



IA PARA DEVS

OPEN API

FASE 03

TechChallenge Fase 3

Pós-Tech em IA para DEVS

FIAP

INTEGRANTES DO GRUPO 18

Beatriz Cardoso Cunha
Francisco Giuan Miranda Ferreira
Maurício Lachaitis da Silva

17 de dezembro de 2024



Relembrando o Desafio do Tech Challenge

Objetivo Principal:

1. Executar o fine-tuning de um modelo de linguagem (ex.: LLaMA, BERT, GPT) usando o dataset "The AmazonTitles-1.3MM".
2. Receber perguntas dos usuários com base em um contexto (título do produto).
3. Gerar respostas baseadas na descrição do produto após o fine-tuning.
4. Documentar e Apresentar:
5. Explicar os parâmetros, ajustes e resultados.
6. Criar um vídeo demonstrando o modelo em ação.

Dataset:

O dataset contém títulos de produtos e suas descrições provenientes da Amazon.



Links Principais:

- https://github.com/beatrizcardc/TC_fase3_Fine_tuning_llama3.git
- <https://youtu.be/yFqfh31Yg10>

Pipeline Completo do Trabalho

Escolha do Dataset

- **Dataset:** The AmazonTitles-1.3MM.
 - **Campos:** Utilizar os campos **título** e **descrição** dos produtos.
 -  **Download e Armazenamento:** Realizar download e armazenar de forma segura no **Google Drive**.
-

Preparação do Dataset

-  **Carregar e Inspecionar:** Carregar e revisar os dados.
 -  **Limpeza:**
 - Remover linhas com valores **ausentes**.
 - Remover **duplicatas**.
 - Normalizar o texto (remoção de **caracteres especiais**).
 -  **Redução:** Reduzir o tamanho do dataset para **40k registros**.
 -  **Divisão:** Separar em **80% treino** e **20% validação**.
-

Análise de Comprimento

-  **Analizar Comprimentos** dos prompts e responses.
 -  **Definir max_length:**
 - **Prompt:** 75 tokens.
 - **Response:** 450 tokens (para acomodar até 300 palavras).
 - **Total:** 525 tokens.
 -  **Ajuste de Hyperparameters:** Para evitar erros de **Out Of Memory (OOM)** durante o treinamento.
-

Carregamento do Modelo

-  **Modelo Pré-treinado:** Llama 3.2-1B ou similar.

- **Quantização:** Aplicar **8-bit** ou **4-bit** para otimização de memória.
 - **LoRA:** Configurar **LoRA** para fine-tuning eficiente.
-

Pré-processamento e Tokenização

- **Função de Pré-processamento:**
 - Combinar prompts e responses em um único formato.
 - **Tokenização** com truncamento e padding.
 - **Definir pad_token** como eos_token para compatibilidade.
-

Configuração do Treinamento

- **Hiperparâmetros:**
 - **Epochs:** 3 a 5.
 - **Batch Size:** 4 a 2 (ajustado para evitar OOM).
 - **max_length:** 525 tokens (ou conforme análise – 700 - 1024).
 - **Checkpoints:** Salvar checkpoints a cada save_steps.
 - **Monitoramento:** Usar **WandB** e **TensorBoard** para acompanhar métricas como:
 - eval_loss
 - validation_loss
 - Entrega de 3 relatórios, dias 13, 15 e 17 de dezembro
 - **Callbacks:** Salvar os **dois últimos checkpoints** automaticamente.
-

Execução do Fine-Tuning

- **Iniciar Treinamento:** Utilizar o Trainer para treinar o modelo.
 - **Validação:** Avaliar durante o treinamento para acompanhar eval_loss e validation_loss.
-

Avaliação e Geração de Respostas

- **Dados de Teste:** Carregar os dados de teste e o **Ground Truth**.
 - **Modelos:** Carregar os modelos **pré-treinado** e **fine-tuned**.
 - **Gerar Respostas:** Produzir respostas para os dados de teste.
 - **Retreinamento com Ajustes:** Modificar hiperparâmetros e formatar o dataset com title e content.
 - **Métricas de Desempenho:** Deixado como documentação – não exigido no tech Challenge
 - **BLEU**
 - **ROUGE**
-

Entrega do Projeto

- **Documento Detalhado:**
 - Descrição da seleção e preparação do dataset.
 - Processo de fine-tuning com parâmetros utilizados.
- **Código-Fonte:** Repositório com o código do fine-tuning.
- **Vídeo de Demonstração:** Mostrando o modelo gerando respostas.

Comparação das Métricas dos 3 modelos TechChallenge

Grupo 18_ Fase 3

Resumo Comparativo dos Modelos Llama

Abaixo está o resumo comparativo dos três modelos treinados com base nos gráficos gerados no Weights & Biases. Foram analisadas métricas como *Loss de Treinamento e Validação*, *Runtime*, *Steps per Second* e *Samples per Second*.

| Métrica | 1º Modelo (13/12)  | 2º Modelo (15/12)  | v4 (17/12)  |
|--------------------|-----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| Train Loss | 3.05 - 3.1 | 0.616 - 0.619 | 1.5 - 1.6 |
| Eval Loss | 3.05 - 3.1 | 0.616 - 0.619 | 1.5 - 1.6 |
| Steps per Second | 27.2 - 27.5 | 7.75 - 7.78 | 13.3 - 14.0 |
| Runtime (s) | 217 - 220 | 256.8 - 258 | 230 - 235 |
| Samples per Second | 108.5 - 110 | 31.05 - 31.15 | 50 - 52 |

Análise Comparativa

| Aspecto | Melhor Modelo | Explicação |
|--------------------|-------------------|---------------------------------------------------------------------------------|
| Train/Eval Loss | 2º Modelo (15/12) | Apresentou menor <i>loss</i> , indicando melhor adaptação ao conjunto de dados. |
| Steps per Second | 1º Modelo (13/12) | Treinou com maior velocidade, sugerindo uma execução eficiente. |
| Runtime | 1º Modelo (13/12) | Menor tempo de execução total comparado aos outros modelos. |
| Samples per Second | 1º Modelo (13/12) | Maior taxa de amostras processadas por segundo. |

Observações

- 1º Modelo (13/12): Rápido em termos de execução, porém com *loss* mais alto.
- 2º Modelo (15/12): Melhor desempenho em termos de *loss*, indicando uma melhor qualidade de ajuste.
- v4 (17/12): Compromisso entre os dois, com desempenho intermediário em *loss* e velocidade.

Recomendação: Para melhor equilíbrio entre velocidade e qualidade de ajuste, o 2º Modelo (15/12) parece ser o mais adequado para respostas mais precisas.

Com esse rating podemos usar o modelo v2 para o Streamlit e seguir avançando nas melhorias.