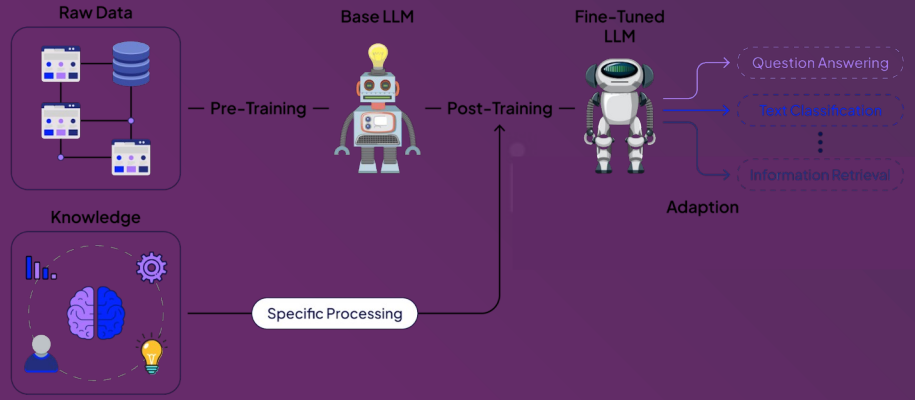
3º.Ciclo de Encontros: 4 x PLN



Treinamento de Grandes Modelos de Linguagem na Prática

Gabriel Lino Garcia - UNESP
Pedro Henrique Paiola - UNESP

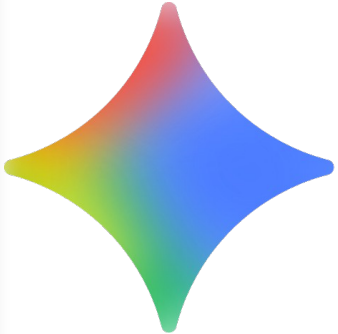
11/05/2026

Sumário

Imagem

O que são LLMs?

Introdução



O que são Large Language Models (LLMs)?

- Grandes Modelos de Linguagem (LLMs) representam uma mudança de paradigma no PLN.
- Impulsionados pela arquitetura Transformer, que permite dar atenção ao contexto.
- LLMs são treinados em textos massivos da internet.
- **Resultado:** geração de texto fluente e coerente em escala.
- **Problema:** para tarefas de precisão ou nicho (direito, medicina, finanças), o modelo generalista é insuficiente.

Duas dimensões de adaptação

1. **Especialização de domínio**
    a. Adaptar o modelo a termos e conceitos de áreas específicas (medicina, direito, etc)
    b. Ganho de maior precisão em tarefas como sumarização de prontuários ou análise de contratos.

2. **Adaptação linguística/cultural**
    c. Maioria dos LLMs é treinada em inglês
    d. Tradução da resposta não resolve problemas culturais
    e. **Solução:** ajuste fino com dados locais

Pipeline Geral

1. **Pré-treinamento -> aprende linguagem**
2. **Continuação de Pré-Treino (CPT) -> aprende domínio**
3. **Fine-tuning -> aprende tarefa**
4. **Alignment -> aprende preferências**

Continuação de Pré-Treino

- Etapa intermediária entre o pré-treinamento geral e o fine-tuning.
- **Objetivo:** ampliar o conhecimento do modelo em um domínio específico (artigos médicos, textos jurídicos, etc)
- **Método:** retomar o treinamento autos supervisionado (CLM ou MLM) sobre um corpus especializado de grande escala.
  - CPT é como uma “especialização geral” antes do treinamento para tarefas nicho.
- **Resultado:** modelo adquire vocabulário, conceitos e fatos do domínio.

Mecanismos de CLM / MLM

- **Modelagem de Linguagem Causal (CLM):** busca prever o próximo token com base nos tokens anteriores.
  - Funciona da esquerda para a direita (unidirecional).
  - Máscara de atenção impede acesso a tokens futuros.
  - Usada em modelos autorregressivos (GPT, Qwen).

- **Modelagem de Linguagem Mascarada (MLM):** corrompe parte dos tokens com [MASK].
  - Perda calculada apenas sobre os tokens mascarados.
  - Considera contexto à esquerda e à direita (bidirecional).
  - Usada em modelos como BERT.

Contextos de Uso do CPT

- **Quando usar CPT?**
  - Adaptação para domínio técnico (direito, medicina, etc).
  - Atualização temporal (conhecimento além da data de corte do modelo).
  - Adaptação de idioma (português- melhora fluência e compreensão cultural).

