Documentação Completa: Fine-Tuning do Llama 3 para Geração de Descrições de Produtos

Projeto: FIAP-TC3-Llama3-FineTune

Autor: Robson Nicácio R. dos Santos

Data: 25 de Setembro de 2025

1. Introdução e Objetivo

Este documento detalha o processo de desenvolvimento do Tech Challenge da Fase 03 da Pós-Graduação "IA para Devs" da FIAP. O objetivo central do projeto foi executar o fine-tuning de um *foundation model* de linguagem (LLM) para uma tarefa específica: gerar descrições de produtos da Amazon a partir de seus títulos.

O projeto seguiu um pipeline completo de Machine Learning, desde a análise e preparação do dataset até o treinamento, avaliação e deploy do modelo final.

2. Seleção do Foundation Model e Framework

2.1. Modelo Base: unsloth/llama-3-8b-bnb-4bit

Para este projeto, o modelo escolhido como base foi o **Llama 3 8B** da Meta, especificamente a versão otimizada unsloth/llama-3-8b-bnb-4bit. A escolha foi motivada pelos seguintes fatores:

- **Desempenho:** O Llama 3 é um dos modelos de código aberto mais avançados disponíveis, com uma forte capacidade de compreensão de linguagem e geração de texto coerente.
- **Eficiência:** A versão da Unsloth já vem quantizada em 4-bit (bnb-4bit), o que reduz drasticamente o consumo de memória VRAM, tornando viável o fine-tuning em GPUs com recursos limitados, como a Nvidia T4 disponível no Google Colab.
- **Formato Nativo:** O modelo possui um template de chat bem definido, o que é crucial para o sucesso do fine-tuning de instrução.

2.2. Framework de Treinamento: Unsloth

A biblioteca **Unsloth** foi utilizada como o principal framework de treinamento. Ela é uma camada de otimização sobre as bibliotecas transformers, PEFT e bitsandbytes, que acelera significativamente o processo de fine-tuning (até 2x mais rápido) e otimiza o uso de memória, permitindo treinar modelos maiores com menos recursos.

3. Preparação do Dataset

Esta foi a etapa mais crítica e iterativa do projeto. O dataset utilizado foi o **"The Amazon Titles-1.3MM"**, com foco no arquivo trn.json.

3.1. Análise Inicial

O dataset consiste em registros JSON, cada um contendo, entre outros campos, title (título do produto) e content (descrição).

A análise inicial de uma amostra dos dados revelou dois problemas principais:

1. **Dados Sujos:** Muitas descrições continham entidades HTML (ex: ’ em vez de ').
2. **Texto *Boilerplate***: Uma grande quantidade de registros terminava com frases repetitivas e de baixo valor, como "--This text refers to the Hardcover edition.".

3.2. Processo de Limpeza e Formatação

Para preparar os dados para o fine-tuning, foi desenvolvido um script em Python com as seguintes etapas:

Iteração 1: Limpeza Básica e Formatação do Prompt

Inicialmente, o foco foi limpar o básico e estruturar o prompt.

```
import json
import os
import html

# ... (definição de caminhos)

LLAMA3_PROMPT_TEMPLATE = """<|begin_of_text|><|start_header_id|>user<|end_header_id|>
```

```
Com base no título do produto, gere a sua descrição.
Título: {}<|eot_id|><|start_header_id|>assistant<|end_header_id|>

{}<|eot_id|>"""

# ... (loop de processamento)
    title = html.unescape(original_record.get('title', ''))
    content = html.unescape(original_record.get('content', ''))
# ...
```

- **Adoção do Template Nativo:** Foi crucial adotar o template de prompt nativo do Llama 3 para que o modelo entendesse corretamente a estrutura de diálogo e a tarefa.

Iteração 2: Remoção de Texto *Boilerplate*

Após um primeiro treinamento, notou-se que o modelo aprendia a gerar o texto boilerplate. Para resolver isso, o script foi aprimorado com expressões regulares (re) para remover essas frases indesejadas.

```
import re

# ...
content = html.unescape(original_record.get('content', ''))
# Remove a frase "--This text refers to..." e qualquer variação dela.
content = re.sub(r'--This text refers to.*', '', content).strip()

# Se após a limpeza o conteúdo ficar vazio, pule o registro.
if not content:
    continue
# ...
```

Iteração 3: Resolvendo o *Overfitting* Estrutural

Mesmo com os dados limpos, o modelo treinado começou a gerar tokens de controle (<|reserved_special_token_...|>) ao final de suas respostas. Isso indicou que ele havia aprendido a estrutura do prompt de forma excessiva, mas não sabia quando a conversa deveria terminar definitivamente.

A solução foi adicionar o token <|end_of_text|> ao final de cada exemplo no dataset, ensinando ao modelo um sinal de parada inequívoco.

```
# Template final, ensinando o modelo a parar
LLAMA3_PROMPT_TEMPLATE = """<|begin_of_text|><|start_header_id|>user<|end_header_id|>

Com base no título do produto, gere a sua descrição.
Título: {}<|eot_id|><|start_header_id|>assistant<|end_header_id|>

{}<|eot_id|><|end_of_text|>""" # <--- TOKEN ADICIONADO AQUI
```