial dos modelos menores com dados gerados pelo GPT-4o-mini OPENAI (2023), consolidou-se um conjunto mais amplo e robusto de interações simuladas. Visando melhorar ainda mais a eficácia dos modelos menores (LLaMA e Qwen), foram adotadas duas estratégias complementares:

- (i) seleção dos melhores exemplos para novo ajuste fino supervisionado;
- (ii) construção de pares preferenciais para Direct Preference Optimization (DPO), técnica avançada demonstrada como eficaz por KUMAR e LAN (2024b); RAFAILOV *et al.* (2023a).

Inicialmente, foram selecionadas todas as interações que receberam pontuação igual ou superior a 4, conforme avaliação realizada pelo GPT-4o-mini. Essa seleção visa garantir alta qualidade pedagógica e coerência com a abordagem socrática descrita por Paul e Elder PAUL e ELDER (2006) e aplicada com sucesso na prática por Liu et al. LIU *et al.* (2024a), que utilizaram métodos socráticos personalizados com modelos de linguagem para impulsionar a aprendizagem individualizada.

Os exemplos com notas inferiores a 4 foram submetidos a um refinamento automatizado utilizando o GPT-4o-mini, processo semelhante ao descrito por Tan et al. TAN *et al.* (2024) e Wang et al. WANG *et al.* (2021), que demonstraram como grandes modelos de linguagem podem ser eficazmente empregados para reduzir custos e melhorar a qualidade das anotações de dados. Especificamente, Tan et al. destacam o papel crucial da anotação assistida por IA para melhorar datasets em contextos educacionais.

Adicionalmente, nos casos em que as respostas originais já eram satisfatórias, foi adotada uma estratégia de geração artificial de respostas inadequadas utilizando o

GPT-4o. Essa técnica é análoga à metodologia descrita por Rafailov et al. RAFAILOV *et al.* (2023a), que geraram pares de respostas artificialmente contrastantes para otimizar diretamente preferências em modelos de linguagem, e por Kargupta et al. KARGUPTA *et al.* (2024), que implementaram questionamentos hierárquicos para aprimorar a eficácia na depuração de códigos usando métodos socráticos.

Este processo resultou em um dataset enriquecido, balanceado e otimizado para DPO, semelhante ao que Liu et al. LIU *et al.* (2024a) propõem ao enfatizar a importância de datasets variados e preferencialmente alinhados na melhoria do ensino personalizado com modelos de linguagem. Além disso, o uso contínuo do GPT-4o-mini para avaliação e geração garante uma abordagem robusta e escalável para o alinhamento dos modelos, conforme argumentado por Gao et al. GAO *et al.* (2023), que examinaram especificamente a eficácia da avaliação automática em contextos educacionais, e Renze e Guven RENZE e GUVEN (2024), que investigaram o impacto da temperatura e outros parâmetros técnicos no desempenho dos modelos em tarefas complexas.

Ao combinar seleção criteriosa, aprimoramento automático e geração estratégica de pares para DPO, este estudo avança significativamente em direção a métodos eficientes e escaláveis de treinamento e ajuste fino de LLMs compactos, alinhando-se com o estado da arte recente KUMAR e LAN (2024b). Essa metodologia fornece uma base sólida para uma análise detalhada das estratégias de aprendizado por preferência discutidas nas próximas seções.

Após o treinamento inicial dos modelos menores com dados gerados pelo GPT-4o-mini OPENAI (2023), consolidou-se um conjunto robusto de interações simuladas. Uma análise minuciosa das respostas evidenciou que apenas uma fração dos exemplos apresentava alto alinhamento pedagógico e qualidade socrática, refletido nas notas atribuídas pelo GPT-4o.

Com base nessa avaliação, foi adotada uma estratégia de qualificação progressiva do dataset: apenas as interações avaliadas com nota igual ou superior a 4 foram selecionadas para as etapas subsequentes. Tal decisão está em consonância com achados da literatura LIU *et al.* (2024a); RAFAILOV *et al.* (2023a), que enfatizam o impacto de datasets de alta qualidade sobre o desempenho de técnicas como DPO.

O novo conjunto qualificado serviu de base para dois eixos experimentais:

- Um novo ajuste fino supervisionado (SFT), restringindo-se a exemplos de excelência para consolidar o padrão socrático ideal nos modelos;
- E a geração de pares preferenciais para o treinamento com Direct Preference Optimization (DPO), possibilitando a comparação direta entre SFT e DPO em diferentes arquiteturas.

Esse processo incremental de qualificação, com uso intensivo de avaliações au-

tomáticas e geração estratégica de exemplos, visa garantir que tanto SFT quanto DPO sejam conduzidos sob condições de máxima qualidade pedagógica e alinhamento socrático. A abordagem reflete o rigor experimental e as melhores práticas observadas em pesquisas recentes sobre LLMs educacionais.

#### 4.1.6 Comparação Geral dos Modelos Avaliados

Com os resultados obtidos nos experimentos descritos nas seções anteriores, torna-se possível realizar uma comparação abrangente do desempenho dos diferentes modelos avaliados após a aplicação do Fine-Tuning supervisionado (SFT) com LoRA e da técnica Direct Preference Optimization (DPO). A análise quantitativa dessas estratégias é apresentada de forma consolidada na Tabela 4.7, onde estão descritas métricas detalhadas que refletem claramente a eficácia dos modelos em gerar respostas socráticas alinhadas com o método pedagógico proposto neste trabalho.

A tabela apresenta as métricas médias, medianas, bem como as proporções de respostas consideradas de alta qualidade pedagógica, ou seja, aquelas que receberam notas iguais ou superiores a 4 na escala estabelecida previamente. Também são destacadas as proporções específicas de respostas avaliadas com a nota máxima (5), representando excelência no alinhamento socrático.

Tabela 4.7: Comparação geral dos modelos avaliados após aplicação do SFT e DPO

Modelo	Avaliações	Média	Mediana	Notas $\geq 4$	Notas 5
Qwen3-0.6B	828	4,26	5,0	78,4%	51,6%
Qwen3-0.6B-DPO	1202	1,51	1,0	0,67%	0,2%
Qwen3-1.7B	905	3,91	4,0	64,0%	41,1%
Qwen3-1.7B-DPO	1345	2,50	2,0	27,8%	14,6%
LLaMA 3.2-3B	768	4,19	4,0	77,9%	48,7%
LLaMA 3.2-3B-DPO	1008	4,65	5,0	90,6%	79,3%
LLaMA 3.2-1B	962	3,48	4,0	51,8%	27,5%
LLaMA 3.2-1B-DPO	1063	3,75	4,0	60,9%	42,1%

Tabela 4.8: Distribuição percentual das notas atribuídas pelo GPT-4o-mini para cada modelo após SFT (avaliação automática)

Modelo	Nota 0	Nota 1	Nota 2	Nota 3	Nota 4	Nota 5
Qwen 0.6B	0,4%	0,6%	2,2%	18,5%	26,8%	51,6%
Qwen 1.7B	2,2%	1,1%	5,3%	27,4%	23,2%	41,1%
LLaMA 3.2 1B	4,9%	3,3%	9,7%	30,2%	24,3%	27,5%
LLaMA 3B	0,9%	0,7%	3,3%	17,2%	29,3%	48,7%
LLaMA 3.2-1B-DPO	1,0%	6,6%	11,8%	19,8%	18,7%	42,1%
LLaMA 3.2-3B-DPO	0,9%	0,7%	3,3%	17,2%	29,3%	48,7%

Tabela 4.8 detalha a distribuição percentual das notas atribuídas pelo GPT-4o-mini às respostas geradas por cada modelo avaliado após os processos de SFT e DPO. Essa análise permite visualizar de maneira mais granular não apenas a média de desempenho, mas também a consistência, robustez e a incidência de respostas de alta e baixa qualidade para cada configuração experimental.