- **CPT x Ajuste Fino Supervisionado (SFT):**
  - CPT expande conhecimento de domínio com dados massivos.
  - SFT ensina comportamento específico para tarefas bem definidas.

Ajuste Fino

Fine-tuning

Ajuste Fino (*Fine-tuning*)

- **O paradigma:** Especialização de um modelo generalista (pré-treinado) a fim de transformá-lo em um modelo especialista.
- **Objetivo:** Adaptar o modelo a domínios de conhecimento ou tarefas específicas.
  - **Importante:** O ajuste não busca ensinar o modelo a gerar linguagem do zero, mas sim, busca refinar competências preexistentes.
- **Analogia:** Formação de um profissional médico
  - **Pré-treinamento:** Faculdade de medicina
  - **O paradigma:** Residência médica

Aprendizagem por Transferência (*Transfer Learning*)

- **Definição:** Reutilização de um modelo desenvolvido para uma tarefa inicial como ponto de partida para o treinamento de um modelo em uma segunda tarefa relacionada.
- **Mecanismo:** O conhecimento de um domínio/tarefa de origem é transferido para o destino.
- **Vantagens:**
  - Melhora o desempenho da tarefa alvo.
  - Acelera o aprendizado.
  - Reduz a necessidade de dados e custos proibitivos.

Ajuste-fino de LLMs

- Os LLMs possuem algumas características que as diferem de outros modelos e que devem ser ressaltadas.
  - Os modelos pré-treinados, por si só, costumam adquirir as chamadas **habilidades emergentes**, de forma que são capazes de realizar uma série de tarefas mesmo sem receber nenhum treinamento adicional para isso.
  - Modelos “genéricos” preparados para uma grande gama de tarefas
  - Porém, podem não atingir o desempenho desejado para uma tarefa e/ou domínio específico para qual queremos aplicar este modelo.

Ajuste Fino Supervisionado (SFT)

- **Funcionamento:** O treinamento, a partir de um modelo pré-treinado, é estendido em um conjunto de dados menor do que o do pré-treinamento, porém com dados rotulados.
- **Estrutura de dados:** Utiliza dados rotulados que demonstram o comportamento desejado (ex, pergunta -> resposta).
- **Etapas:**
  - **Seleção do modelo base:** arquitetura, tamanho e afinidade com a tarefa final.
  - **Dataset:** Coleta, limpeza e formatação de dados.
  - **Hiperparâmetros:** Taxa de aprendizado, épocas, etc.
  - **Treinamento, validação e teste:** treinamento em lotes e monitoramento contínuo em dados de validação.

Ajuste Fino Supervisionado (SFT)

- **Exemplos de dados rotulados:**
  - **Análise de sentimento de *tweets* (classificação)**
    - **Tweet (entrada):** “aaaaaaaa amei tanto essas polaroids, nem sei expressar o quanto eu to apaixonada de vdd” - **Sentimento (saída/rótulo):** Positivo
    - **Tweet (entrada):** “eu tava quase conseguindo três estrelas poxa :(“ - **Sentimento (saída/rótulo):** Negativo
  - **Perguntas e respostas sobre a área médica (geração de texto)**
    - **Pergunta (entrada):** “O que é a síndrome de Chediak-Higashi?”
    - **Resposta (saída/rótulo) :** “A síndrome de Chediak-Higashi é uma condição que afeta muitas partes do corpo, especialmente o sistema imunológico. Essa doença danifica as células do sistema imunológico [...]”

Diferenciando Técnicas de Adaptação

- **Continuação de Pré-treino vs Ajuste Fino**
  - **Pré-treino contínuo:** expande o conhecimento geral utilizando dados não rotulados.
  - **Ajuste Fino:** ensina habilidades específicas utilizando dados rotulados, tomando como base um modelo já pré-treinado.
- **Engenharia de Prompts vs Ajuste Fino:**
  - **Engenharia de Prompts:** Busca fazer uma “adaptação” sem treinamento ou modificação de parâmetros, guiando o modelo na inferência através dos próprios dados de entrada.
  - **Ajuste Fino:** Modifica os parâmetros internos do modelo.

