Instruction Finetuning

Tópicos em Ciência de Dados

Prof. Dr. Denis Mayr Lima Martins

Pontifícia Universidade Católica de Campinas

Objetivos de Aprendizagem

- Explicar a necessidade e função do Ajuste Fino de Instruções (AFI)
- Descrever a metodologia de treinamento e perda: detalhar a estrutura dos dados de AFI (instrução, contexto, resposta alvo) e explicar a função da perda
- Avaliar o desempenho do modelo: utilizar métricas de avaliação e entender o papel de modelos externos (*LLM-as-a-Judge*) na avaliação da qualidade do alinhamento.

Baseado no Livro Build a Large Language Model From Scratch de Sebastian Raschka

Code repository: https://github.com/rasbt/LLMs-from-scratch

Relembrando: O Conceito de Fine-Tuning

- **Definição:** É o processo de utilizar um modelo pré-treinado como base e treiná-lo adicionalmente em um *dataset* menor e específico de um domínio ou tarefa.
- **Objetivo:** Adaptar o modelo ao novo contexto, aprimorando o desempenho em aplicações especializadas, como tradução de linguagem, análise de sentimento ou sumarização.
- **Vantagem:** O *fine-tuning* se baseia no conhecimento pré-existente do modelo, o que reduz substancialmente os requisitos computacionais e de dados em comparação com o treinamento do modelo do zero (*pre-training*).

Por Que os LLMs Tradicionais Falham com Diretivas

- **Objetivo do Pré-treinamento:** Os LLMs são otimizados para o reconhecimento de padrões linguísticos, minimizando o erro de previsão contextual da próxima palavra em vastos *corpora*.
    - O modelo prevê o próximo *token* em uma sequência com base em padrões estatísticos.
- **A Limitação:** Este objetivo de previsão do próximo *token* não otimiza inerentemente o modelo para seguir instruções explícitas do usuário.
    - Sem treinamento adicional, um LLM de base simplesmente *completa* um *prompt*, em vez de fornecer uma *resposta* útil.
    - Exemplo: Solicitar "*me ensine a fazer pão*" pode resultar em "*em um forno de casa*" (uma conclusão gramaticalmente correta, mas inútil).
- **A Solução:** O AFI refina o modelo pré-treinado para interpretar as consultas do usuário como instruções formais que exigem ações específicas.

O Que é Ajuste Fino de Instruções (AFI)?