Observa-se que o modelo **LLaMA 3.2-3B-DPO** apresenta a distribuição mais concentrada nas notas superiores, com 48,7% das respostas avaliadas como nota máxima (5) e 29,3% com nota 4, resultando em mais de 78% das respostas classificadas como excelentes. Paralelamente, o percentual de respostas consideradas insatisfatórias (notas 0, 1 ou 2) é inferior a 5%, indicando um comportamento altamente alinhado com o padrão socrático esperado e um baixo risco de respostas inadequadas.

Por sua vez, o modelo **Qwen3-0.6B** demonstra um perfil mais polarizado no SFT, com mais da metade das respostas na nota máxima (51,6%) e uma quantidade significativa de notas 4 (26,8%), mas também com um salto expressivo de respostas intermediárias (18,5% para nota 3), sugerindo uma maior variabilidade na performance. Já após o DPO, a distribuição do Qwen3-0.6B se desloca dramaticamente para as notas baixas, evidenciando a deterioração já comentada: mais de 80% das respostas foram classificadas entre 0 e 2, praticamente anulando a produção de respostas excelentes.

Modelos intermediários, como o **LLaMA 3.2-1B-DPO**, demonstram uma melhora na distribuição percentual de notas 5 (42,1%) em relação à sua versão SFT (27,5%), embora ainda não alcancem a concentração observada nos modelos de maior capacidade. Nota-se que as proporções de respostas medianas (notas 3 e 4) também permanecem relevantes nesses modelos, indicando um equilíbrio entre excelência e solidez.

Esses dados, quando analisados em conjunto, reforçam que a arquitetura do modelo, a escala de parâmetros e o método de ajuste fino exercem impacto decisivo sobre a distribuição das avaliações. Modelos de maior porte ajustados com DPO apresentam não apenas médias superiores, mas também uma maior densidade de respostas excelentes e um controle mais efetivo sobre respostas insatisfatórias, evidenciando sua maior adequação para tarefas que demandam alto rigor pedagógico e estabilidade de desempenho.

A análise cuidadosa desses resultados revela alguns pontos essenciais para compreensão da eficácia dos métodos utilizados. Primeiramente, observa-se que o modelo **LLaMA 3.2-3B após treinamento com DPO** destacou-se significativamente dos demais, apresentando o desempenho mais consistente e expressivo, com 79,3% de suas respostas consideradas excelentes (nota 5), e uma média geral superior a todos os outros modelos (4,65).

Por outro lado, verifica-se uma deterioração acentuada no desempenho dos modelos **Qwen3-0.6B** e **Qwen3-1.7B** após aplicação do DPO, fato que não foi antecipado inicialmente e merece destaque. Uma hipótese para esse comportamento inesperado pode estar relacionada à ausência de treinamento supervisionado prévio no formato Alpaca antes da aplicação do DPO, causando desalinhamentos no processo de otimização direta de preferências nesses modelos específicos.

A comparação também evidencia que, embora o **LLaMA 3.2-1B-DPO** tenha apresentado uma melhoria incremental sobre sua versão original, os ganhos não foram tão expressivos quanto os observados para o modelo de 3 bilhões de parâmetros. Tal resultado corrobora a expectativa teórica de que modelos com maior capacidade representacional tendem a se beneficiar mais substancialmente de ajustes finos e técnicas avançadas de alinhamento, como o DPO.

Esses resultados reforçam a importância de considerar tanto a arquitetura quanto as características específicas do pré-treinamento dos modelos ao aplicar técnicas avançadas de ajuste fino. A expressiva variação observada no desempenho dos diferentes modelos indica que abordagens genéricas podem não ser igualmente eficazes em todas as circunstâncias, sugerindo que ajustes específicos e cuidadosos são essenciais para maximizar a eficácia de cada técnica.

Em suma, os resultados obtidos neste estudo apontam claramente para a eficácia diferenciada do DPO dependendo da arquitetura do modelo empregado, destacando o **LLaMA 3.2-3B-DPO** como o modelo mais adequado para aplicações práticas do método socrático em contextos educacionais. Adicionalmente, essa comparação detalhada fornece subsídios importantes para futuras pesquisas e aplicações práticas que busquem a otimização eficiente de modelos compactos de linguagem com abordagens avançadas de treinamento por preferência direta.

#### 4.1.7 Avaliação Comparativa dos Modelos após SFT seguido de DPO

Visando aprofundar a análise dos métodos de alinhamento, foi conduzida uma nova rodada experimental aplicando-se o ajuste supervisionado (SFT) seguido do Direct Preference Optimization (DPO) a todos os modelos avaliados neste estudo, **com exceção do Qwen**. Esta exclusão se deve ao desempenho insatisfatório do Qwen nas etapas anteriores, o que indicou baixa relevância para as comparações nesta nova rodada experimental. Diferentemente dos experimentos anteriores, nos quais o DPO foi aplicado diretamente aos modelos base (sem SFT prévio), esta abordagem sequencial tem por objetivo investigar o impacto combinado das duas técnicas na geração de respostas socráticas de alta qualidade.

Nesta etapa, cada modelo foi submetido ao fine-tuning supervisionado (SFT)

seguido do treinamento com DPO (*DPOTrainer*), conduzidos conforme o mesmo procedimento descrito nas seções anteriores. A avaliação automática das respostas continuou sendo realizada pelo GPT-4o-mini, garantindo a comparabilidade entre os diferentes pipelines.

As métricas de desempenho consolidadas para os modelos da família LLaMA 3.2, após a aplicação do pipeline SFT+DPO, são apresentadas na Tabela 4.9

Tabela 4.9: Métricas dos modelos LLaMA 3.2 após SFT seguido de DPO

Modelo	Média	Mediana	% Notas $\geq 4$	% Notas 5	% Notas 0
LLaMA 3.2-1B SFT+DPO	3,82	4,0	64,31	37,99	1,10
LLaMA 3.2-3B SFT+DPO	4,79	5,0	95,33	85,77	0,30

Os resultados indicam um desempenho notavelmente superior para o modelo LLaMA 3.2-3B SFT+DPO. Este alcançou uma média de 4,79 e uma mediana de 5,0, o que significa que pelo menos metade de suas respostas atingiu a pontuação máxima na escala socrática. Além disso, 95,33% de suas respostas foram consideradas de alta qualidade (nota  $\geq 4$ ), com a vasta maioria (85,77%) sendo de qualidade excelente (nota 5). A robustez do modelo também é evidenciada pela baixa incidência de respostas inadequadas, com apenas 0,30% das avaliações recebendo nota 0. O modelo LLaMA 3.2-1B SFT+DPO também apresentou um desempenho sólido, com média de 3,82 e 64,31% de suas respostas classificadas com nota igual ou superior a 4. A distribuição absoluta das notas, detalhada na Tabela 4.10, reforça esses achados.

