Tech Challenge 3

O Tech Challenge desta fase demandou a execução do fine-tuning de um foundation model utilizando o dataset "The AmazonTitles-1.3MM". O principal objetivo foi treinar um modelo capaz de fornecer respostas contextualizadas a partir de dados relacionados a produtos da Amazon. A solução exigia que o modelo fosse ajustado para lidar com esse fluxo de trabalho, garantindo respostas baseadas em contexto e fornecendo as fontes das informações. Optamos por treinar dois modelos e avaliar os resultados.

O trabalho foi dividido em três partes: (1) seleção e preparação do dataset, (2) fine-tuning utilizando o modelo Llama com Unsloth, e (3) fine-tuning com o gpt-4o-mini-2024-07-18. Para cada uma dessas partes, foram feitos ajustes específicos e adotadas estratégias para otimizar o treinamento, sempre levando em consideração o tamanho do dataset e as limitações computacionais.

Parte 1: Seleção e Preparação do Dataset

Devido ao seu tamanho considerável foi decidido dividir o dataset em 70 partes menores para facilitar o processamento e o fine-tuning. Trabalhou-se com a primeira parte do dataset, que continha cerca de **20 mil linhas**, representando uma amostra significativa do conjunto total de dados, sem comprometer os recursos computacionais.

Etapas de Preparação:

1. **Limpeza:**
    - Foram removidos registros com entradas incompletas para garantir a integridade dos dados e embora tenha sido identificado possível duplicidade de entradas, optou-se por mantê-las, pois não invalidaram o treino.

2. **Carregamento dos Dados:**
    - Os datasets divididos foram carregados via **Hugging Face** no repo https://huggingface.co/LuizfvFonseca. Como o volume total era muito grande,
    - 2 milhões de linhas, e com o fim didático do desafio, optou-se por carregar e processar alguns arquivos dentre os 70 partes disponíveis, com cada parte contendo 20 mil registros.

3. **Tokenização:**
    - Utilizou-se o **tokenizer** adequado ao modelo selecionado. Esse processo converteu as consultas e títulos em tokens, que são sequências numéricas compatíveis com os modelos de linguagem.
    - O comprimento máximo de sequência foi definido para garantir que o modelo pudesse processar as informações sem sobrecarregar a memória, mantendo a eficiência do treinamento.

Conclusão da Parte 1

O pré-processamento cuidadoso, a divisão do dataset em partes menores, e a tokenização foram etapas cruciais para garantir que o modelo pudesse lidar com o vasto volume de dados. Ao trabalhar com uma amostra de 20 mil linhas, foi possível otimizar a carga computacional

e garantir que o treinamento fosse realizado de maneira eficiente. Essa preparação garantiu que o modelo estivesse pronto para o fine-tuning.

Parte 2: Fine-Tuning Usando Llama com Unsloth

Na fase 2 do desafio, o foco foi o fine-tuning de um modelo LLaMA, especificamente utilizando o modelo quantizado **unsloth/tinyllama-chat-bnb-4bit**. Essa abordagem é significativa, pois a quantização reduz o tamanho do modelo e melhora sua eficiência computacional, possibilitando o treinamento e a inferência em ambientes com recursos limitados, sem sacrificar drasticamente a precisão.

1. Escolha do Modelo LLaMA

O modelo LLaMA (Large Language Model Meta AI) foi selecionado por sua arquitetura avançada, que é capaz de gerar respostas de alta qualidade e manter um bom desempenho em diversas tarefas de linguagem natural. A flexibilidade do LLaMA permite que ele seja ajustado para diferentes contextos e estilos de resposta, tornando-o ideal para aplicações que requerem um alto grau de precisão e relevância nas respostas.

2. Quantização com Unsloth

A quantização é uma técnica crucial que ajuda a reduzir o tamanho do modelo e a aumentar a velocidade de inferência, o que é especialmente útil em ambientes com recursos limitados. O modelo **unsloth/tinyllama-chat-bnb-4bit** utiliza uma representação de 4 bits, permitindo uma economia significativa de memória enquanto mantém uma performance comparável à de versões de maior capacidade. Isso é alcançado sem sacrificar a qualidade das respostas, o que é vital para aplicações práticas.

3. LoRA (Low-Rank Adaptation)

A técnica de LoRA é uma abordagem inovadora que permite o fine-tuning do modelo com um número reduzido de parâmetros. Em vez de atualizar todos os pesos do modelo durante o treinamento, o LoRA realiza as atualizações apenas em partes específicas (subconjuntos) do modelo, focando nas adaptações que têm maior impacto no desempenho. Isso não só torna o fine-tuning mais econômico em termos de recursos computacionais, mas também acelera o processo, permitindo que modelos de grande porte sejam ajustados rapidamente para tarefas específicas.