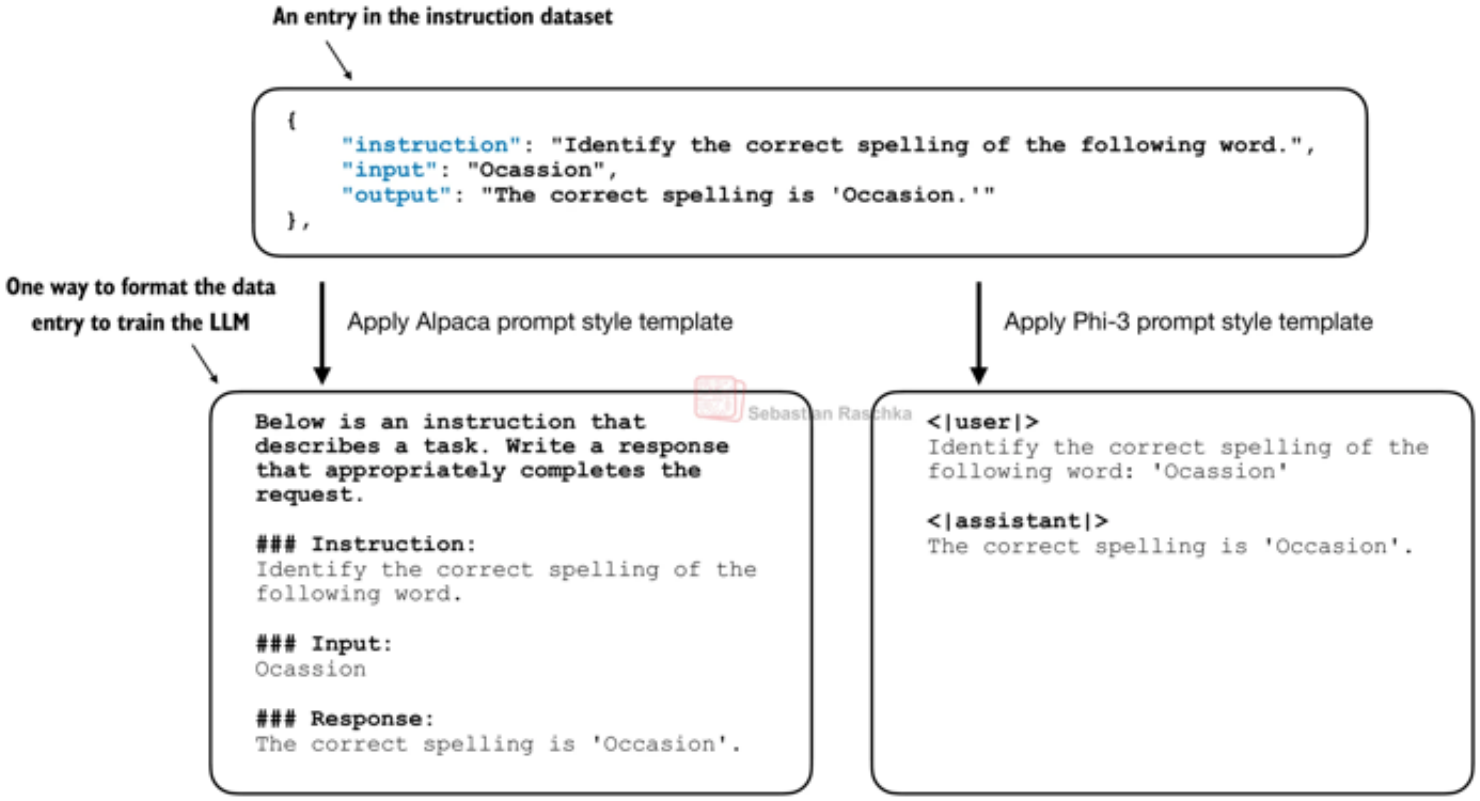
- **Definição:** AFI é uma técnica de ajuste fino que refina LLMs pré-treinados para aderir a instruções de tarefas específicas.
- **Metodologia:** Envolve treinamento supervisionado em conjuntos de dados que consistem em pares explícitos de *prompt*-resposta.
- **Função Chave:** O AFI preenche a lacuna entre a capacidade inerente de previsão da próxima palavra do LLM e o objetivo definido pelo ser humano de aderir a diretivas.
- **Benefícios:**
    1. **Alinhamento:** Conecta o objetivo de pré-treinamento com o objetivo de seguir instruções.
    2. **Controlabilidade:** Restringe os *outputs* do modelo para alinhá-los com as características desejadas (e.g., formato ou conhecimento de domínio).
    3. **Generalização:** Modelos ajustados por instruções demonstram forte desempenho *zero-shot* e *few-shot* em tarefas não vistas.

Anatomia de uma Amostra de Dados AFI

O AFI requer pares de instruções e seus *outputs* de alta qualidade correspondentes.

- **1. Instrução (A Diretiva):** Define claramente a tarefa necessária.
    - *Exemplo:* "Traduza a seguinte frase para o Francês".
- **2. Input/Contexto (O Conteúdo):** Informações suplementares opcionais relevantes para a tarefa.
    - *Exemplo:* "A frase a traduzir: 'O processo de ajuste fino é complexo.'".
- **3. Resposta Alvo (A Resposta Ouro):** O *output* de referência de alta qualidade que demonstra a conclusão correta da tarefa.

Prompt Style Template



Estratégias de Coleta de Dados I

A curadoria e o dimensionamento de pares de instrução-*output* de alta qualidade são desafios centrais do AFI.

