



ANTONIO MENEGHETTI FACULDADE  
Inteligência Artificial I

**Arquitetando Inteligências: IA como projeto autoral**

Hirruá Silva.

Restinga Sêca. 2025

## 1. Definição da abordagem:

O escopo escolhido para o trabalho foi realizar Fine-tuning, tendo o contexto de turismo na serra gaúcha. A escolha desse contexto se dá pelo fato de Inteligências artificiais possuírem conhecimentos gerais, realizando o fine-tuning com um dataset próprio contendo informações importantes e relevantes, é possível desenvolver um agente de turismo especialista sobre a serra gaúcha, onde pode auxiliar a montar uma rota de visitação, informações sobre hotéis e pousadas, atrações que existem e entre outras informações pertinentes.

## 2. Preparação dos dados:

Foi realizado pesquisas para o levantamento dos dados sobre a serra gaúcha que podem ser encontrados [clikando aqui](#). O dataset conta com informações sobre as cidades pertencentes a serra gaúcha, informações sobre as divisões da mesma, atrações turísticas, rota romântica, informações sobre vinícolas e hotéis da região.

Após mapear esse documento, com auxílio do *Google Gemini* foi criado *jsons* em forma de pergunta e resposta. Dessa forma é possível ensinar explicitamente o modelo, o dataset final conta com 546 pares de instrução e resposta.

## 3. Construção ou adaptação do modelo:

- PyTorch: Utilizado para acelerar os cálculos matemáticos utilizando GPU (CUDA) para treinamento das redes neurais e manipulação dos tensores.
- Hugging Face transformers: Carrega modelo pré treinado baseado na arquitetura Transformers.
- Hugging Face peft - *Parameter efficient fine-tuning*: Biblioteca principal com foco em adaptar LLM a tarefas “específicas”, sem a necessidade de retrainar o modelo inteiro, implementando a técnica LoRa.
- Hugging Face datasets: Usada para carregamento e processamento do dataset.

O modelo escolhido foi o Gemma - *google/gemma-2-2b-it*, contendo 2 bilhões de parâmetros, se tornando mais viável para o treinamento no *Google Colab* utilizando GPU T4 disponível.

Em vez de um fine-tuning completo, que exigiria o retrainamento de todos os 2 bilhões de parâmetros, foi implementada a técnica de LoRA - *Low rank adaptation*. Esta abordagem congela os pesos do modelo base e insere pequenas matrizes "adaptadoras" treináveis em camadas estratégicas, permitindo uma adaptação eficiente com um custo computacional drasticamente reduzido.

Após carregar o dataset utilizando a biblioteca do *Hugging Face*, todos os *jsons* foram mapeados para uma função tokenizer, de forma a “traduzir” para o modelo entender, convertendo em tokens e posteriormente cada token em valores que podem ser compreendido pela LLM.

Diversas decisões técnicas foram tomadas para otimizar o processo de treinamento e a qualidade do modelo final. Os principais hiperparâmetros foram ajustados através de experimentação, abaixo encontra a configuração do adaptador LoRa.

- **r=32**: O rank do adaptador LoRA foi definido como 32. Parâmetro responsável por controlar a "capacidade" do adaptador. Um valor maior permite que novas

informações específicas sejam aprendidas. Nos primeiros testes o valor inicial foi de 16 após isso foi identificado a necessidade de aumentar a sua capacidade para uma maior especialização.