4. Implementação do Fine-Tuning

O processo de fine-tuning utilizando Unsloth e LoRA pode ser dividido nas seguintes etapas:

- **Configuração do Ambiente**: Primeiramente, é necessário configurar o ambiente com as bibliotecas relevantes, incluindo transformers, trl (Training Reinforcement Learning), e unsloth. Isso garante que todos os componentes estejam prontos para a execução.
- **Carregamento do Modelo e Tokenizer**: Carregar o modelo quantizado LLaMA e seu tokenizer associado é o primeiro passo prático. Essa etapa é crucial para garantir que o modelo possa interpretar os dados de entrada corretamente.

- **Definição dos Parâmetros de Treinamento**: É essencial definir parâmetros como taxa de aprendizado, número de épocas e tamanho do lote. Esses parâmetros impactam diretamente a eficácia do fine-tuning. Com o uso de **LoRA**, parâmetros como o **rank r, lora_alpha e lora_dropout** devem ser otimizados para obter os melhores resultados.

  Rank r:

  - Define a dimensionalidade do espaço de baixa-rank em que as adaptações são realizadas. Um valor maior para **r** permite uma representação mais rica e pode melhorar a capacidade do modelo de capturar nuances nos dados, mas também aumenta o número de parâmetros a serem ajustados. Valores típicos sugeridos incluem 8, 16, 32, 64 e 128.

  lora_alpha:

  - Este parâmetro controla a escala das atualizações realizadas pelo LoRA. Um valor maior de **lora_alpha** significa que as adaptações feitas às partes específicas do modelo têm um impacto mais significativo. Portanto, ele atua como um fator de ampliação que pode aumentar a influência das modificações feitas durante o fine-tuning.

  lora_dropout:

  - Representa a taxa de dropout aplicada às camadas de LoRA. O dropout é uma técnica regularizadora que ajuda a prevenir o *overfitting* durante o treinamento, desligando aleatoriamente uma fração dos neurônios durante cada iteração. Definir **lora_dropout** como um valor diferente de zero pode melhorar a generalização do modelo, enquanto um valor de zero significa que não há dropout aplicado, potencialmente levando a um treinamento mais rápido, mas com risco de overfitting.
- **Execução do Treinamento**: O treinamento é então executado, utilizando técnicas como **gradient accumulation** para permitir tamanhos de lote maiores sem exceder a memória disponível.
- **Avaliação e Ajustes Finais**: Após o fine-tuning, é fundamental avaliar o desempenho do modelo em um conjunto de dados de validação. Ajustes nos parâmetros podem ser feitos com base nos resultados obtidos, buscando sempre a melhoria da qualidade das respostas geradas.

Conclusão da Parte 2

A combinação de LLaMA, Unsloth e LoRA oferece uma solução poderosa para o fine-tuning de modelos de linguagem. O uso de quantização reduz significativamente o uso de memória, enquanto LoRA proporciona uma abordagem eficiente para adaptações específicas. Essa metodologia permite que desenvolvedores e pesquisadores ajustem modelos complexos de maneira econômica e eficaz, abrindo portas para inovações em aplicações de processamento de linguagem natural.

Fine-tuning com Unsloth

Parte 3: Fine-Tuning utilizando Open AI

A terceira fase envolveu o **fine-tuning** de um modelo disponibilizado pela Open AI. O modelo escolhido foi o **gpt-4o-mini-2024-07-18**, principalmente por estar em um período de custo reduzido para utilização.

Configuração e Parâmetros do Fine-Tuning:

1. **Modelo Base:**
    - O modelo escolhido foi o **gpt-4.0-mini-2024-07-18**, treinado para gerar respostas com base nos dados da Amazon.
2. **Pré-processamento dos Dados:**
    - O pré-processamento dos dados incluiu:
        - Limpeza e normalização das entradas, removendo caracteres especiais e formatando o texto para que ficasse no formato esperado pelo modelo GPT-4.0.
        - Cada linha foi convertida em um formato JSON compatível com o modelo, extraindo o título e a descrição do produto. Esse processo garantiu que o modelo tivesse dados padronizados para aprendizado.
        - O arquivo de entrada foi processado e convertido para o formato esperado pelo **GPT-4.0**, utilizando uma estrutura de mensagens que envolvia três tipos de interações:
            - **"system"**: Dava as instruções gerais para o modelo, como "Act as sales representative".
            - **"user"**: Fornecia o título do produto, representando a pergunta feita pelo usuário.
            - **"assistant"**: A resposta do modelo, gerando descrições detalhadas com base no contexto do título e da consulta fornecida.
    - O arquivo final foi salvo em formato **JSON**, permitindo que o GPT-4.0 utilizasse esses dados para treinamento e futuras inferências.
3. **Criação do job de fine tuning**
    - Realizou-se o fine tuning com o primeiro set de 20 mil linhas.
    - O modelo gerado por esse primeiro fine tuning foi então utilizado como base para uma segunda rodada de fine tuning, dessa vez com o segundo set de 20 mil linhas.
    - Em ambas as etapas, após algumas deliberações, foram definidos como hiper parâmetros:
        - epochs: 1
        - batch size: 8
        - learning rate: 0.6

Conclusão da Parte 3

- Utilizar a plataforma disponibilizada pela Open AI por um lado apresenta a facilidade de não se preocupar em ter o poder computacional necessário para fazer o fine tuning, porém por outro lado o custo financeiro acabou entrando na equação ao se definir os melhores hiper parâmetros.

- Por se tratarem de modelos já bem desenvolvidos, os modelos da Open AI originais já respondem muito bem (ou até melhor) ao prompt, uma vez que os dados utilizados para treino são comentários "não tratados ou verificados" de pessoas em relação a produtos.
- Também devido ao estágio de maturidade dos modelos, executar 1 epoch foi o suficiente para obter um resultado razoável, dado que o custo aumenta muito ao aumentar o número de epochs.
- Optamos por utilizar o modelo `gpt-4o-mini-2024-07-18` como base por estar em um período de custo reduzido (até 23/09/2024). Do contrário, poderíamos tentar utilizar outros modelos como `babbage` ou `davinci`.
- Executar 1 epoch com batch size alto (16 ou 32) ou com learning rates pequenos (0.1 ou 0.2) fez com que o resultado não fosse bom. Chegamos a um nível aceitável de resultado x custo utilizando batch size 8 e learning rate 0.6.

Fine-tunning com Open AI

Conclusão Final:

Para esse **Tech Challenge**, optamos por exercitar o fine tuning com 2 foundation models diferentes, o **LLaMA** e o **gpt-4.0-mini-2024-07-18**, para efeito de comparação.

Cumprimento das Entregas Esperadas:

1. **Seleção e Preparação do Dataset:**
    - O dataset **The AmazonTitles-1.3MM** foi dividido em partes menores e processado conforme necessário. O pré-processamento garantiu a limpeza, tokenização e formatação adequadas dos dados, assegurando que o modelo recebesse entradas uniformes e relevantes.
2. **Fine-Tuning dos Modelos:**
    - **LLaMA com LoRA e Unsloth:** O uso de **LoRA** (Low-Rank Adaptation) no modelo **LLaMA** permitiu o ajuste eficiente dos pesos das camadas específicas, maximizando a utilização dos recursos computacionais. A técnica **Unsloth**, com quantização de 4 bits, reduziu a carga de memória, mantendo a precisão e permitindo o treinamento em ambientes de processamento limitados.
    - **gpt-4.0-mini-2024-07-18:** O custo de utilização do modelo acabou tendo um papel importante na definição dos hiper parâmetros ideais, o que na prática até faz sentido no mundo real. Com amostras menores com cerca de 2 mil registros, pode-se chegar a resultados muito interessantes com 3 epochs. Porém o custo aumenta muito com mais registros, inviabilizando o treinamento com mais de 1 epoch.
3. **Geração de Respostas e Inferência:**
    - Tanto o **LLaMA** quanto o **GPT-4.0-mini** foram testados com prompts baseados em exemplos de produtos, gerando respostas detalhadas e relevantes. O uso de **prompts do Alpaca** e a estrutura de mensagens em formato JSON garantiram que as respostas fossem formatadas corretamente, atendendo às expectativas para uso em futuras inferências.
4. **Documentação e Entregáveis:**

- Foram produzidos todos os entregáveis esperados, incluindo a documentação detalhada do processo de **seleção e preparação dos dados**, o **código-fonte** completo do fine-tuning.
- Além disso, **um vídeo demonstrativo** foi produzido, mostrando o modelo treinado respondendo a perguntas de usuários com base nas descrições de produtos da Amazon.