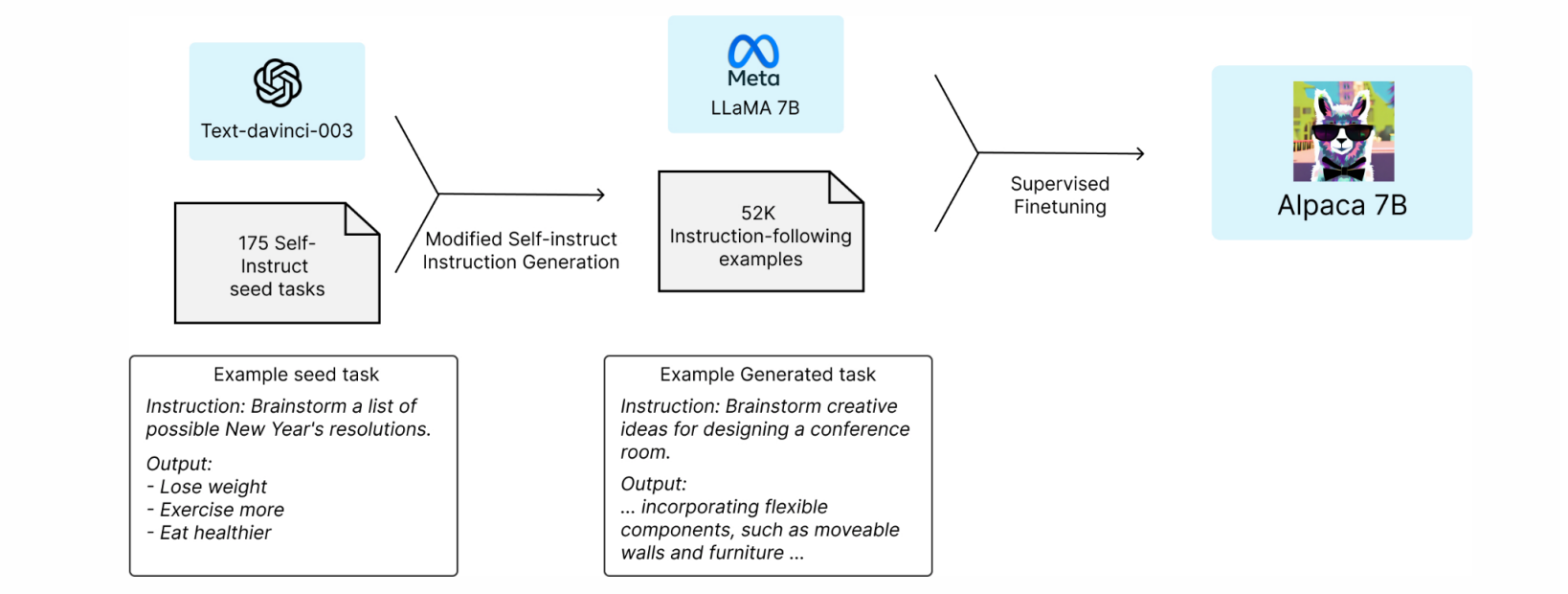
- **1. Dados Criados por Humanos:** Dados anotados manualmente ou obtidos diretamente, confiando apenas na coleta e verificação humana.
    - **Prós:** Geralmente a mais alta qualidade e consistência.
    - **Contras:** Demorado e custoso para grandes escalas.
    - Exemplos: Databricks Dolly (15K instâncias, abrangendo 7 tipos como Q&A, escrita criativa), Meta LIMA (1K exemplos cuidadosamente selecionados).
- **2. Integração de Dados de Conjuntos de Dados Anotados:**
    - Envolve a conversão de conjuntos de dados de PLN existentes (e.g., NLI, análise de sentimentos) em pares de instrução-*output* usando *templates*.
    - Isto formaliza diversas tarefas de PLN em um formato unificado *sequence-to-sequence*.
    - Exemplos: FLAN (transforma 62 *benchmarks* de PLN), P3 (integra 170 conjuntos de dados de PLN).

Estratégias de Coleta de Dados II

LLMs podem ser usados para aumentar conjuntos de dados de AFI quando a criação manual é inviável.