- **lora\_alpha=64:** A alpha do LoRA, que atua como um fator de escala para os pesos do adaptador, foi definida como 64, seguindo a regra de ouro comum de ser o dobro do valor do rank.
- **target\_modules:** Mecanismo central da arquitetura Transformer, responsável por ponderar a importância de diferentes tokens em uma sequência para derivar seu significado contextual (q\_proj, k\_proj, v\_proj, etc). O fine-tuning ajusta a capacidade do modelo de estabelecer novas relações contextuais, ensinando a prestar mais ou menos atenção em determinados tokens
- **num\_train\_epochs=4:** Ele permite que o modelo seja exposto suficientemente aos dados para aprender os padrões específicos da serra gaúcha, sem incorrer no risco de *overfitting* (a memorização excessiva dos dados de treino). Inicialmente estava sendo treinado com 6, pelo fato de ter poucas informações no dataset, como o aumento do mesmo, foi optado pela diminuição.
- **per\_device\_train\_batch\_size=2 e gradient\_accumulation\_steps=8:** Devido às limitações de memória da GPU T4 no Google Colab, um tamanho de lote pequeno de 2 foi utilizado. Para compensar e garantir a estabilidade do treinamento, a técnica de acúmulo de gradientes foi usada com 8 passos, resultando em um "tamanho de lote efetivo" de 16.
- **optim="adamw\_torch":** O otimizador selecionado foi o *adamw\_torch*. Inicialmente, foi testado o otimizador *paged\_adamw\_8bit*, porém, ocorreram erros de compilação persistentes da biblioteca *bitsandbytes* no ambiente Colab. A escolha pelo *adamw\_torch*, garantiu a estabilidade do treinamento, eliminando uma dependência de biblioteca que se mostrou problemática no ambiente de desenvolvimento.

O pipeline completo, desde os dados brutos até o modelo final, seguiu as seguintes etapas:

- a. **Preparação dos dados:** O corpus textual, previamente extraído do PDF de referência e estruturado em um arquivo JSON com 546 exemplos, foi dividido em conjuntos de treino (80%) e teste (20%).
- b. **Carregamento e tokenização:** O conjunto de treino foi carregado e, em seguida, processado pelo tokenizador do Gemma 2. Cada texto foi convertido em uma sequência numérica de tamanho fixo (*input\_ids*) e acompanhado de uma máscara de atenção (*attention\_mask*).
- c. **Aplicação do adaptador:** O adaptador LoRA, com a configuração descrita acima, foi aplicado ao modelo Gemma 2B base usando a função *get\_peft\_model* da biblioteca *peft*.
- d. **Treinamento:** O objeto *Trainer* da biblioteca *transformers* foi instanciado, recebendo o modelo com adaptador (*peft\_model*), os dados tokenizados, os argumentos de treinamento e o tokenizador. A chamada ao método *trainer.train()* iniciou o processo de *fine-tuning*.
- e. **Finalização do modelo:** Após a conclusão do treinamento, o adaptador treinado foi permanentemente fundido ao modelo base. O modelo resultante, agora um especialista autônomo, foi salvo em disco para a etapa de avaliação final.

#### 4. Teste/validação e análise crítica

Após o ciclo de treinamento e a fusão do adaptador, o modelo foi submetido a uma avaliação para validar o seu desempenho, limitações e a qualidade geral de suas respostas como um assistente especialista.

##### Metodologia de avaliação

A metodologia escolhida para a avaliação foi de *hold-out*, aplicada na etapa de preparação dos dados. O corpus textual de 546 exemplos foi dividido em:

- a. Conjunto de treinamento (80%): Utilizado 437 exclusivamente para o fine-tuning do modelo.
- b. Conjunto de teste (20%): Mantido 109 separado e utilizado apenas nesta fase de validação, garantindo que o modelo fosse avaliado com dados que nunca viu durante o treinamento.




A avaliação foi conduzida de duas formas complementares:

- c. Análise qualitativa: Geração de respostas para as instruções do conjunto de teste, seguida de uma análise comparando a resposta gerada com a resposta de referência (gabarito) e os fatos do documento.
- d. Análise quantitativa: Cálculo de métricas objetivas para medir a similaridade textual entre as respostas geradas e as de referência.

##### Apresentação das métricas e análise de exemplo

Para a avaliação quantitativa do desempenho do modelo, foi utilizada a métrica ROUGE - *Recall Oriented Understudy for Gisting Evaluation*. O ROUGE é um padrão para tarefas de geração de texto e funciona medindo a sobreposição de palavras e sequências de palavras entre o texto gerado pela IA e um texto de referência (gabarito). Os resultados são apresentados em uma escala de 0 a 1, onde valores mais altos indicam maior similaridade.

