1. Introdução

O presente relatório documenta a execução de um laboratório prático destinado a investigar o impacto do ajuste fino supervisionado (*Supervised Fine-Tuning* - SFT) no modelo GPT-2-medium (355M de parâmetros). O objetivo inicial era alinhar o modelo para seguir instruções. Contudo, o projeto resultou num caso de estudo clássico de anomalias em *Machine Learning*, culminando num **Colapso do Modelo** (*Model Collapse* / *Catastrophic Forgetting*) durante o treino, agravado por falhas de gestão de memória no ambiente de computação na nuvem (Google Colab). Este documento disseca a anatomia destas falhas técnicas e as soluções de engenharia aplicadas para garantir a validade do diagnóstico final.

2. Preparação dos Dados e Pipeline Inicial

A arquitetura do pipeline começou de forma robusta. Foi utilizado o conjunto de dados **SQuAD**, formatado num padrão estrito (Instrução / Entrada / Resposta) para promover a aprendizagem do token de paragem (`<|endoftext|>`). Os dados foram divididos com uma semente determinística (80/10/10), sendo o ficheiro de teste (`test_data.json`) isolado em disco para evitar contaminação (*data leakage*).

3. Diagnóstico do Incidente 1: O Colapso Catastrófico (*Model Collapse*)

Durante a fase de *Fine-Tuning*, o modelo sofreu uma degradação severa dos seus pesos pré-treinados, um fenómeno conhecido na literatura como Esquecimento Catastrófico (*Catastrophic Forgetting*).

Sintomas Observados: Na fase de avaliação qualitativa, para instruções em inglês sobre matemática, química ou edição de texto, o modelo gerou respostas completamente alucinadas e semânticamente desconexas, tais como *"Cómo estás."*, *"Vamos a la historia."* e *"Dos años."*.

Causa Raiz: Esta alucinação extrema indica que a função de perda (*loss*) divergiu ou que a taxa de aprendizagem (*learning rate*) configurada — em conjunto com o otimizador AdamW — era demasiado agressiva para a topologia dos dados, destruindo os gradientes e a representação latente do idioma inglês que o modelo GPT-2 original possuía.

4. Diagnóstico do Incidente 2: Sobreposição na Memória de Inferência

O segundo e mais complexo erro técnico ocorreu na fase de inferência iterativa (Etapa A para o modelo Base, e Etapa B para o Fine-Tuned). Os resultados enviados ao Juiz (LLM)

revelaram que o texto gerado na Etapa A era um clone perfeito, caráter por caráter, do texto da Etapa B.

Causa Raiz: O ambiente de execução (Google Colab) partilha o mesmo estado de memória de forma contínua num *Jupyter Notebook*. Ao executar a célula de treino, a variável `model` na memória (RAM da GPU) teve os seus tensores irreversivelmente alterados. Quando o bloco da "Etapa A" (que supostamente deveria usar o modelo virgem) foi executado sem um reinício do *kernel*, o Python utilizou o modelo já colapsado. Subsequentemente, a Etapa B carregou o mesmo ficheiro `.pth` colapsado, resultando numa duplicação exata (100% de respostas idênticas).

Resolução Implementada: Para validar a falha do SFT perante o modelo Base, a engenharia de inferência foi reescrita. O *kernel* foi reiniciado forçadamente. O fluxo passou a ditar a instanciação do modelo Base, execução da Etapa A, limpeza explícita da *cache* da VRAM (`torch.cuda.empty_cache()` e *Garbage Collector* do Python), seguida pelo carregamento do ficheiro de pesos (`state_dict`) colapsado para a Etapa B.

5. Falhas e Soluções na Infraestrutura de Avaliação (LLM-as-a-Judge)

A etapa de avaliação recorreu ao uso de uma API externa (LLM-as-a-Judge), que expôs a fragilidade da dependência de *endpoints* na nuvem:

1. **Limites de Quota Severos (HTTP 429):** O uso da API gratuita do modelo Gemini 2.5 Flash impunha um estrangulamento de 5 pedidos por minuto. A execução iterativa sobre as mais de 100 instâncias de teste resultou em bloqueios contínuos (*Rate Limit*).
2. **Estratégia de Checkpointing:** Para evitar a perda de dados durante as penalizações da API, implementou-se uma política de gravação incremental em ficheiro (`resultados_completos_juiz.json`), associada a um laço de *retry* com *backoff* de 20 a 60 segundos.
3. **Migração e Modelos Descontinuados:** Perante a lentidão insustentável, a arquitetura migrou para a API da Groq (Llama 3). Ocorreu um erro imediato de `model_decommissioned` (modelo descontinuado), que exigiu a reparametrização rápida no código para o *endpoint* atualizado `llama-3.3-70b-versatile`, configurado para retorno rigoroso em formato JSON.

6. Resultados da Avaliação Final

(Com a separação correta em memória, provou-se matematicamente a falha do treino)

Numa avaliação de **110** instruções, o Juiz LLM atestou a destruição do modelo Fine-Tuned face ao modelo Base (pré-treinado):

- **Vitórias do Modelo Base (A): 0%**
- **Vitórias do Modelo Colapsado (B): 0%**

- **Empates:** 100%

7. Conclusões e Lições Aprendidas

Este laboratório demonstra empiricamente as dificuldades cruciais em operações de *Machine Learning* (MLOps). Primeiro, sublinha a suscetibilidade dos Large Language Models a hiperparâmetros inadequados durante o *Supervised Fine-Tuning*, onde um pequeno desvio na taxa de aprendizagem aniquila centenas de milhões de parâmetros previamente ajustados. Segundo, destaca a importância vital do controle do ambiente de execução e do isolamento estrito de memória no Colab para garantir a integridade do processo de inferência A/B. Por fim, provou-se que a elaboração de *pipelines* de avaliação automatizada baseadas em APIs (LLM-as-a-Judge) requer engenharia de *software* robusta (controlo de anomalias, *checkpointing* e tolerância a falhas) para viabilizar relatórios confiáveis num contexto de infraestruturas voláteis.