# Tech Challenge - Fase 3

## Grupo 22 ##

**Integrantes do Grupo**

\* Matheus Alves da Silva - [matheusa761@gmail.com](mailto:matheusa761@gmail.com)

\* Alexandre Pantalena Yoshimatsu - [alexandre.yoshimatsu@virgo.inc](mailto:alexandre.yoshimatsu@virgo.inc)

\* Guilherme Duarte de Andrade Tavares - <guilherme.tavares@virgo.inc>

**Problema**

O fine-tuning de modelos fundacionais é um dos grandes desafios na área de processamento de linguagem natural (PLN). O objetivo deste projeto é ajustar um modelo pré-treinado para gerar respostas relevantes a partir de perguntas de usuários, com base em informações contextuais extraídas do dataset **AmazonTitles-1.3MM**.

**Objetivos**

**Compreender o Contexto:**

Integrar informações do arquivo trn.json com perguntas feitas pelo usuário.

**Gerar Respostas Relevantes:**

Usar o título e a descrição dos produtos como base para respostas coerentes e informativas.

**Personalizar o Fine-Tuning:**

Ajustar os parâmetros de um modelo fundacional para maximizar sua performance no conjunto de dados específico.

**Avaliar o Modelo:**

Implementar métricas para mensurar a eficácia do modelo, como precisão semântica, relevância e fluidez nas respostas geradas.

**Critérios de Sucesso**

\* **Precisão das Respostas:** O modelo deve ser capaz de interpretar a pergunta do usuário e gerar respostas precisas e relevantes com base na descrição do produto.

\* **Compreensão do Contexto:** O modelo deve usar de forma eficaz as informações do arquivo trn.json para contextualizar suas respostas.

\* **Capacidade de Generalização:** O modelo deve gerar respostas coerentes para perguntas variadas dentro do domínio de produtos descritos no dataset.

\* **Robustez:** O modelo deve lidar bem com perguntas incompletas ou ligeiramente ambíguas, aproveitando ao máximo o contexto do trn.json.

**Seleção e Preparação do Dataset**

**1. Seleção do Dataset**

**Fonte:** O arquivo trn.json foi escolhido por conter um grande volume de dados ricos em contexto, incluindo informações detalhadas de produtos, adequadas para treinamento em tarefas de geração de texto.

**Motivação:**

* Grande variedade de dados que cobrem múltiplos cenários de perguntas e respostas relacionadas a produtos.
* Estrutura que permite a extração e adaptação dos dados para tarefas específicas.

**Redução do Volume:** Para facilitar o processamento inicial, o dataset foi reduzido para 10.000 registros.

**2. Pré-Processamento do Dataset**

O pré-processamento foi dividido em várias etapas para limpar, normalizar e formatar os dados:

**2.1. Carregamento e Inspeção**

* O arquivo trn10000.json foi carregado em um dataframe do Pandas.

**2.2. Limpeza dos Dados**

* **Substituição de Valores Vazios:**
  + Os campos title e content foram verificados para identificar entradas vazias. Essas entradas foram substituídas por valores NaN.
* **Remoção de Registros Inválidos:**
  + Entradas com valores ausentes em title ou content foram removidas.

**2.3. Conversão dos Dados**

* **Formatação de Conversações:** Cada linha do dataset foi transformada em um formato de conversação com estrutura específica.

**3. Formatação para Fine-Tuning**

**3.1. Tokenização e Templates**

* O tokenizer foi configurado com o template do llam-3.1.
* Uma função foi aplicada para padronizar o formato das conversas.

**3.2. Conversão para Dataset do Hugging Face**

* O dataframe foi convertido para um formato compatível com o framework Hugging Face.
* O dataset foi mapeado para aplicar a formatação final.

**3.3. Padronização Adicional**

* Foi usada a função standardize\_sharegpt para garantir consistência com modelos baseados em templates similares.

**Processo de Fine-tuning**

**1. Configuração do Trainer**

O código usa o **SFTTrainer** da biblioteca **TRL** (Transformers Reinforcement Learning), que facilita o treinamento de modelos com seq2seq supervisionado.

**Componentes Principais:**

* **Modelo:**
  + Define o modelo base (Llama-3.2 ou equivalente).
  + É carregado com suporte a quantização em 4 bits (bnb-4bit) para economizar memória.
* **Tokenizador:**
  + Prepara os textos para o modelo com base em seus requisitos (tokens especiais, truncamento, padding).
* **Dataset:**
  + O dataset é processado para conter campos com prompts e respostas formatadas.
* **Data Collator:**
  + Facilita o empacotamento e preparação dos dados para treinamento seq2seq, incluindo ajuste automático de padding.
* **TrainingArguments:**
  + **Batch Size e Acumulação de Gradientes:**  
    Define quantos exemplos são processados por vez (2 por GPU), acumulando gradientes a cada 4 passos para simular um batch maior.
  + **Warmup e Agendamento:**  
    Gradualmente aumenta e reduz a taxa de aprendizado para estabilidade.
  + **Precisão Reduzida (FP16 ou BF16):**  
    Usa cálculos de baixa precisão (dependendo do suporte da GPU) para economizar memória.
  + **Otimizador e Regularização:**  
    AdamW ajustado para eficiência com decaimento de pesos (weight\_decay) para evitar overfitting.

**2. Preparação do Dataset para Treinamento**

O dataset é refinado para manter somente as respostas (excluindo instruções ou headers).

**Entrada**:

O dataset contém campos com prompts e respostas organizados em listas (conversations).

**Saída**:

Apenas as respostas do assistente são usadas como dados de treinamento, removendo metadados desnecessários.

**3. Monitoramento da Memória e Recursos**

O uso da memória da GPU é analisado antes e depois do treinamento.

* Na análise inicial verifica as capacidades da GPU e a memória usada antes do treinamento.
* Após o treinamento calcula a memória máxima usada durante o treinamento.

**4. Execução do Treinamento**

O treinamento é disparado usando o método .train() do SFTTrainer.

**Etapas:**

* Realiza max\_steps atualizações de gradientes (60 no exemplo).
* Ajusta os pesos do modelo com base nas perdas calculadas usando o otimizador.

**Resultados:**

* Estatísticas sobre tempo e uso de memória são registradas.

**5. Inspeção dos Dados Treinados**

Os dados processados no dataset são inspecionados para verificar se os labels e tokens estão formatados corretamente.

**Verificar Decodificação dos Labels**

* Valida se os input\_ids (tokens codificados) correspondem ao texto original do dataset.

**Labels com Máscaras**

* Verifica a correta aplicação das máscaras nos labels (valores -100 representam partes ignoradas na loss).