Tabela 4.10: Distribuição absoluta das notas atribuídas pelo GPT-4o-mini aos modelos LLaMA 3.2 após SFT seguido de DPO

Nota	LLaMA 3.2-1B SFT+DPO	LLaMA 3.2-3B SFT+DPO
0	12	3
1	38	7
2	114	5
3	224	31
4	286	94
5	413	844

Observa-se que, para o modelo LLaMA 3.2-3B, a grande maioria das respostas se concentrou nas notas 4 e 5, com um total de 938 respostas nessas categorias, em contraste com apenas 15 respostas nas categorias 0, 1 e 2 somadas. A Figura 4.3 ilustra essa tendência de forma visual, onde a curva de distribuição percentual do modelo LLaMA 3.2-3B SFT+DPO demonstra um acentuado crescimento em direção às notas mais altas, superando todos os outros pipelines avaliados



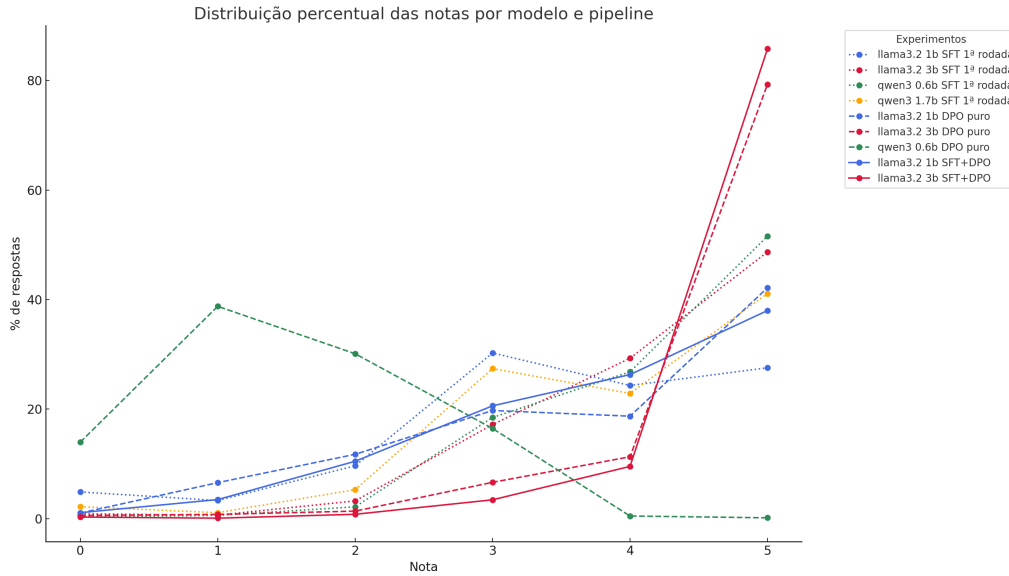


Figura 4.3: Distribuição percentual das notas por modelo e pipeline.

Em síntese, os dados demonstram que o pipeline de treinamento sequencial SFT+DPO foi altamente eficaz, especialmente para o modelo de maior porte (3B), gerando respostas com alto grau de alinhamento ao método socrático e com notável consistência. A análise aprofundada e as implicações pedagógicas destes resultados serão discutidas a seguir.

## 4.2 Validação da Generalização do Modelo em Domínios Inéditos

A capacidade de um modelo de linguagem de generalizar seu aprendizado é um aspecto crucial para validar sua real eficácia, especialmente em tarefas educacionais. Embora os modelos frequentemente apresentem bons resultados em tarefas semelhantes às utilizadas no treinamento (in-domain), é fundamental avaliar sua robustez em contextos inéditos (out-of-domain - OOD), ou seja, em problemas que não fizeram parte do conjunto de dados de ajuste fino CHEN *et al.* (2024).

Estudos recentes reforçam a importância de testar explicitamente a generalização dos LLMs ajustados CHEN *et al.* (2024); WU *et al.* (2025). Wu *et al.* (2025) WU *et al.* (2025) sugerem que o fine-tuning com dados de alta qualidade gerados por modelos de IA, uma abordagem semelhante à empregada nesta dissertação, pode levar a uma maior robustez e a uma menor degradação do desempenho quando o modelo é exposto a tarefas fora de seu domínio original de treinamento. Isso ressalta a importância de verificar empiricamente se tal robustez se manifesta.

Com base nessas premissas, esta seção detalha um experimento complementar de validação. O objetivo é avaliar se o modelo com melhor desempenho (LLaMA 3.2-3B

com pipeline SFT+DPO) consegue aplicar a competência socrática aprendida a um conjunto de problemas de programação de domínios não vistos durante o fine-tuning.

Para garantir a comparabilidade e o rigor metodológico, o protocolo de avaliação foi mantido idêntico aos experimentos anteriores: as interações geradas para os novos problemas foram submetidas ao mesmo avaliador automatizado (GPT-4o-mini) e pontuadas de acordo com a rubrica de qualidade socrática definida na Tabela 4.1. As métricas de análise, como a média das notas e o percentual de respostas de alta qualidade (nota  $\geq 4$ ), também permanecem as mesmas, permitindo uma comparação direta do desempenho do modelo em cenários in-domain versus out-of-domain.

### 4.2.1 Validação Externa dos Modelos

Após a obtenção dos resultados apresentados anteriormente, realizou-se uma etapa adicional de validação externa com o objetivo de avaliar o desempenho dos modelos em questões nunca vistas anteriormente durante o processo de treinamento. Para isso, criou-se um conjunto específico composto por cinco questões inéditas de programação em Python. Este conjunto buscou verificar se os modelos efetivamente internalizaram o método socrático, mantendo uma abordagem pedagógica consistente em contextos não contemplados no fine-tuning.

A validação externa teve como premissa testar a generalização e a capacidade dos modelos em manter o estilo socrático diante de novas situações. A seleção das cinco questões abrangeu diversos tópicos e níveis de complexidade, proporcionando uma análise mais robusta da eficácia pedagógica dos modelos.

As respostas geradas pelos modelos foram avaliadas com auxílio do GPT-4o-mini, adotando-se os mesmos critérios utilizados nas etapas anteriores: clareza, relevância pedagógica, aderência ao método socrático e qualidade geral das interações. As notas atribuídas encontram-se apresentadas na Tabela 4.11.

Tabela 4.11: Resultados da validação externa das respostas geradas pelo modelo LLaMA 3.2-3B SFT-DPO

Nota atribuída (GPT-4o-mini)	Número de respostas	Percentual (%)
0	28	7,7
1	1	0,3
2	3	0,8
3	13	3,6
4	37	10,2
5	281	77,4
<b>Total</b>	<b>363</b>	<b>100,0</b>

Os resultados obtidos demonstraram que o modelo LLaMA 3.2-3B SFT-DPO conseguiu manter um nível significativamente elevado na aplicação do método socrático diante das questões inéditas, com aproximadamente 87,6% das respostas avaliadas como de alta qualidade (notas 4 e 5). Destaca-se especialmente a proporção de respostas consideradas excelentes (nota 5), que alcançou expressivos 77,4%.

Por outro lado, embora minoritária, a ocorrência de respostas inadequadas ou insuficientes (notas 0 a 2), que somaram aproximadamente 8,8%, sugere espaço para futuros aprimoramentos. Esse resultado indica que, apesar do excelente desempenho geral, ainda há necessidade de uma maior diversidade e profundidade das situações incluídas nos dados de treinamento para garantir uma robusta generalização do método socrático.

Esta etapa adicional de validação externa reforça a importância de incorporar avaliações periódicas semelhantes em futuros ciclos de treinamento, visando conti-

nuamente aprimorar a qualidade, consistência pedagógica e aplicabilidade prática dos modelos desenvolvidos.