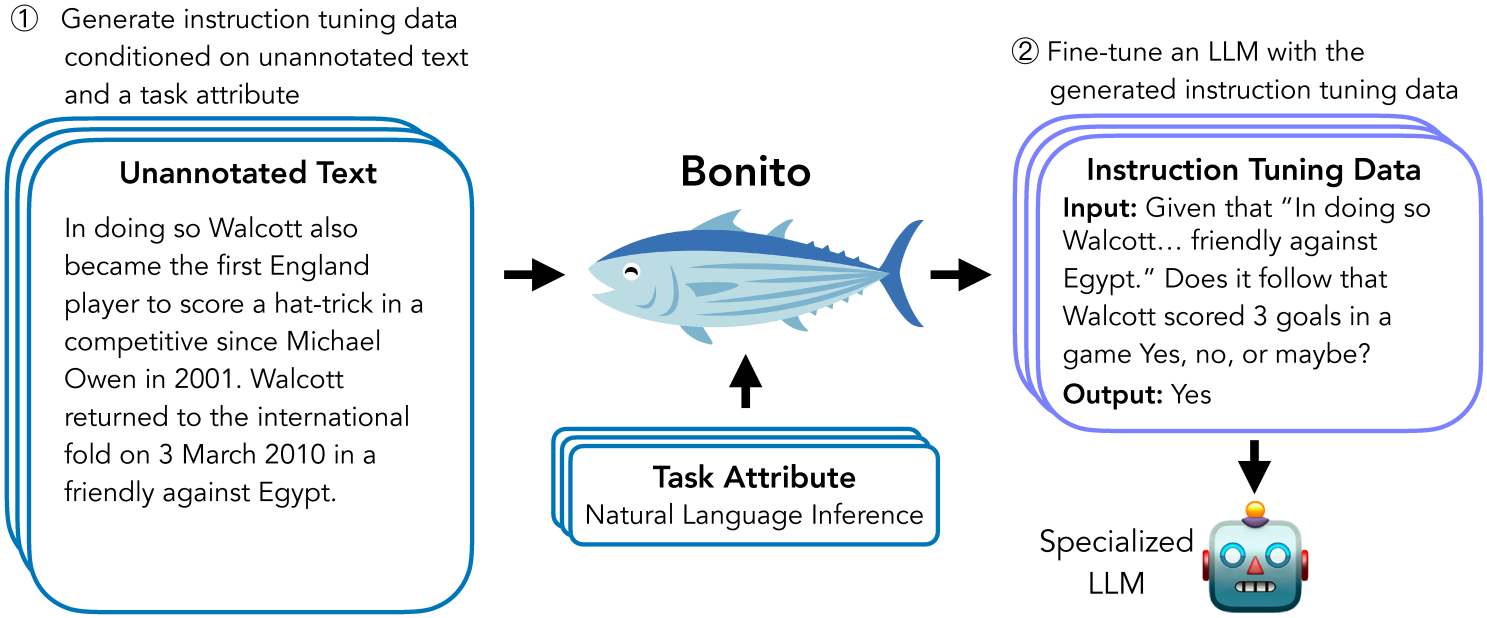
- **Self-Instruct**: Começa com um pequeno conjunto de pares sementes. Um LLM gera novas instruções e outra instância gera *outputs* correspondentes.
    - *Exemplo:* O modelo Alpaca usou 52K pares sintéticos gerados desta forma.
- **Bonito**: Converte texto não anotado em datasets de treino para AFI. Modelo base: Mistral-7B
- **Magpie**: Gera dados de instrução solicitando a um LLM alinhado (e.g., Llama 3 8B Instruct) com um *template* de pré-consulta para sintetizar instruções e respostas de forma totalmente automática.