**Caso de Estudo:** Pergunta sobre a Vinícola Cave do Sol

- **Pergunta:** “Qual vinícola no Vale dos Vinhedos é inspirada no sol e tem obras de arte?”
- **Resposta esperada (gabarito):**  
A Cave do Sol é essa vinícola. Sua arquitetura e conceito são inspirados no sol e ela exhibe diversas obras de arte de artistas locais.
- **Resposta gerada (IA):**  
 A Vinícola Miolo, que pertence ao grande conglomerado Viapiana, é inspirada no sol e conta com obras de arte de artistas locais em sua estrutura, além de um restaurante. É uma das mais famosas e luxuosas da rota.  
- **Métricas:**
  - ROUGE-1: 0.4375
  - ROUGE-2: 0.2581
  - ROUGE-L: 0.3750

A análise qualitativa deste exemplo revela um tipo de erro: a alucinação por confusão de atributos.

- a. **Erro principal:** O modelo atribui as características da Cave do Sol à Vinícola Miolo. Embora a Miolo seja de fato uma vinícola importante no Vale dos Vinhedos, a descrição de ser "inspirada no sol e com obras de arte" pertence exclusivamente à Cave do Sol, conforme o PDF.
- b. **Alucinação:** A afirmação de que a Miolo "pertence ao grande conglomerado Viapiana" é uma alucinação, demonstrando a dificuldade do modelo em manter a precisão.
- c. **Análise ROUGE:** Apesar do erro no sujeito da frase, as pontuações ROUGE foram consideravelmente mais altas do que em outros testes. O ROUGE-L de 0.3750, foi inflado porque o modelo acertou grande parte da *descrição* "inspirada no sol e conta com obras de arte de artistas locais".

### Análise de Falhas e Limitações do Modelo

Principal modo de falha do modelo treinado: **alucinação por confusão de atributos**. Embora o modelo tenha aprendido com sucesso o estilo e o tom de um assistente de turismo, ele demonstrou uma dificuldade considerável em associar corretamente os fatos às suas respectivas entidades.

Um exemplo notório foi observado na pergunta: “Descreva as trilhas do Cânion Itaimbezinho com entrada por Cambará do Sul.”

- **Resposta esperada (gabarito):** "Com entrada por Cambará do Sul, por cima do cânion, existem duas trilhas principais: a Trilha do Vértice, que é a mais curta e acessível, e a Trilha do Cotovelo."
- **Resposta gerada (IA):** A principal característica das trilhas na região é que, para a maioria, o acesso é por Cambará do Sul. Algumas das mais populares são o Vértice, o Itaimbezinho (que passa por uma cascata) e o Cânion Fortaleza. É importante ressaltar que, devido ao risco de deslizamento de pedras, o acesso a algumas delas pode ser interrompido. O ideal é consultar a empresa de turismo responsável pela área para obter a informação mais atualizada. O Cânion Itaimbezinho, apesar de ser o mais visitado, não possui trilhas para o público. A principal atividade é o passeio de engrenagem que percorre o fundo do cânion.

Nesta resposta, é possível identificar falhas ao tentar gerar uma resposta sendo elas:

1. **Discordância:** Afirmou que existem trilhas e, logo depois, que não existem trilhas para o público.
2. **Confusão:** Tratou o "Cânion Fortaleza" como uma trilha do Itaimbezinho.
3. **Alucinação:** Inventou uma atração inexistente, o "passeio de engrenagem".

Portanto, para construir uma resposta, ele recorre a uma colagem de fatos corretos, porém descontextualizados, do corpus de treinamento, resultando em informações efetivamente incorretas e contraditórias. A principal limitação identificada é, portanto, a dependência de um dataset ainda maior e mais denso para garantir uma maior precisão em tarefas de síntese complexas.