Espectro de Ajuste Fino

- A estratégia de adaptação pode variar, gerando trade-offs entre desempenho, custo e complexidade.
- **Principais abordagens:**
  - **Ajuste Fino Completo (FFT):** Modifica todos os parâmetros.
  - **Ajuste Eficiente (PEFT):** Ajusta um subconjunto de parâmetros.
  - **Técnicas de Alinhamento (AI *Alignment*):** Estendem o SFT para alinhar o modelo com preferências humanas.

Ajuste Fino Completo (FFT)

- **Conceito:** A forma mais direta de especialização.
  - Todos os parâmetros do modelo são descongelados e treinados.
- **Mecanismo:** Da primeira até as camadas finais, toda rede neural é atualizada.
- **Vantagens:**
  - Potencial de atingir o desempenho máximo do modelo
  - Concede maior flexibilidade
  - Permite a reconfiguração profunda para aprender nuances complexas e específicas de novos domínios.

Ajuste Fino Completo (FFT)

- **Desvantagens:**
  - Custo computacional e de memória.
  - Armazenamento e implantação.
  - Esquecimento catastrófico
  - Risco de sobreajuste.
  - Amplificação de vieses.

Parameter-Efficient Fine-Tuning (PEFT)

- **Problema:** Fine-tuning tradicional é caro e exige recursos massivos.
- **O que é:** os métodos de *Parameter-Efficient Fine-Tuning* (PEFT) buscam adaptar modelos atualizando frações mínimas de parâmetros.
- **Objetivos:** Reduzir memória/tempo e preservar o conhecimento pré-treinado.
- **Atuação:** Funciona como uma 'camada extra' enquanto o modelo base fica congelado.

Low-Rank Adaptation (LoRA)

- O LoRA (Low-Rank Adaptation) é um método de PEFT proposto em 2021 por um grupo de pesquisadores da Microsoft.

- A técnica LoRA consiste em congelar os pesos pré-treinados do modelo e injetar nas camadas da arquitetura Transformer um conjunto de matrizes decompostas ajustáveis (treináveis), reduzindo assim o número total de parâmetros durante o ajuste.

Low-Rank Adaptation (LoRA)

$$h(x) = W_0 x + \Delta W x = W_0 x + BAx$$

Where, $W_0, \Delta W \in R^{d \times k}$

$B \in R^{d \times r}$

$A \in R^{r \times k}$

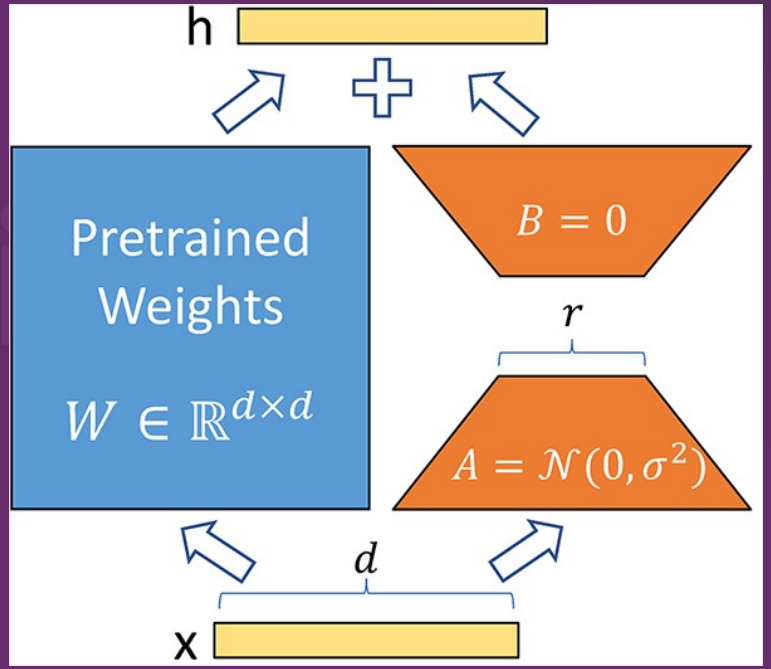
h() = uma camada oculta que será ajustada

x = entrada da camada oculta h

W_o = matriz de pesos original de h

ΔW = matriz de parâmetros treináveis injetada em h

ΔW = BA



Low-Rank Adaptation (LoRA)