### 4.3 Plano de Amostragem e Validação Humana

A última etapa do experimento visa comparar as avaliações automáticas atribuídas pelo GPT-4o-mini à escala de socratismo (0–5) com as avaliações de três avaliadores humanos independentes, utilizando o mesmo conjunto de respostas geradas pelo modelo LLaMA 3.2-1B SFT+DPO. Essa análise permitirá quantificar o grau de concordância entre humanos e modelo, identificando eventuais divergências sistemáticas.

Para assegurar representatividade de todas as classes da escala, especialmente considerando que a distribuição original é desbalanceada, adotou-se **amostragem estratificada proporcional com alocação mínima por estrato**, conforme práticas recentes em avaliação de modelos de linguagem em conferências de alto impacto como FISCH *et al.* (2024); ZHANG *et al.* (2024a).

#### Fundamentos da amostragem estratificada

A amostragem estratificada consiste em dividir a população em grupos homogêneos (estratos) e selecionar amostras de cada grupo. No presente caso, cada estrato corresponde a uma classe da escala de notas (0 a 5). Essa técnica é indicada quando:

1. A distribuição das classes é desigual;
2. É necessário avaliar métricas específicas por classe;
3. O orçamento de avaliações humanas é limitado, exigindo maximizar a eficiência estatística.

Estudos recentes demonstram que a estratificação reduz a variância das estimativas e evita a sub-representação de classes raras, problema recorrente em avaliações de LLMs FISCH *et al.* (2024).

#### Cálculo do tamanho mínimo da amostra

Para determinar o tamanho mínimo da amostra total, aplicou-se a fórmula clássica para amostras aleatórias simples, seguida da correção para população finita COCHRAN (1977).

Primeiro, calcula-se o tamanho para uma população infinita:

$$n_0 = \frac{Z^2 \cdot p \cdot (1 - p)}{E^2} \quad (4.1)$$

onde:

- $Z$  é o valor da normal padrão para o nível de confiança ( $95\% \Rightarrow Z = 1,96$ );
- $p$  é a proporção estimada (assumimos  $p = 0,5$  para máxima variabilidade);
- $E$  é o erro máximo tolerado ( $E = 0,05$ ).

Assim:

$$n_0 = \frac{1,96^2 \cdot 0,5 \cdot 0,5}{0,05^2} \approx 384,16$$

Aplicando a correção para população finita ( $N = 1087$ ):

$$n = \frac{n_0}{1 + \frac{n_0 - 1}{N}} \quad (4.2)$$

$$n \approx \frac{384,16}{1 + \frac{383,16}{1087}} \approx 269$$

Portanto, o tamanho mínimo global da amostra para este estudo é de  $n_{\text{total}} \approx 269$  exemplos.

### Distribuição da amostra por estrato

A alocação proporcional simples definiria  $n_k = N_k \cdot \frac{n_{\text{total}}}{N}$ . Contudo, para evitar classes com amostras muito pequenas, adotou-se um **piso mínimo**  $n_{\text{min}} = 20$ , exceto para estratos cuja população  $N_k$  seja inferior a esse valor (neste caso, usa-se  $n_k = N_k$ ). Na prática, a amostra utilizada para a validação humana contém  $n_{\text{total}} = 295$  itens (derivada da coluna *Avaliador GPT*), e a classe 0 precisou ser complementada com **8 exemplos adicionais** para atingir o piso ( $12 \rightarrow 20$ ).

Tabela 4.12: População original (1087) vs. amostra usada na validação (295) por classe de nota.

Classe ( $k$ )	População (N=1087)		Amostra (n=295)		Peso $w_k$
	$N_k$	Proporção (%)	$n_k$	Proporção (%)	
0	12	1.10%	20	6.78%	0.163
1	38	3.5%	20	6.78%	0.516
2	114	10.49%	31	10.51%	0.998
3	224	20.61%	50	16.95%	1.216
4	286	26.31%	67	22.71%	1.159
5	413	38%	107	36.27%	1.048
<b>Total</b>	<b>1087</b>	<b>100%</b>	<b>295</b>	<b>100%</b>	—

Os pesos de pós-estratificação  $w_k$  são definidos por

$$w_k = \frac{(N_k/N)}{(n_k/n_{\text{total}})},$$

e estão reportados na Tabela 4.12 para transparência e eventual reponderação de métricas agregadas.

Notamos que a classe 0 estava sub-representada no conjunto rotulado pelo GPT-4o-mini (12 exemplos no universo de 1087), de modo que adicionamos 8 exemplos dessa classe na amostra final para atingir o piso mínimo de 20 por estrato, preservando a viabilidade de estimativas por classe.

### Definição formal

Seja  $y_i^{(H)}$  a nota consensual atribuída pelos avaliadores humanos para o exemplo  $i$ , e  $y_i^{(G)}$  a nota atribuída pelo GPT-4o-mini, o MAE simples é definido como:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N \left| y_i^{(H)} - y_i^{(G)} \right| \quad (4.3)$$

onde  $N$  é o número total de exemplos avaliados.

### 4.3.1 Métrica de Avaliação: Erro Absoluto Médio e Acurácia Ponderada

Para quantificar a diferença entre as notas atribuídas pelo modelo GPT-4o-mini e o consenso dos avaliadores humanos, optou-se por utilizar o *Erro Absoluto Médio* (*Mean Absolute Error* – MAE) como métrica inicial. O MAE mede, em média, o quanto a previsão de um sistema se afasta do valor real observado, sendo intuitivo e amplamente utilizado em tarefas de avaliação com escalas ordinais (FISCH *et al.*, 2024; ZHENG *et al.*, 2023).

#### Definição Formal e Ponderação por Representatividade

Seja  $y_i^{(H)}$  a nota consensual atribuída pelos avaliadores humanos para o exemplo  $i$ , e  $y_i^{(G)}$  a nota atribuída pelo GPT-4o-mini, o MAE simples é definido como:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N \left| y_i^{(H)} - y_i^{(G)} \right| \quad (4.4)$$

onde  $N$  é o número total de exemplos avaliados na amostra.

Contudo, uma análise da distribuição das notas na amostra de avaliação ( $N = 295$ ) revelou um desbalanceamento em relação à população completa de dados ( $N =$

1087). Certas classes de notas estavam super-representadas na amostra, enquanto outras estavam sub-representadas. Para corrigir esse viés amostral e obter uma estimativa de erro mais fidedigna à performance do modelo na população geral, foi introduzido o **Erro Absoluto Médio Ponderado** (*Weighted Mean Absolute Error* – WMAE).

O WMAE ajusta a contribuição de cada erro individual com base na representatividade de sua classe. O peso  $w_k$  para cada classe  $k$  (de 0 a 5) é calculado como a razão entre a proporção da classe na população ( $P_k$ ) e sua proporção na amostra ( $p_k$ ):  $w_k = P_k/p_k$ . A fórmula do WMAE é então:

$$\text{WMAE} = \frac{\sum_{i=1}^N w_k |y_i^{(H)} - y_i^{(G)}|}{\sum_{i=1}^N w_k} \quad (4.5)$$

onde  $w_k$  é o peso correspondente à classe da nota de referência  $y_i^{(H)}$ .

### Avaliação por Classificação Binária

Para além da análise na escala ordinal, o problema foi abstraído para uma tarefa de classificação binária, simplificando a interpretação do desempenho do modelo em distinguir interações socráticas de não socráticas. As notas foram agrupadas em duas categorias:

- **Não Socrático:** Notas de 0 a 2.
- **Socrático:** Notas de 3 a 5.

Neste cenário, métricas de classificação se tornam mais apropriadas. A performance foi avaliada por meio da **Matriz de Confusão**, que detalha os acertos e erros do modelo, incluindo Verdadeiros Positivos (TP), Verdadeiros Negativos (TN), Falsos Positivos (FP) e Falsos Negativos (FN).

