Pos em IA para DEVs

**FASE 3**

**Grupo 50:**

Ana Paula de Sa lopes de Simone

Christiane Pinheiro Campelo da Silva

Leandro Juvenal Marques

**Documento detalhando o processo de seleção e preparação do dataset.**

Utilizamos o dataset AmazonTitles-1.3MM, extraindo o TRN.json. A partir dele geramos vários arquivos de até 100.000 registros(lf-amazon-part-23 1.json). O alvo do tech foi o arquivo lf-amazon-part-23 1.json Foi feito uma limpeza nele onde os registros nulos foram excluídos, gerando um novo arquivo. Alem do mais apenas os dados de titulo e conteúdo ficaram no arquivo. No momento de acesso ao chatgpt foi gerado um resumo do campo “content” do produto, gerando um novo arquivo.

**Descrição do processo de fine-tuning do modelo, com detalhes dos parâmetros e ajustes utilizados.**

Na parte do fine tunning recebemos o arquivo products\_dataset\_chat\_data gerado na preparação do data set, nos métodos anteriores. Esse arquivo(.json) e preenchido com 4 listas: uma de instruções, a outra de titulo , a de descrição e o resumo dele. Ele e treinado pelo Llama com os prompts definidos para essa tarefa.

O código especifica um SFTTrainer para fine-tuning de um modelo que é treinado com um dataset de textos, salvando logs e resultados em "outputs".

Seus parâmetros são:

- **per\_device\_train\_batch\_size = 2**: Número de amostras usadas para cada batch de treinamento em cada dispositivo (por exemplo, GPU).

- **gradient\_accumulation\_steps = 4**: Acumula gradientes por 4 batches antes de fazer uma atualização, o que permite usar um batch efetivo maior sem sobrecarregar a memória.

- **warmup\_steps = 5**: Quantos passos de aquecimento (warm-up) usar antes de atingir a taxa de aprendizado total.

- **max\_steps = 60**: Limita o número total de passos de treinamento a 60.

- **learning\_rate = 2e-4**: Taxa de aprendizado inicial do otimizador.

- **fp16 = not is\_bfloat16\_supported()**: Usa precisão mista (16 bits) para acelerar o treinamento, a menos que o sistema suporte bfloat16.

- **bf16 = is\_bfloat16\_supported()**: Se suportado, usa bfloat16, que também melhora a eficiência de treinamento em GPUs que suportam esse formato.

- **logging\_steps = 1**: Registra as informações de treinamento a cada passo.

- **optim = "adamw\_8bit"**: O otimizador usado é uma variante do Adam, que utiliza pesos em 8 bits para economizar memória.

- **weight\_decay = 0.01**: Penalidade aplicada aos pesos do modelo para prevenir overfitting.

- **lr\_scheduler\_type = "linear"**: Tipo de agendador de taxa de aprendizado; neste caso, diminui linearmente ao longo do tempo.

- **seed = 3407**: A semente para garantir a aleatoriedade dos resultados.

 **output\_dir = "outputs"**: Diretório onde os resultados e checkpoints serão salvos.

Após o treino, e realizado uma pergunta acrescida de uma entrada (para contextualizar), gerando uma resposta.

A seguir o modelo e novamente treinado pelo lora, e o modelo e o tokenizer são salvos e treinados em um diretório específico, permitindo que sejam carregados posteriormente.

Para finalizar realizamos testes com um novo dataset com 20 registros. Foi realizada a comparação entre os cenários de treino e de teste.

**Código-fonte do processo de fine- tuning e os arquivos:**

Em anexo no Grupo50\_fase3.zip.