- A aplicação da decomposição de matrizes permite que sejam treinadas matrizes menores que a matriz de peso original, tornando o treino menos custoso.
- Exemplo:
  - Imagine que W tem dimensões $d \times k = 512 \times 64$
  - Aplicando LoRA com rank $r = 8$, teriamos as matrizes:
    - $A: r \times k \rightarrow \quad 8 \times 64 = 512$ parâmetros
    - $B: d \times r \rightarrow 512 \times 8 \quad = 4096$ parâmetros

  - Total de parâmetros treináveis sem LoRA: 32.768
  - Total de parâmetros treináveis com LoRA: 4608
  - **Redução de ~86%** de parâmetros a serem ajustados

QLoRA

- O **QLoRA (Quantized LoRA)**, por sua vez, consiste na aplicação do LoRA em modelos quantizados.
  - A quantização é uma técnica usada para reduzir o tamanho de uma LLM ao mesmo tempo que tenta manter sua qualidade. Pode-se fazer uma analogia com o processo de compressão de imagens.
  - A quantização consiste em representar os pesos do modelo com uma menor precisão.
- Objetivo do QLoRA: reduzir a necessidade de VRAM necessária para o treinamento de uma LLM

QLoRA

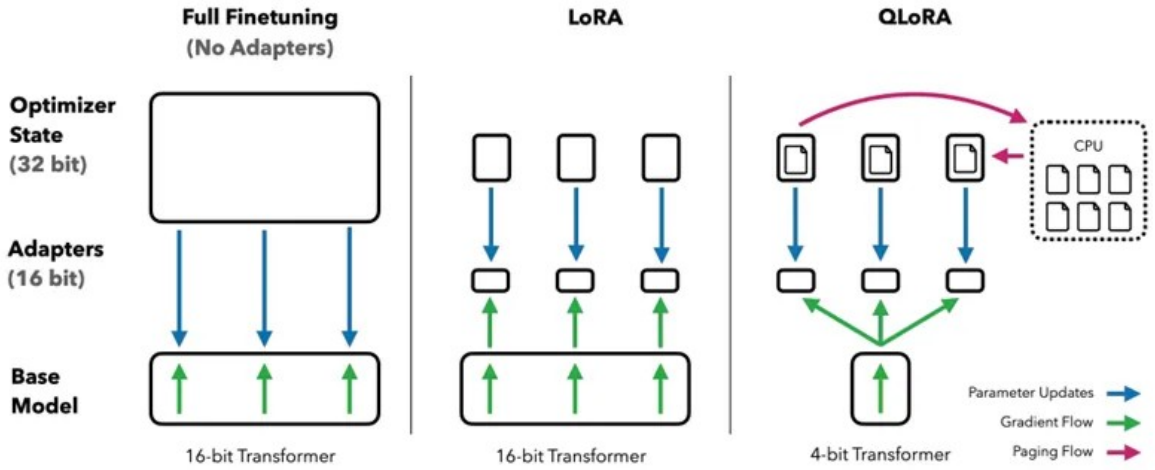


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

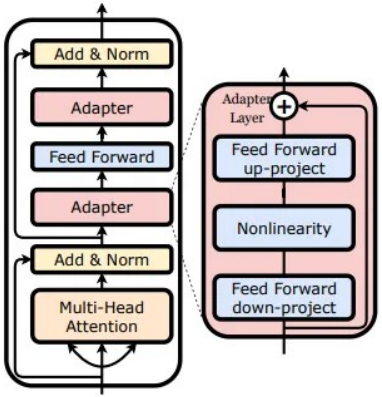
QLoRA

- Vantagens:
  - Obtém resultados semelhantes ao LoRA
  - Menor consumo de memória (diminui requisitos de *hardware*, especialmente de GPU)
  - Melhor capacidade em lidar com picos de memória

- Desvantagens:
  - Treino mais lento devido às fases de quantização e de desquantização.
    - Exemplo: em um experimento real, o uso do QLoRA, em comparação ao LoRA, trouxe uma economia de 33% do uso de memória da GPU, porém com um aumento de 39% no tempo de treinamento.

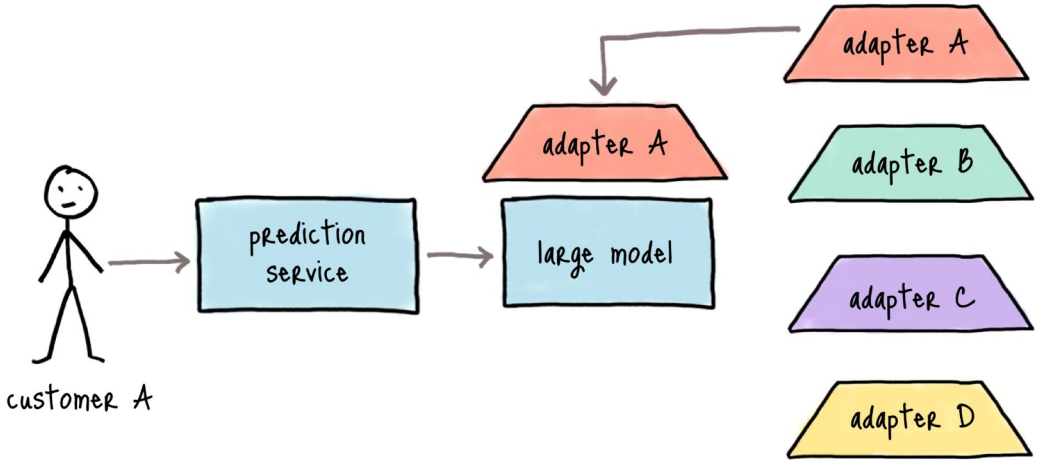
Adapter

- ***Adapters*** são **módulos treináveis**, criados para serem leves e facilmente integráveis em diversos pontos da arquitetura de um LLM. Ao realizar o fine-tuning destes módulos, ao invés do modelo todo, os adapters facilitam a customização de modelos pré-treinados para tarefas específicas.
  - Normalmente um adapter consiste em 2 camadas fully-connected com uma camada de ativação não linear entre elas.
  - Podemos utilizar adapters juntamente com LoRA ou o QLoRA



Adapter

- Uma propriedade interessante do uso de *Adapters* é que o modelo original se mantém intacto, preservando o conhecimento adquirido no pré-treinamento.
  - Por este motivo, os *Adapters* também são cambiáveis, de forma que podemos treinar diferentes *adapters* para tarefas/línguas/domínios específicos e trocar quais estão carregados no modelo conforme a conveniência.



In-Context Learning

- **In-Context Learning:** Aprendizado na inferência via *prompt* (sem alterar pesos).
  - Através dessa técnica, dados rotulados são inseridos diretamente no *prompt* para servir de exemplos, para o modelo "aprender" a partir destes exemplos
  - **One e Few-Shot Prompting**: idealmente não deveríamos chamar de **One ou Few-Shot Learning**, visto que não temos ajustes nos parâmetros no modelo, mas por estar associado ao In-Context Learning, muitas vezes encontraremos essas nomenclatura.
- **Zero-Shot:** o modelo recebe apenas a descrição em linguagem natural da tarefa, sem nenhum exemplo de como executá-la.
- **One-Shot:** O modelo recebe a descrição da tarefa acompanhada de um único exemplo.
- **Few-Shot:** O modelo recebe a descrição da tarefa e múltiplos exemplos (tipicamente entre 2 a poucas dezenas).

Few-Shot Learning

- Usando a definição clássica de **Few-Shot Learning**, podemos realizar o ajuste fino supervisionado do modelo um conjunto de dados deliberadamente pequeno, mas de alta qualidade.
- **LIMO (*Less-is-More Reasoning Hypothesis*)**: postula que em modelos onde o conhecimento de um domínio já foi extensivamente codificado, o raciocínio sofisticado pode emergir a partir de demonstrações mínimas, mas estrategicamente desenhadas, de processos cognitivos.

Few-Shot Learning

- **Vantagens**
  - Dispensa a necessidade de grandes bases de dados
  - Treinamento menos custoso
  - Ajuda a evitar o esquecimento catastrófico

- **Desvantagens**
  - Para uma adaptação de domínio pode não ser suficiente, pois um conjunto pequeno de amostras pode não contemplar todo o conhecimento necessário para que o modelo consiga responder adequadamente qualquer instrução sobre este domínio

AI Alignment

Conceito e métodos

AI Alignment

- **Definição:** AI Alignment é o desafio de garantir que sistemas de IA otimizem objetivos compatíveis com intenções e valores humanos.
- O problema se torna mais crítico à medida que LLMs, agentes autônomos e sistemas de decisão ganham maior autonomia e capacidade de generalização.
- Trata-se de um tema central em segurança de IA, pois comportamentos aparentemente corretos podem divergir dos objetivos humanos reais.