Alpaca



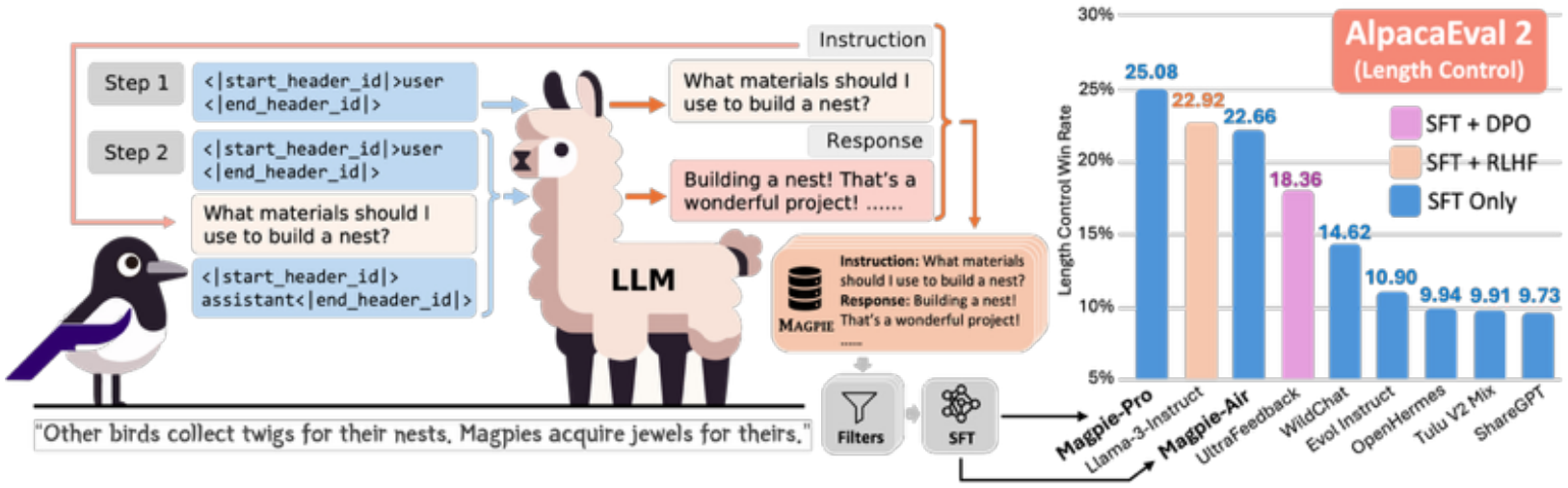
Workflow Alpaca. AFI sobre o modelo base Llama-7B. Qualidade similar ao modelo da OpenAI, mas muito menor e mais barato de reproduzir. Fonte: Weights and Biases.

Bonito



Workflow do framework Bonito. Fonte: Learning to Generate Instruction Tuning Datasets for Zero-Shot Task Adaptation.

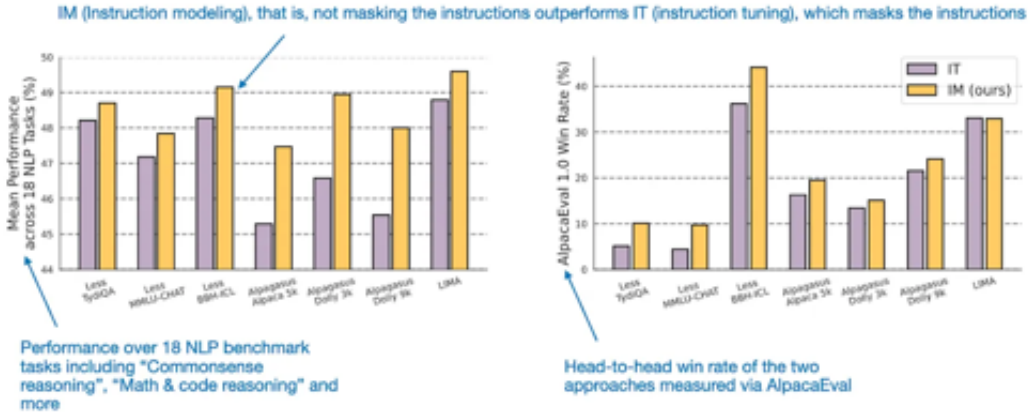
Magpie



Workflow do framework Magpie. Step 1: apenas pre-query template como entrada para LLM e geração autorregressiva de instrução. Step 2: Combinação de post-query template e outra pre-query template. Fonte: Magpie: Alignment Data Synthesis from Scratch by Prompting Aligned LLMs with Nothing.