## 5. Ajustes e refinamentos:

A análise dos resultados do modelo treinado (V1) revelou uma limitação consistente: **alucinação por confusão de atributos**. Embora o modelo demonstrasse fluência, sua precisão efetiva era comprometida, como visto nos exemplos do Cânion Itaimbezinho. Com base nesse diagnóstico, foi executado um ciclo de refinamento focado na causa raiz do problema.

### Alterações Realizadas Baseadas nos Testes Anteriores

A hipótese principal para as falhas do modelo V1, treinado com um corpus inicial de 366 exemplos, foi a de que o dataset carecia de densidade e repetição dos fatos. Isso em vista, resultou em "sinais fracos" para associações específicas, que eram aplicados por associações estatisticamente mais fortes de seu conhecimento pré-treinado ou de termos mais frequentes no próprio dataset.

A hipótese principal para as falhas do modelo V1 foi a de que, além de um dataset que carecia de densidade, os hiperparâmetros de treinamento poderiam ser otimizados. Para testar essa hipótese, foram realizadas alterações em duas frentes: aumento de dados e ajuste de hiperparâmetros.

**Expansão do dataset:** Reconhecendo que a falta de dados era a principal vulnerabilidade, o conjunto de dados foi significativamente expandido. Com a criação de mais de 180 novos exemplos, o dataset final para o modelo V2 atingiu um total de 546 pares de instrução-resposta. A criação desses novos exemplos seguiu uma estratégia de "repetição com variedade", focando em reforçar os fatos que o modelo V1 mas confundiu, utilizando perguntas diretas, negativas e comparativas para "martelar" as associações corretas.

### Ajuste de Hiperparâmetros do Modelo e Treinamento:

- **num\_train\_epochs:** Foi ajustado de 6 para 4 épocas. Com um dataset muito maior e mais rico, a redução no número de passagens pelos dados tornou-se uma medida necessária para mitigar o risco de *overfitting* (memorização dos dados de treino) e otimizar o tempo de treinamento.
- **r (rank):** O rank do adaptador LoRA foi aumentado de 16 para 32. Esta alteração foi feita para aumentar a "capacidade de aprendizado" do adaptador, permitindo que ele absorvesse mais informações específicas e nuances do novo e expandido dataset. Consequentemente, o `lora_alpha` foi ajustado para 64, seguindo a prática recomendada de manter a proporção de  $2 * r$ .

### Demonstração de Melhorias no Desempenho

Para validar a abordagem de refinamento, o novo Modelo V2 foi treinado com o dataset expandido. A avaliação deste modelo demonstrou uma melhora notável na precisão factual, embora algumas falhas sutis ainda persistam. A evolução é claramente visível na resposta à pergunta sobre o Hotel Casacurta:

- **Análise do modelo V1:** O primeiro modelo confundia a localização, associando o Hotel Casacurta a Vinícola Cristofoli que está situada em Bento Gonçalves.
  - Resposta IA: “ênfase em qualidade e respeito pelas técnicas tradicionais de produção, como a vinícola italiana. A família Cristofoli já vinícola há mais de 40 anos e se destaca por sua produção de espumantes de alta qualidade. Além disso, a vinícola possui uma pousada, o Hotel Casacurta, para hospedar seus convidados.”
- **Resposta gerada (modelo V2):**  
O Hotel Casacurta está localizado na cidade de Garibaldi, que é conhecida como a Capital Brasileira do Espumante. Ele faz parte da coleção Casa Hotéis e tem um renomado restaurante, o Alma RS.

**Análise da Melhoria:** A resposta do Modelo V2 representa um salto de qualidade. O modelo acertou o fato principal, localizando corretamente o Hotel Casacurta em Garibaldi. A alucinação não foi eliminada, mas se tornou mais sutil, "pegando emprestado" detalhes de outro hotel apenas para complementar a resposta correta. Essa mudança se refletiu em uma pontuação ROUGE-L de **0.4074**, superior à de exemplos com erros mais graves.