Motivação

- Em aplicações reais, muitos objetivos envolvem **critérios subjetivos, valores éticos** e múltiplas dimensões de decisão.
- Nesses cenários, definir manualmente uma função de recompensa totalmente alinhada é frequentemente impraticável.
- Pequenos desalinhamentos podem gerar comportamentos perigosos ou não intencionais, especialmente em contextos críticos.

Autonomia e risco

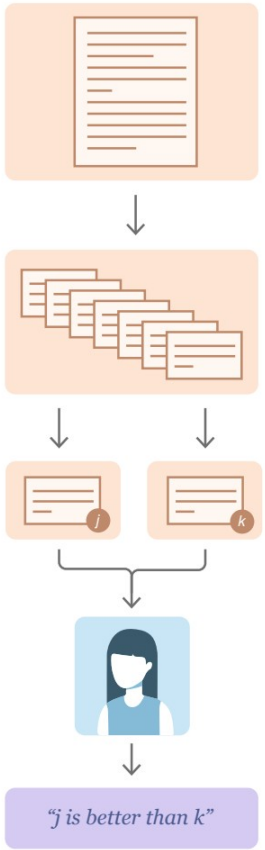
- Agentes artificiais percebem o ambiente, tomam decisões e executam ações de forma autônoma.
- Exemplos: assistentes conversacionais, robôs autônomos, sistemas de recomendação e LLMs integrados a agentes de planejamento.
- Quanto maior o **poder de ação** e a **autonomia** do agente, maior o potencial de **desalinhamento**.

RLHF

- O RLHF utiliza **comparações humanas** entre respostas para treinar um **modelo de recompensa**.
- Em seguida, a política do modelo é otimizada para maximizar essa recompensa, geralmente com **PPO**.
- Vantagem: maior flexibilidade e potencial de generalização.
- Limitações: alto custo computacional, maior complexidade e risco de *reward hacking*.

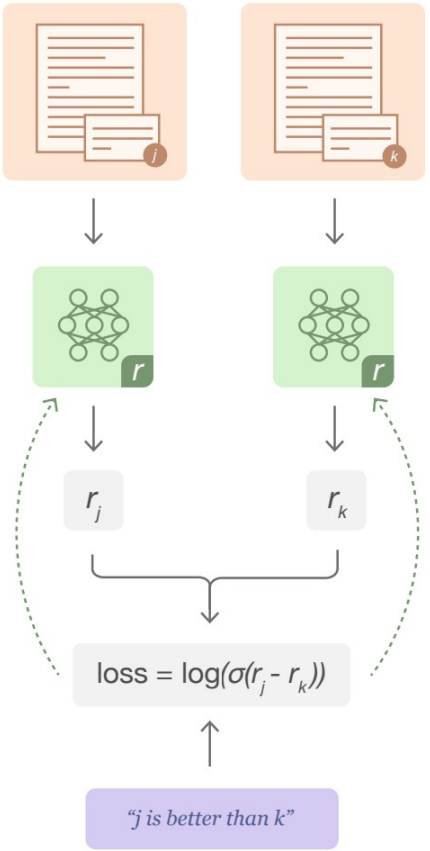
RLHF

- **1º passo:** Coletar *feedback* humano
  - São coletadas ou geradas possíveis respostas para uma determinada entrada;
  - Respostas possíveis são avaliados por um anotador humano;
  - Anotador humano classifica das respostas da melhor a pior.



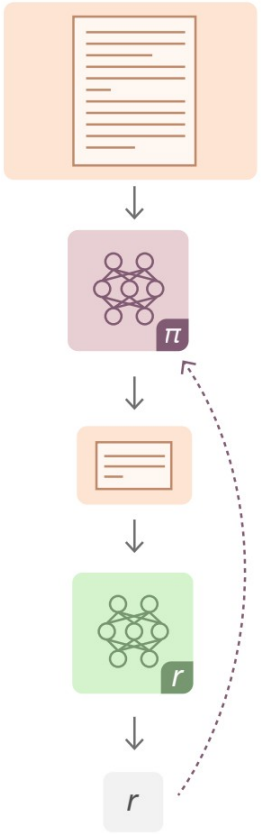
RLHF

- **2º passo:** treinar modelo de recompensa
  - A partir do feedback humano, treina-se um modelo para emitir uma pontuação para um par (*prompt*, resposta)
  - Este modelo pode ser treinado como um simples classificador:
    - 0 = par rejeitado
    - 1 = par aceito