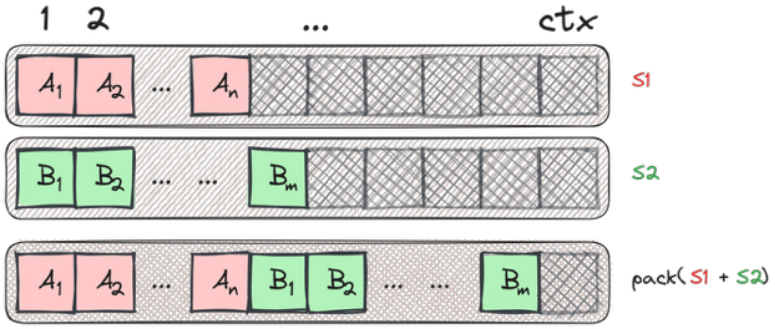
Pré-processamento

1. **Formatação do Prompt:** Adote um estilo de *prompt* consistente (e.g., Alpaca) para todas as amostras de treinamento.
2. **Tokenização:** Converta o texto formatado de instrução-resposta em IDs de *token*.
3. **Colagem/Preenchimento Customizado (packing):** Uma função de colagem customizada é usada para preencher sequências dentro de um lote (ou *batch*) até o comprimento da sequência mais longa nesse lote.
4. **Mascaramento de Instrução (Opcional):** Mascarar IDs de *token* que correspondem à instrução impede que a função de perda seja calculada sobre o texto da instrução. Isto força o modelo a focar o treinamento na geração da *resposta*. Mas... (veja figura ao lado).

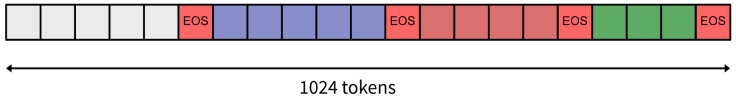


Instruction Masking x Instruction Modeling (não mascara instrução) em (https://arxiv.org/html/2405.14394v2). Fonte: Raschka.

Packing



Packing: Combinando múltiplas amostras em uma única sentença. Fonte: Laurens Weitkamp.



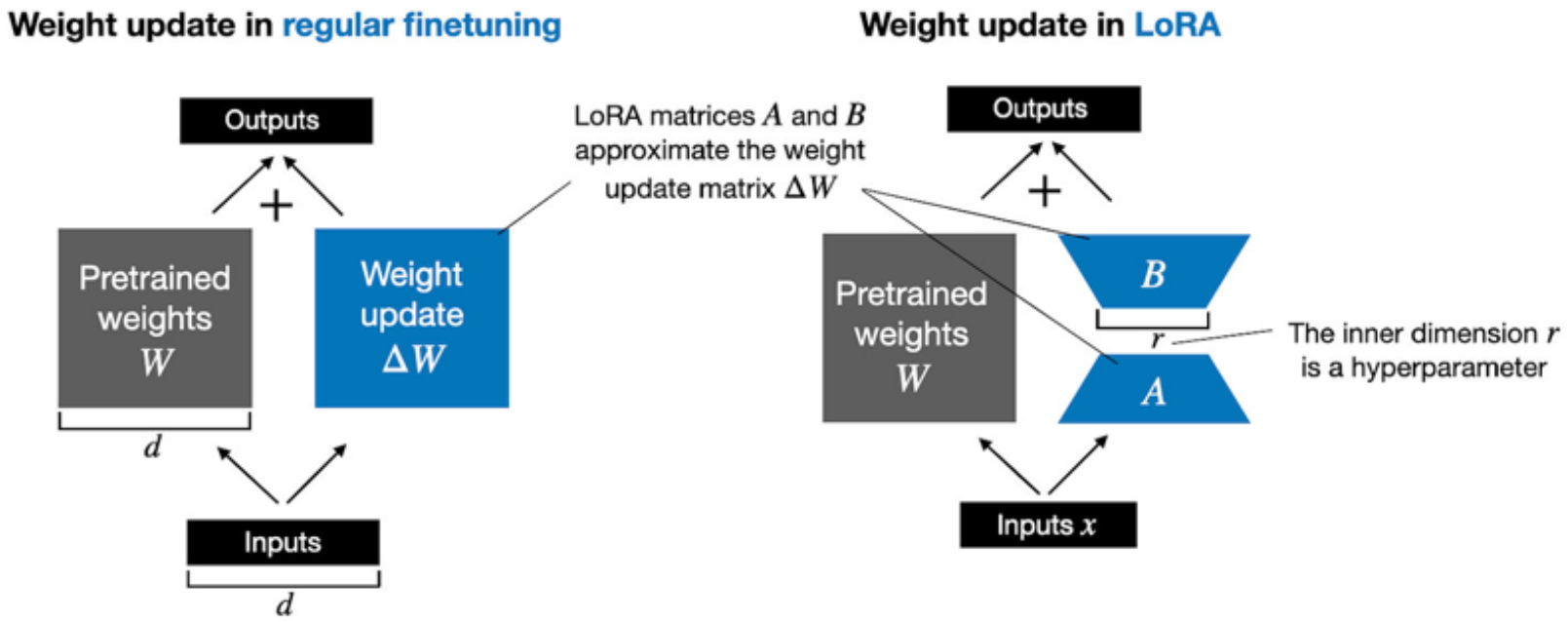
Packing: Otimização do tamanho do contexto. Fonte: Weights and Biases.