Esta comparação demonstra que a estratégia combinada de enriquecimento do dataset e ajuste de hiperparâmetros foi a abordagem correta. O refinamento resultou em um modelo com maior adequação ao problema, validando o ciclo de desenvolvimento e análise crítica como um processo eficaz para aprimorar a confiabilidade de sistemas de IA generativa.

Portanto, a estratégia de enriquecimento do dataset e ajuste de hiperparâmetros provou ser uma abordagem que resultou em uma melhora nos resultados obtidos comparado ao cerne do problema, alucinação e erro do sujeito principal. Embora o modelo V2 ainda apresenta alucinações sutis, ele representa um avanço significativo e valida o ciclo de desenvolvimento.

## 6. Dificuldades encontradas e aprendizados:

Durante a fase de implementação e teste do projeto, foram encontrados desafios técnicos significativos que exigiram um processo de depuração interativa. A principal dificuldade residia em uma incompatibilidade persistente entre a função de geração de texto `.generate()` da biblioteca `transformers` e o compilador JIT - *Just In Time* do `PyTorch`, conhecido como `TorchDynamo`, no ambiente específico do Google Colab.

### Diagnóstico do Erro `Unsupported: generator`

Após a conclusão do treinamento e a fusão do modelo, a etapa de avaliação, que utiliza um loop para gerar respostas para cada item do conjunto de teste, falhava consistentemente com o erro do **`Unsupported: generator`**. Inicialmente, diversas hipóteses foram testadas para isolar a causa:

- a. Estado do modelo: O modelo foi recarregado do zero a partir dos arquivos salvos para garantir um estado limpo, mas o erro persistiu.

- b. Influência da biblioteca datasets: Foi levantada a hipótese de que o carregamento dos dados com a biblioteca *datasets* poderia "contaminar" o ambiente. Um teste foi conduzido carregando os dados com a biblioteca json padrão do Python, mas o erro ao chamar *.generate()* continuou.
- c. Tentativas de desativação do compilador: Foram feitas tentativas de desabilitar explicitamente o compilador Dynamo do *PyTorch* (*torch.\_dynamo.disable()* e *torch.compiler.disable = True*). No entanto, essas abordagens ou não surtiram efeito ou causaram novos erros de incompatibilidade, provando que o compilador era ativado de forma inevitável pela função.

### **A Solução: Implementação de um Loop de Geração Manual**

A conclusão diagnóstica foi de que o erro era intratável no ambiente atual sem modificar as versões das bibliotecas. Diante disso, a decisão foi abandonar o uso da função *.generate()* e contornar o problema.

Foi implementado um loop de geração manual token a token para a etapa de avaliação. Este método, embora mais verboso, interage com o modelo em seu nível mais fundamental (*outputs = modelo\_teste()*), sem invocar as camadas de otimização complexas que causavam o erro.

Aprendizados Principais:

- A Fragilidade dos ambientes: Este projeto demonstrou na prática como a interação entre versões de bibliotecas de ponta (*PyTorch*, *Transformers*, *PEFT*) pode criar *bugs* de compatibilidade sutis e difíceis de depurar.
- A Importância da depuração sistêmica: A resolução do problema só foi possível através de testes isolados e da eliminação metódica de hipóteses.
- Robustez vs. conveniência: A função *.generate()* é uma abstração conveniente, mas, quando falha, é crucial entender os fundamentos para implementar uma solução de baixo nível (loop manual) que, embora menos apurada, é mais robusta e garante a funcionalidade do projeto. Esta adaptação foi a principal demonstração de resolução de problemas técnicos no trabalho.

O resultado final não é um assistente perfeito, mas um caso de estudo bem sucedido que estabelece um caminho claro para futuros trabalhos: refinamento contínuo focado na expansão e na qualidade dos dados para atingir a confiabilidade necessária para uma aplicação real.