A partir dela, calculou-se a **Acurácia Ponderada** (*Weighted Accuracy*), que, de forma análoga ao WMAE, ajusta a contagem de acertos com base nos pesos de representatividade  $w_k$  das classes binárias. A Acurácia Ponderada oferece a medida mais robusta do desempenho do modelo, pois reflete sua capacidade de classificação corrigida pelo viés da amostra.

### Resultados da Avaliação Quantitativa

A aplicação das métricas de avaliação revelou uma performance robusta do modelo GPT-4o-mini, tanto na análise de regressão quanto na de classificação.

Inicialmente, ao analisar a escala ordinal completa de 0 a 5, o Erro Absoluto Médio (MAE) não ponderado foi de 1.034. No entanto, ao aplicar os pesos para corrigir o viés da amostragem, o Erro Absoluto Médio Ponderado (WMAE) subiu para

1.053. Este aumento, embora sutil, indica que os erros do modelo foram ligeiramente mais expressivos nas classes sub-representadas na amostra.

Na avaliação da tarefa de classificação binária (Socrático vs. Não Socrático), o modelo alcançou uma acurácia global de 79.66%. A análise detalhada dos tipos de erro é apresentada na Matriz de Confusão (Tabela 4.13).

Tabela 4.13: Matriz de Confusão para a classificação binária.

		Previsto pelo GPT	
		Não Socrático	Socrático
Real	Não Socrático	47	36
	Socrático	24	188

A matriz evidencia que o erro mais comum do modelo foi o de Falso Positivo (36 ocorrências), classificando uma resposta como socrática quando, na verdade, não era. As métricas de precisão, revocação (*recall*) e F1-Score, detalhadas na Tabela 4.14, complementam a análise.

Tabela 4.14: Métricas de desempenho por classe.

Classe	Precisão	Revocação	F1-Score
Não Socrático	0.66	0.57	0.61
Socrático	0.84	0.89	0.86

Finalmente, ao ponderar a acurácia pela representatividade das classes na população, obtém-se a métrica mais fidedigna do desempenho do sistema. A **Acurácia Ponderada** alcançou o valor de **83.84%**, um aumento de mais de quatro pontos percentuais em relação à acurácia não ponderada. Este resultado final reforça a conclusão de que a performance do modelo é notavelmente eficaz, especialmente ao se ajustar sua avaliação à distribuição real dos dados.



# Capítulo 5

## Conclusões e Trabalhos Futuros

### 5.1 Síntese dos Resultados

Este estudo teve como objetivo central avaliar a eficácia de modelos compactos de linguagem (LLMs) submetidos a técnicas avançadas de ajuste fino, especificamente o Fine-Tuning supervisionado via adaptadores LoRA e a Direct Preference Optimization (DPO), para gerar interações socráticas aplicadas ao ensino de programação. A metodologia experimental rigorosa adotada permitiu uma análise detalhada dos modelos selecionados (LLaMA 3.2 1B e 3B, Qwen 0.6B e 1.7B), utilizando o GPT-4o como referência qualitativa e avaliador automático.

Inicialmente, os experimentos destacaram a importância crítica da qualidade dos dados sintéticos utilizados para treinamento. A investigação sobre o hiperparâmetro de temperatura mostrou que valores intermediários, especialmente entre 0.2 e 0.4, foram os mais eficazes para gerar interações socráticas consistentes e pedagogicamente valiosas. Este ajuste resultou em um conjunto robusto de 847 interações qualificadas, que serviram como base para as etapas subsequentes de ajuste fino.

Na etapa seguinte, a avaliação comparativa dos métodos de ajuste fino revelou um cenário complexo e altamente dependente da arquitetura específica dos modelos. O ajuste fino supervisionado (SFT) com LoRA apresentou-se como uma base inicial sólida para todos os modelos, garantindo uma estabilidade inicial significativa. Entretanto, a aplicação direta da técnica DPO sem treinamento supervisionado prévio evidenciou resultados divergentes, particularmente prejudiciais para a família Qwen, que apresentou deterioração substancial. Este resultado sinaliza que a eficácia plena da DPO requer um treinamento intermediário adequado, sugerindo que adaptações prévias específicas são críticas para o sucesso deste método em diferentes contextos e arquiteturas.

O destaque indiscutível em todas as etapas experimentais foi o modelo LLaMA 3.2-3B, especialmente após o treinamento sequencial com SFT seguido de DPO.

Este modelo obteve uma média de avaliação notável de 4,79 e uma mediana perfeita de 5,0, com 95,33% das respostas sendo classificadas como de alta qualidade (notas  $\geq 4$ ), das quais 85,77% receberam a nota máxima (5). Estes resultados refletem claramente um alinhamento pedagógico superior com a abordagem socrática pretendida. Além disso, a robustez desse modelo foi confirmada em uma validação externa com questões inéditas de programação em Python, mantendo um desempenho consistente e de alta qualidade, com 87,6% das respostas classificadas como excelentes. Essa validação reforça a capacidade de generalização do modelo e sua efetividade prática.

A análise crítica dos resultados experimentais sugere que modelos maiores, como o LLaMA 3.2-3B, beneficiam-se significativamente mais das técnicas avançadas de ajuste fino (SFT seguido de DPO), confirmando teoricamente sua superior capacidade representacional em comparação a modelos menores. Ademais, fica clara a importância da qualidade inicial dos dados para ajustes supervisionados, ressaltando que uma curadoria rigorosa e progressiva do dataset é essencial para o sucesso subsequente das técnicas de otimização. Finalmente, os experimentos também destacam o impacto crucial da arquitetura e do pré-treinamento específicos sobre a eficácia dos métodos empregados, enfatizando a necessidade de estratégias adaptativas individualizadas para diferentes contextos.

Concluindo, o framework proposto mostrou-se eficaz e economicamente viável para desenvolver agentes educacionais de IA escaláveis e de alta performance pedagógica, comparável inclusive a modelos significativamente maiores, como o GPT-4o. O modelo LLaMA 3.2-3B destacou-se como a solução mais robusta e promissora, oferecendo uma abordagem clara e eficiente para enfrentar os desafios estruturais no ensino personalizado de programação.

## 5.2 Originalidade e Contribuições

Esta dissertação apresenta avanços significativos em relação ao estado da arte no uso de grandes modelos de linguagem (LLMs) para ensino socrático em programação. Embora estudos recentes explorem o método socrático com LLMs e a aplicação de técnicas como LoRA e Direct Preference Optimization (DPO) para ajuste fino de modelos AL-HOSSAMI *et al.* (2023b); HU *et al.* (2022); KUMAR e LAN (2024a,b); LIU *et al.* (2024a); RAFAILOV *et al.* (2023a); RASCHKA (2023), este trabalho distingue-se por propor, implementar e analisar criticamente um fluxo experimental completo, padronizado e automatizável, adequado a modelos compactos e de baixo custo computacional.