Eficiência: Ajuste Fino com Eficiência de Parâmetros (PEFT)

O Ajuste Fino Completo (*Full Fine-Tuning*) é caro e corre o risco de *esquecimento catastrófico*. Os métodos PEFT reduzem drasticamente os custos.

- **Low-Rank Adaptation (LoRA):** A técnica PEFT mais comum.
    - Mantém a maioria dos parâmetros LLM pré-treinados congelados.
    - Injeta pequenas matrizes de decomposição de baixa classificação (A e B) nos parâmetros de atenção.
    - Reduz drasticamente o número de parâmetros treináveis (e.g., 10.000x de redução para GPT-3) e o uso de memória.
- **Quantized LoRA (QLoRA):** Uma extensão do LoRA otimizando ainda mais a memória.
    - Quantiza os pesos base do LLM congelado para precisão ultrabaixa (e.g., 4-bit).
    - Permite o ajuste fino de modelos de alta qualidade usando uma única GPU de consumo.
- Veja também o vídeo no Youtube: LoRA explained (and a bit about precision and quantization).

LoRA



Finetuning convencional x finetuning LoRA. Fonte: Raschka.

In [ ]:

```
import torch.nn as nn

class LoRALayer(nn.Module):
    def __init__(self, in_dim, out_dim, rank, alpha):
        super().__init__()
        std_dev = 1 / torch.sqrt(torch.tensor(rank).float())
        self.A = nn.Parameter(torch.randn(in_dim, rank) * std_dev)
        self.B = nn.Parameter(torch.zeros(rank, out_dim))
        self.alpha = alpha

    def forward(self, x):
        x = self.alpha * (x @ self.A @ self.B)
        return x
```

In [ ]:

```
class LinearWithLoRA(nn.Module):

    def __init__(self, linear, rank, alpha):
        super().__init__()
        self.linear = linear
        self.lora = LoRALayer(
            linear.in_features, linear.out_features, rank, alpha
        )

    def forward(self, x):
        return self.linear(x) + self.lora(x)
```

LoRA

LoRA can even outperform full finetuning training only 2% of the parameters

Full finetuning
Only tune bias vectors
Prompt tuning
Prefix tuning

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

ROUGE scores

Table 4: Performance of different adaptation methods on GPT-3 175B. We report the logical form validation accuracy on WikiSQL, validation accuracy on MultiNLI-matched, and Rouge-1/2/L on SAMSum. LoRA performs better than prior approaches, including full fine-tuning. The results on WikiSQL have a fluctuation around $\pm 0.5\%$, MNLI-m around $\pm 0.1\%$, and SAMSum around $\pm 0.2/\pm 0.2/\pm 0.1$ for the three metrics.

Resultados LoRA. Fonte: Lightning AI.