RLHF

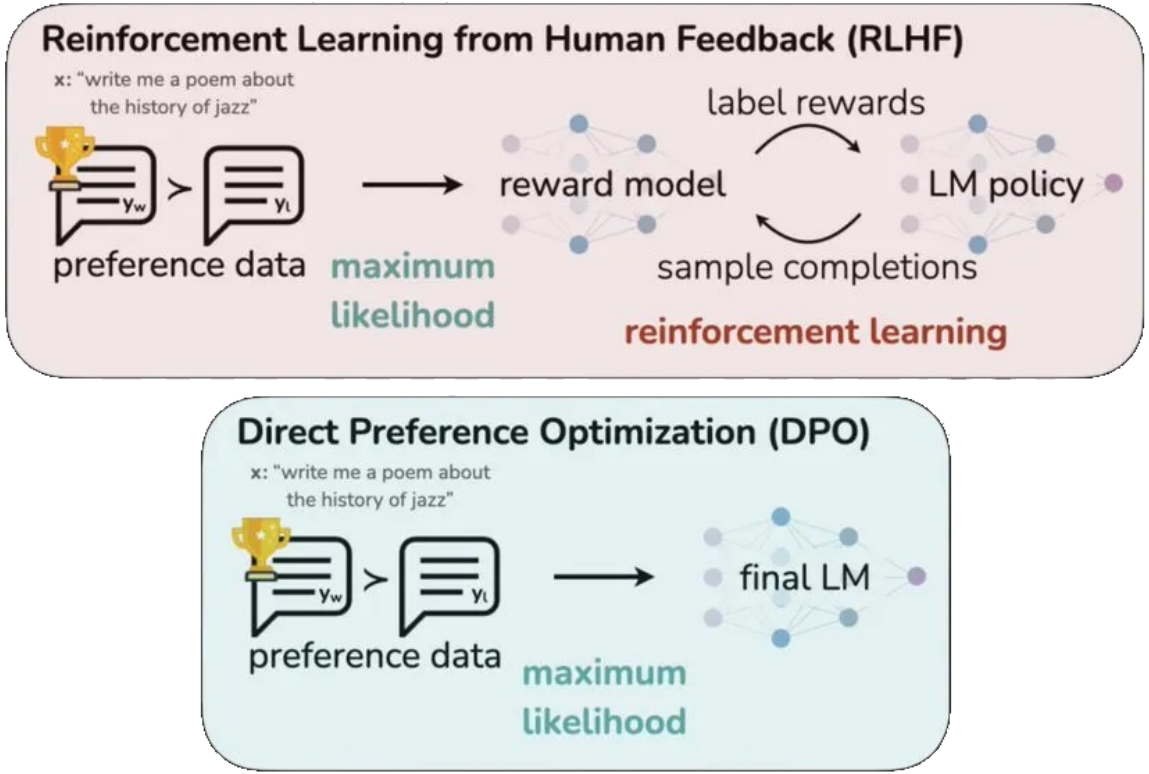
- **3º passo:** RLHF *fine-tuning*
  - Por fim, se realiza o ajuste-fino do LLM utilizando o modelo de recompensa treinado no passo anterior.
  - Para isso se utiliza algum algoritmo de aprendizado por reforço. No geral, aplica-se o *Proximal Policy Optimization* (PPO)



DPO

- O DPO elimina a necessidade de aprender explicitamente um modelo de recompensa.
- A política é otimizada **diretamente a partir de preferências humanas**, usando uma perda supervisionada.
- Vantagens: **simplicidade**, **estabilidade** e **eficiência computacional**.
- Limitação principal: forte dependência da qualidade dos dados de preferência.

DPO


Reinforcement Learning from Human Feedback (RLHF)
x: "write me a poem about the history of jazz"
preference data
maximum likelihood
label rewards
reward model
LM policy
sample completions
reinforcement learning
Direct Preference Optimization (DPO)
x: "write me a poem about the history of jazz"
preference data
maximum likelihood
final LM

Estudo de Caso

Dr. Bode

Estudo de Caso

Para acessar o código, basta apontar sua câmera
para o QR Code ao lado ou copiar o link abaixo.

URL:
https://colab.research.google.com/drive/1qwdmps8a3uZOa4omIRCwwS9nnrpT29XM#scrollTo=B3a6Xtwmw_80&uniqifier=2

Obrigado!