As principais contribuições e elementos de originalidade deste estudo são:

- **Proposição de um pipeline integrado e automatizado** para geração, avaliação, seleção e ajuste fino de modelos LLMs compactos, utilizando tanto LoRA quanto DPO, com validação baseada em notas atribuídas automaticamente por um avaliador LLM (GPT-4o) KUMAR e LAN (2024b); LIU *et al.* (2024a); OPENAI (2023).
- **Foco em eficiência paramétrica e democratização do acesso à IA educacional**, ao priorizar modelos de até 3B parâmetros e métodos de treinamento economicamente viáveis, permitindo uso em hardware restrito e replicação por pesquisadores ou docentes sem infraestrutura de alto desempenho HU *et al.* (2022); KAPLAN *et al.* (2020); RASCHKA (2023).
- **Comparação sistemática e robusta** entre caminhos de ajuste supervisionado tradicional (SFT + LoRA) e otimização por preferência (DPO), evidenciando os trade-offs e benefícios de cada abordagem em contextos de ensino socrático KUMAR e LAN (2024a); RAFAILOV *et al.* (2023a,b).
- **Metodologia escalável de avaliação automática**, eliminando a necessidade de grandes equipes de anotadores humanos e acelerando o ciclo de experimentação em educação com IA GOEL *et al.* (2023); KUMAR e LAN (2024b); LIU *et al.* (2024a); TÖRNBERG (2023); WANG *et al.* (2021).
- **Produção de guidelines práticos e replicáveis** para o ajuste fino de LLMs em tarefas educativas, permitindo a adoção do framework proposto em contextos variados, inclusive no cenário brasileiro.

Com isso, a presente dissertação contribui tanto para a pesquisa acadêmica quanto para a prática docente, oferecendo um caminho objetivo e mensurável para o desenvolvimento de agentes educacionais inteligentes, fundamentado em bases empíricas e alinhado às tendências internacionais em IA e Educação.

### 5.3 Limitações

Apesar dos resultados positivos, este estudo apresenta limitações importantes que devem ser reconhecidas. A principal dificuldade esteve na infraestrutura computacional: a ausência de uma GPU dedicada obrigou a realização dos experimentos em um notebook Apple M2, o que tornou o tempo de execução dos treinamentos bastante elevado, mesmo utilizando modelos considerados compactos. A solução encontrada foi recorrer ao Google Colab, que, embora útil, impôs restrições de tempo de uso e acesso, limitando o número de experimentos e a complexidade dos testes conduzidos. Mesmo utilizando LoRA para ajuste fino, técnica que permite treinar

apenas uma fração dos parâmetros do modelo, todo o processo ficou condicionado à disponibilidade de recursos gratuitos ou de baixo custo.

Em razão dessas limitações de infraestrutura, optou-se por testar exclusivamente o rank 16 no LoRA, sem explorar outras configurações de ranks que poderiam impactar significativamente o desempenho e o custo computacional dos modelos. Da mesma forma, não foi possível experimentar técnicas de quantização como o QLoRA, que poderiam ter reduzido ainda mais o uso de memória e acelerado o treinamento, ampliando o escopo das análises e dos experimentos. Portanto, os resultados reportados referem-se apenas ao cenário específico avaliado, o que limita a generalização para outros contextos de ajuste fino e quantização.

Outra limitação relevante refere-se à etapa de avaliação das respostas: utilizou-se o modelo GPT-4o como juiz automático, reconhecido na literatura por sua qualidade, mas cujo uso envolveu custos financeiros expressivos ao longo da pesquisa. Além disso, não foi realizada uma validação cruzada com avaliadores humanos, o que poderia ter aprofundado a análise qualitativa e garantido maior robustez metodológica aos resultados. Embora referências recentes sugiram que o GPT-4 pode ser comparável a avaliadores humanos em tarefas de avaliação pedagógica, não foi possível, neste estudo, comprovar essa equivalência empiricamente.

O escopo de modelos testados também foi limitado: por conta dos desafios operacionais e da necessidade de otimizar recursos, a avaliação experimental restringiu-se às famílias Qwen e LLaMA, deixando de fora outros modelos abertos promissores que poderiam apresentar comportamentos distintos frente às técnicas adotadas.

Finalmente, destaca-se a ausência de avaliação longitudinal com estudantes reais, o que impede afirmar, com base em evidências práticas, o impacto do framework proposto sobre a aprendizagem ao longo do tempo. Futuras pesquisas que incluam usuários reais e experimentos em ambientes educacionais podem contribuir para validar, de modo mais amplo, a eficácia da abordagem.

Em resumo, os desafios de infraestrutura, a limitação na configuração dos experimentos (incluindo o uso exclusivo do rank 16 e a ausência de testes com quantização), o escopo reduzido de modelos e a dependência da validação automática representam pontos a serem considerados na interpretação dos resultados e orientam caminhos para investigações futuras.

## 5.4 Propostas para Trabalhos Futuros

A partir das limitações reconhecidas e dos resultados promissores obtidos, delineiam-se diversas áreas para futuras pesquisas que possam ampliar a relevância, a eficácia e a aplicabilidade do framework proposto. As seguintes propostas visam aprofundar a investigação e aprimorar o uso de LLMs no ensino socrático de

programação:

- **Estudo Longitudinal sobre Eficácia Pedagógica com Usuários Reais:** Realizar uma pesquisa longitudinal com estudantes em cursos de programação para avaliar o impacto do framework no desempenho acadêmico, na retenção de conhecimento e nos níveis de engajamento. Um estudo controlado, comparando grupos com e sem acesso à ferramenta, permitiria validar empiricamente a eficácia do método socrático mediado por IA ao longo do tempo.
- **Análise Sistemática de Hiperparâmetros de Fine-Tuning Eficiente:** Conduzir uma investigação aprofundada sobre o impacto de diferentes configurações de *ranks* do LoRA e a aplicação de técnicas de quantização como o QLoRA. Analisar sistematicamente o *trade-off* entre o custo computacional (tempo de treinamento e uso de memória), o tamanho do modelo e a qualidade pedagógica das respostas, a fim de estabelecer diretrizes otimizadas para cenários com recursos computacionais restritos.
- **Expansão e Avaliação com um Espectro Maior de Modelos Abertos:** Ampliar o estudo comparativo para incluir outros modelos de linguagem de código aberto promissores (e.g., Mistral, Gemma, entre outros). A avaliação sistemática desses modelos com o pipeline SFT+DPO proposto pode revelar alternativas com melhor custo-benefício e identificar quais arquiteturas são mais receptivas a este tipo de alinhamento pedagógico.
- **Desenvolvimento de um Framework de Avaliação Híbrido:** Para mitigar a dependência de um único avaliador automático (GPT-40) e os custos associados, propõe-se o desenvolvimento de um framework de avaliação híbrido. Este incluiria uma validação cruzada com anotadores humanos especialistas em pedagogia e programação, permitindo não apenas aprofundar a análise qualitativa, mas também calibrar e validar a precisão dos avaliadores automáticos.
- **Personalização Adaptativa Baseada em Perfis Cognitivos do Aluno:** Investigar métodos para a identificação automática de perfis cognitivos e estilos de aprendizagem dos estudantes com base em seu histórico de interações. Um sistema capaz de adaptar dinamicamente a profundidade, o ritmo e o estilo do questionamento socrático às necessidades individuais de cada aluno tem o potencial de tornar o processo educacional significativamente mais eficaz e personalizado.
- **Aplicação do Framework a Outras Tarefas de Programação:** Estender a aplicação do método socrático para além da depuração de código (*de-*

*bugging*). Investigar a eficácia do framework em outras tarefas complexas do ciclo de desenvolvimento de software, como o auxílio na refatoração de código (*refactoring*), na otimização de algoritmos ou na concepção inicial da lógica de um programa.

- **Análise de Potenciais Vieses** Realizar uma análise aprofundada sobre possíveis vieses (bias) no comportamento do tutor de IA. Investigar se o modelo favorece determinados estilos de codificação, se pode reforçar concepções errôneas ou se a sua condução socrática limita a criatividade do aluno.
- **Integração de Recursos Multimodais no Diálogo Socrático:** Explorar a integração de elementos multimodais na interação, como a capacidade do sistema de gerar ou analisar diagramas de fluxo, visualizar estruturas de dados ou apresentar trechos de código com *syntax highlighting* dinâmico. Avaliar se a multimodalidade pode aprimorar a compreensão de conceitos abstratos de programação.

A execução dessas propostas pode não apenas fortalecer as contribuições desta dissertação, mas também impulsionar o desenvolvimento de ferramentas de IA educacionais mais robustas, acessíveis e pedagogicamente eficazes.

# Referências Bibliográficas

- AGHAJANYAN, A., GUPTA, A., SHRIVASTAVA, A., et al., 2021, “Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 785–801. Association for Computational Linguistics.
- AL-HOSSAMI, E., BUNESCU, R., SMITH, J., et al., 2023a, “Can Language Models Employ the Socratic Method? Experiments with Code Debugging”, *arXiv preprint arXiv:2310.03210*. Disponível em: <<https://arxiv.org/abs/2310.03210>>. Acessado em: 10 fev. 2024.
- AL-HOSSAMI, E., BUNESCU, R., TEEHAN, R., et al., 2023b, “Socratic Questioning of Novice Debuggers: A Benchmark Dataset and Preliminary Evaluations”. In: *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 709–726, b.
- AL-HOSSAMI, E., BUNESCU, R., TEEHAN, R., et al., 2024, “Socratic Questioning of Novice Debuggers: A Benchmark Dataset and Preliminary Evaluations”. In: *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 709–726.
- ALIZADEH, M., KUBLI, M., SAMEI, Z., et al., 2025, “Open-source LLMs for text annotation: a practical guide for model setting and fine-tuning”, *Journal of Computational Social Science*, v. 8, n. 1, pp. 1–25.
- ANG, J., OTHERS, 2024. “SocratiQ: A Benchmark for Socratic Question Generation”. <http://arxiv.org/html/2409.05511>. Disponível em: <http://arxiv.org/html/2409.05511>.
- BO, X., ZHANG, Z., DAI, Q., et al., 2024, “Reflective Multi-Agent Collaboration based on Large Language Models”. In: *Advances in Neural Information Processing Systems 38 (NeurIPS 2024)*.
- BORGEAUD, S., MENSCH, A., HOFFMANN, J., et al., 2022, “Improving language models by retrieving from trillions of tokens”, *Proceedings of*

*the International Conference on Machine Learning (ICML)*, v. 162, pp. 2206–2240.

BROWN, T. B., MANN, B., RYDER, N., et al., 2020, “Language Models are Few-Shot Learners”, v. 33, pp. 1877–1901. Disponível em: <[https://proceedings.neurips.cc/paper\\_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf)>.

CHANG, K., XU, S., WANG, C., et al., 2024, “Efficient Prompting Methods for Large Language Models: A Survey”, *arXiv preprint arXiv:2404.01077*. Disponível em: <https://arxiv.org/pdf/2404.01077>.

CHEN, Z., OTHERS, 2024, “Unveiling the Generalization Power of Fine-Tuned Large Language Models”. In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics. Disponível em: <<https://aclanthology.org/2024.naacl-long.51/>>.

COCHRAN, W. G., 1977, *Sampling Techniques*. 3 ed. New York, John Wiley & Sons. ISBN: 9780471162407.

DEBNATH, I., HASSAN, M. S., ALGHAMDI, T., et al., 2025, “A Comprehensive Survey of Prompt Engineering Techniques in Large Language Models”, *Faculty Publications*, v. 523. Disponível em: <[https://digitalcommons.odu.edu/ece\\_fac\\_pubs/514/](https://digitalcommons.odu.edu/ece_fac_pubs/514/)>.

DEVLIN, J., CHANG, M.-W., LEE, K., et al., 2019, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of NAACL-HLT*, pp. 4171–4186. Association for Computational Linguistics.

DING, Y., HU, H., ZHOU, J., et al., 2024a, “Boosting Large Language Models with Socratic Method for Conversational Mathematics Teaching”, *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*.

DING, Y., HU, H., ZHOU, J., et al., 2024b, “Boosting Large Language Models with Socratic Method for Conversational Mathematics Teaching”, *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*. Disponível em: <<https://arxiv.org/abs/2407.17349>>.



- DU, Y., CHEN, L., HE, X., 2024a, “Large Language Models are Catalyzing Chemistry Education”, *ChemRxiv preprint*. Disponível em: <<https://chemrxiv.org/engage/66772be25101a2ffa8412ee0>>. Acessado em: 15 fev. 2024.
- DU, Y., LI, S., TORRALBA, A., et al., 2024b, “Improving Factuality and Reasoning in Language Models through Multiagent Debate”. In: *Proceedings of the 41st International Conference on Machine Learning (ICML)*, b. Disponível em: <<https://openreview.net/pdf?id=zj7YuTE4t8>>.
- FISCH, A., MAYNEZ, J., HOFER, R., et al., 2024, “Stratified prediction-powered inference for effective hybrid evaluation of language models”, *Advances in Neural Information Processing Systems*, v. 37, pp. 111489–111514.
- GAO, L., ZHANG, P., HUANG, W., 2024, “AI-Driven Interactive Learning: The Role of Large Language Models in STEM Education”, *ChemRxiv preprint*. Disponível em: <<https://chemrxiv.org/engage/ai-in-stem-education>>. Acessado em: 15 fev. 2024.
- GAO, R., XIAO, P., LIN, T., 2023, “Grading AI: Evaluating the Performance of LLMs in Educational Contexts”, *Journal of Educational Data Science*, v. 11, pp. 78–99.
- GATTI, G., OTHERS, 2023. “Fine Tuning a Large Language Model for Socratic Interactions”. <https://giovannigatti.github.io/socratic-llm/>. Acesso em: 11 mai. 2025.
- GOEL, A., GUETA, A., GILON, O., et al., 2023, “Llms accelerate annotation for medical information extraction”. In: *Machine Learning for Health (ML4H)*, pp. 82–100. PMLR.
- GOOGLE, 2022. “A escassez de profissionais de tecnologia no Brasil e seu impacto no ecossistema de startups”. Disponível em: <<https://blog.google/intl/pt-br/produtos/a-escassez-dos-profissionais-de-tecnologia-no-brasil-e-seu-consequente-imp>>. Acessado em: 10 out. 2023.
- HOULSBY, N., GIURGIU, A., JASTRZEBSKI, S., et al., 2019, “Parameter-efficient transfer learning for NLP”. In: *International Conference on Machine Learning*, pp. 2790–2799. PMLR.
- HOWARD, J., RUDER, S., 2018, “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Associa-*

*tion for Computational Linguistics (ACL 2018)*, pp. 328–339. Association for Computational Linguistics.

HU, E. J., SHEN, Y., WALLIS, P., et al., 2022, “LoRA: Low-Rank Adaptation of Large Language Models”. In: *Proceedings of the Tenth International Conference on Learning Representations (ICLR)*. Disponível em: <<https://openreview.net/forum?id=nZeVKeeFYf9>>. Available at <https://openreview.net/forum?id=nZeVKeeFYf9>.

KAPLAN, J., MCCANDLISH, S., HENIGHAN, T., et al., 2020, “Scaling Laws for Neural Language Models”, *arXiv preprint*. Disponível em: <<https://arxiv.org/abs/2001.08361>>.

KARGUPTA, P., AGARWAL, I., HAKKANI-TUR, D., et al., 2024, “Instruct, Not Assist: LLM-based Multi-Turn Planning and Hierarchical Questioning for Socratic Code Debugging”, *arXiv preprint arXiv:2406.11709*. Disponível em: <<https://arxiv.org/abs/2406.11709>>.

KUMAR, N. A., LAN, A., 2024a, “Improving Socratic Question Generation using Data Augmentation and Preference Optimization”, *arXiv preprint arXiv:2403.00199*. Disponível em: <<https://arxiv.org/abs/2403.00199>>. Acessado em: 10 fev. 2024.

KUMAR, N. A., LAN, A., 2024b, “Improving Socratic Question Generation using Data Augmentation and Preference Optimization”, *arXiv preprint arXiv:2403.00199*. Disponível em: <<https://arxiv.org/abs/2403.00199>>.

LEWIS, P., PEREZ, E., PIKTUS, A., et al., 2020, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”, *Advances in Neural Information Processing Systems (NeurIPS)*, v. 33, pp. 9459–9474.

LI, G., HAMMOUD, H. A. A. K., ITANI, H., et al., 2023, “CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society”. In: *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

LIU, H., WANG, X., ZHANG, Y., 2024a, “SocraticLM: Exploring Socratic Personalized Teaching with Large Language Models”, *arXiv preprint arXiv:2402.01742*. Disponível em: <<https://arxiv.org/abs/2402.01742>>. Acessado em: 15 fev. 2024.

LIU, J., PAN, Z., ZHU, K., et al., 2024b, “SocraticLM: Exploring Socratic Personalized Teaching with Large Language Models”, *arXiv preprint*

- arXiv:2404.09321*. Disponível em: <<https://arxiv.org/abs/2404.09321>>.
- LUO, Y., YANG, Z., MENG, F., et al., 2023, “An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning”, *arXiv preprint arXiv:2308.08747*. Disponível em: <<https://arxiv.org/abs/2308.08747>>.
- MINSKY, M., 1988, *The Society of Mind*. New York, NY, Simon and Schuster. ISBN: 9780671657130.
- OPENAI, 2023, “GPT-4 Technical Report”, *arXiv preprint arXiv:2303.08774*. Disponível em: <<https://arxiv.org/abs/2303.08774>>.
- PADESKY, C., 2020. “Socratic Dialogue Rating Scale & Coding Manual v5.0”. [https://www.padesky.com/wp-content/uploads/2023/07/Socratic-Dialogue-RatingSCALE-MANUAL\\_v5.pdf](https://www.padesky.com/wp-content/uploads/2023/07/Socratic-Dialogue-RatingSCALE-MANUAL_v5.pdf). Acesso em: 11 mai. 2025.
- PAUL, R., ELDER, L., 2006, *The Art of Socratic Questioning*. Tomales, CA, Foundation for Critical Thinking.
- PAUL, R., ELDER, L., 2019, *Critical Thinking: Tools for Taking Charge of Your Professional and Personal Life*. Pearson Education.
- QUINTANA, C., REISER, B. J., DAVIS, E. A., et al., 2004, “A scaffolding design framework for software to support science inquiry”, *Journal of the Learning Sciences*, v. 13, n. 3, pp. 337–386.
- RAFAILOV, R., SHARMA, A., MITCHELL, E., et al., 2023a, “Direct Preference Optimization: Your Language Model is Secretly a Reward Model”, *arXiv preprint arXiv:2305.18290*. Disponível em: <<https://arxiv.org/abs/2305.18290>>.
- RAFAILOV, R., SHARMA, A., MITCHELL, E., et al., 2023b, “Direct Preference Optimization: Your Language Model is Secretly a Reward Model”, *Neural Information Processing Systems (NeurIPS)*. Disponível em: <<https://openreview.net/forum?id=XPZiaotutsD>>.
- RASCHKA, S., 2023. “Parameter-Efficient LLM Finetuning With Low-Rank Adaptation (LoRA)”. <https://sebastianraschka.com/blog/2023/llm-finetuning-lora.html>. Accessed: 2025-04-14.

- RENZE, M., GUVEN, E., 2024, “The Effect of Sampling Temperature on Problem Solving in Large Language Models”, *arXiv preprint arXiv:2402.05201*. Disponível em: <<https://arxiv.org/abs/2402.05201>>.
- SAHOO, R. K., SRINIVAS, A., PANDA, D., et al., 2024, “A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications”, Disponível em: <<https://rotmandigital.ca/wp-content/uploads/2024/09/A-Systematic-Survey-of-Prompt-Engineering-in-Large-Language-Models.pdf>>.
- SEPROSC, 2023. “Falta de professores na área de TI pode desacelerar o processo de formação de novos profissionais”. Disponível em: <<https://www.seprosc.com.br/n.php?ID=423&T=falta-de-professores-na-area-de-ti-pode-desacelerar-o-processo-de-formacao>>. Acessado em: 10 out. 2023.
- TAN, Z., LI, D., WANG, S., et al., 2024, “Large language models for data annotation and synthesis: A survey”, *arXiv preprint arXiv:2402.13446*.
- TÖRNBERG, P., 2023, “Chatgpt-4 outperforms experts and crowd workers in annotating political twitter messages with zero-shot learning”, *arXiv preprint arXiv:2304.06588*.
- VATSAL, DUBEY, R., 2024, “A Survey of Prompt Engineering Methods in Large Language Models”, Disponível em: <[https://www.researchgate.net/publication/382363222\\_A\\_Survey\\_of\\_Prompt\\_Engineering\\_Methods\\_in\\_Large\\_Language\\_Models\\_for\\_Different\\_NLP\\_Tasks](https://www.researchgate.net/publication/382363222_A_Survey_of_Prompt_Engineering_Methods_in_Large_Language_Models_for_Different_NLP_Tasks)>.
- VYGOTSKY, L. S., 1978, *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA, Harvard University Press.
- WANG, S., LIU, Y., XU, Y., et al., 2021, “Want to reduce labeling cost? GPT-3 can help”, *arXiv preprint arXiv:2108.13487*.
- WEI, J., WANG, X., SCHUURMANS, D., et al., 2022a, “Chain of Thought Prompting Elicits Reasoning in Large Language Models”, *arXiv preprint arXiv:2201.11903*. Disponível em: <<https://arxiv.org/abs/2201.11903>>.
- WEI, J., WANG, X., SCHUURMANS, D., et al., 2022b, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”, *Advances in Neural Information Processing Systems (NeurIPS)*. Disponível em: <<https://arxiv.org/abs/2201.11903>>.

- WOOD, D., BRUNER, J. S., ROSS, G., 1976, “The role of tutoring in problem solving”, *Child Psychology & Psychiatry*, v. 17, n. 2, pp. 89–100.
- WU, C., OTHERS, 2025, “Clear Minds Think Alike: What Makes LLM Fine-tuning Robust? A Study of Token Perplexity”, *arXiv preprint arXiv:2501.14315*. Disponível em: <<https://arxiv.org/html/2501.14315v1>>.
- ZHANG, Y., LI, X., KUMAR, S., et al., 2024a, “Statistically Robust Evaluation of Large Language Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*, a.
- ZHANG, Y., SUN, R., CHEN, Y., et al., 2024b, “Chain of Agents: Large Language Models Collaborating on Long-Context Tasks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*, b. Disponível em: <<https://openreview.net/pdf?id=LuCLf4BJsr>>.
- ZHENG, L., CHIANG, W.-L., SHENG, Y., et al., 2023, “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena”, *arXiv preprint arXiv:2306.05685*. Disponível em: <<https://arxiv.org/abs/2306.05685>>.