Métricas de Avaliação I: Quantitativas e Técnicas

A avaliação mede o quão bem o modelo ajustado fino generaliza e adere aos objetivos. Leia mais em Patterns for Building LLM-based Systems & Products.

- **Cross-Entropy Loss (Perda de Entropia Cruzada):** A métrica fundamental monitorada durante o treinamento e a validação.
    - Quantifica a diferença entre a distribuição de probabilidade prevista pelo modelo e a distribuição real de *tokens*.
- **Métricas de PLN Tradicionais (para tarefas específicas):**
    - **BLEU:** Mede a proximidade entre traduções geradas e de referência (Tradução Automática, Sumarização).
    - **Acurácia/F1 Score:** Usado para tarefas de classificação e QA.
- **Avaliação de Codificação:**
    - **HumanEval**: Consiste em 164 problemas de programação para avaliar a capacidade do modelo de gerar programas corretos a partir de *docstrings*.
- **Aderência à Instrução:**
    - **IFEval (Instruction Following Evaluation)**: Testa especificamente a capacidade de um modelo de seguir restrições explícitas, como contagem de palavras ou formatação de *output* necessária.

Métricas de Avaliação II: Alinhamento e Pontuação

Para geração aberta, as métricas puramente automatizadas geralmente são insuficientes, exigindo avaliação centrada no ser humano.

- **LLM-como-Juiz (*LLM-as-a-Judge*):** Utiliza um LLM altamente capaz (e.g., Llama 3 8B) para avaliar a qualidade dos *outputs*.
    - O modelo Juiz recebe o *input*, o *output* correto e a resposta do modelo ajustado fino, fornecendo uma pontuação numérica (e.g., 0 a 100).
    - Isto é eficiente para avaliação em grande escala.
- **Benchmarking de Alinhamento:**
    - **MT-Bench:** Usa 80 questões multi-turno de alta qualidade para avaliar o alinhamento com a preferência humana, cobrindo tarefas como escrita, codificação e raciocínio.
    - **WildBench:** Curado a partir de interações reais do usuário, apresentando 1.024 instruções desafiadoras que exigem pensamento crítico.
- **Métricas de Segurança:** Avaliam respostas do LLM quanto a toxicidade, viés e aderência a diretrizes de segurança. Exemplos incluem Llama Guard 2/3 e ShieldGemma.

Armadilhas Comuns e Limitações em AFI

O AFI está sujeito a modos de falha específicos que podem minar a utilidade a longo prazo.

- **Esquecimento Catastrófico (*Catastrophic Forgetting*):** O ajuste fino em novas tarefas pode fazer com que o modelo perca o conhecimento pré-treinado.
    - *Mitigação:* Usar técnicas PEFT (LoRA/QLoRA) para congelar a maioria dos pesos base.
- **Alinhamento Superficial (*Superficial Alignment*):** O modelo aprende apenas padrões de superfície e estilos (e.g., formato de *output* ou tom) em vez de melhorar o raciocínio subjacente.
    - Isto levanta a preocupação de que os ganhos de desempenho dependam fortemente das tarefas representadas no dado de treinamento.
- **Dependência da Qualidade dos Dados:** O desempenho depende criticamente da qualidade, diversidade e cobertura de tarefas do conjunto de dados de instrução.
    - Dados mal selecionados (especialmente sintéticos) podem reforçar vieses ou deficiências.

Resumo e Leitura Adicional

- O AFI (SFT) é essencial para alinhar a previsão do próximo *token* dos LLMs com os objetivos do usuário.
- O AFI depende de pares de instrução-resposta de alta qualidade e diversificados, gerados manualmente (Flan, Dolly) ou sinteticamente (Self-Instruct, Evol-Instruct).
- O processo de treinamento usa uma perda de objetivo duplo e se beneficia de aprimoramentos arquiteturais (e.g., arquitetura de dois fluxos) e PEFT (LoRA/QLoRA).
- A avaliação requer métricas quantitativas (Entropia Cruzada, HumanEval) e técnicas qualitativas (LLM-como-Juiz, checagens de segurança).
- **Leitura Adicional**:
    - Instruction Pretraining LLMs
    - Instruction Tuning for Large Language Models: A Survey
    - The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities