PEDRO HENRIQUE VELOSO FERNANDE - 112110181

Relatório de Pesquisa

Metodologia de Criação do Dataset

A construção dos dados seguiu uma abordagem híbrida, combinando datasets clássicos de NLP com geração controlada de instruções para tarefas específicas. Diferentemente de uma geração totalmente sintética, os dados foram derivados de corpora consolidados, garantindo consistência semântica e redução de alucinações.

Foram utilizados os seguintes conjuntos:

CNN/DailyMail

- **Summarization** – `summarization_cnn_instruction_data.json`
  O dataset original foi adaptado para o formato de instruções utilizado no repositório *LLMs from Scratch*.
- **Question Answering** – `qa_cnn_instruction_data.json`
  As instruções foram geradas por meio do transform do RAGAS, garantindo que as respostas fossem estritamente fundamentadas no campo `reference_contexts` (o artigo), evitando alucinações.

SST-2 – `sst2_instruction_data.json`

- **Sentiment analysis**
  Conversão direta do dataset supervisionado para o formato de instrução.

AG News – `agnews_instruction_data.json`

- **Classification**
  A partir dos highlights dos artigos foram extraídas palavras-chave que serviram como base para a rotulação.

SQuAD – `qa_squad_instruction_data.json`

- **Question answering**
  Adaptado para o formato de instrução.
  Entretanto, devido ao tamanho reduzido do modelo, o treinamento apresentou dificuldades de convergência.

Dataset do Sebastian Raschka – `sebastian_instruction_data.json`

Para complementar e aprimorar o conjunto de dados construído, foi realizada a concatenação com o dataset original disponível no repositório de Sebastian

Raschka. Foram empregadas 1000 instruções adicionais dele.

Curadoria e Filtragem

Para o subconjunto de QA do CNN, foi realizada uma etapa de auditoria automática com uma LLM externa, verificando se:

- toda informação da resposta vinha dos artigos
- não havia dependência semântica da pergunta para inferir a resposta

Além disso, foram removidas personas que introduziam ruído linguístico, reduzindo o dataset de **300 → 273 instruções válidas**.

exemplo:

```
{
  "instruction": "Whee wa Cui Hongfang from?",
  "input": "",
  "output": "Cui Hongfang was from Heilongjiang Province in north-eastern China."
}
```

Decisões e Problemas Encontrados

Primeira Tentativa: Problema com o Conhecimento do Modelo

Inicialmente, tentei treinar o modelo apenas com as instruções que criei. A ideia era que o fine-tuning seria suficiente para ensinar o GPT-2 a seguir instruções. O problema foi que o GPT-2 não tem conhecimento factual sobre notícias — mesmo notícias antigas não fazem parte do que ele aprendeu no pré-treinamento. Como o modelo não tinha essa base de conhecimento sobre jornalismo, o fine-tuning de instruções sozinho não conseguiu compensar.

O resultado foi ruim: **validation loss ficou em torno de 2.2** e as respostas eram muito instáveis e inconsistentes. Isso mostra que o fine-tuning de instruções funciona melhor quando os dados de treino estão próximos do conhecimento que o modelo já possui.

Segunda Tentativa: Limitações de Memória

Depois, tentei treinar com **2.000 instruções** usando **batch size 8**. Porém, o Colab (16GB de VRAM) não aguentou. O problema foi o tamanho das sequências: os datasets de CNN/DailyMail (Summarization) e SQuAD (Question Answering) têm textos muito longos, o que consumia muita memória. Mesmo quando reduzi para batch size 4, ainda deu erro de memória.

A solução foi reduzir a quantidade de dados: usei **1.000 instruções** para o modelo completo (model full) e **500 instruções** para o modelo pequeno (model

small). Com isso, consegui completar o treinamento sem estourar a memória. Mesmo com menos dados, os resultados mostraram uma melhora significativa comparado ao modelo base.

Treinamento

Configuração do Experimento (Hiperparâmetros)

O processo de treinamento foi conduzido utilizando o script disponibilizado no repositório *LLMs from Scratch*, com os seguintes hiperparâmetros:

- **Batch Size:** 8
- **Learning Rate (LR):** 5e-5 (0.00005)
- **Weight Decay:** 0.1
- **Épocas:** 2

Modelos e Divisão de Dados

Foram realizados dois experimentos independentes utilizando a arquitetura **GPT-2 (pré-treinado)**:

1. **Modelo GPT-2 Full:** Treinado com o conjunto completo de 1.000 instruções.
   - *Split:* 800 (Treino) | 100 (Validação) | 100 (Teste).
2. **Modelo GPT-2 Small:** Treinado com um subconjunto de 500 instruções.
   - *Split:* 400 (Treino) | 50 (Validação) | 50 (Teste).

Resultados do Treinamento

Os valores de perda de validação (validation loss) obtidos foram:

Modelo	Val Loss
Full	**1.158**
Small	**1.283**

As curvas de treinamento e validação podem ser observadas nos gráficos abaixo:

Modelo Full (1000 instruções):

Modelo Small (500 instruções):

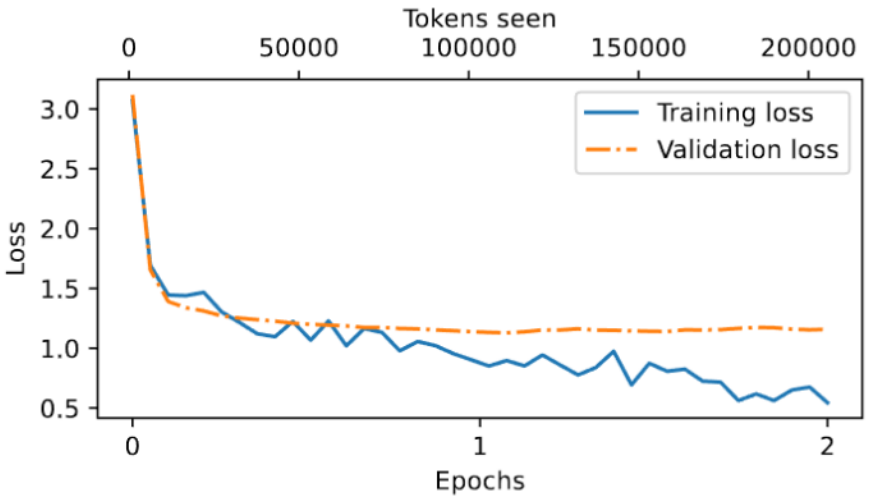

Figure 1: Loss Plot Full Model

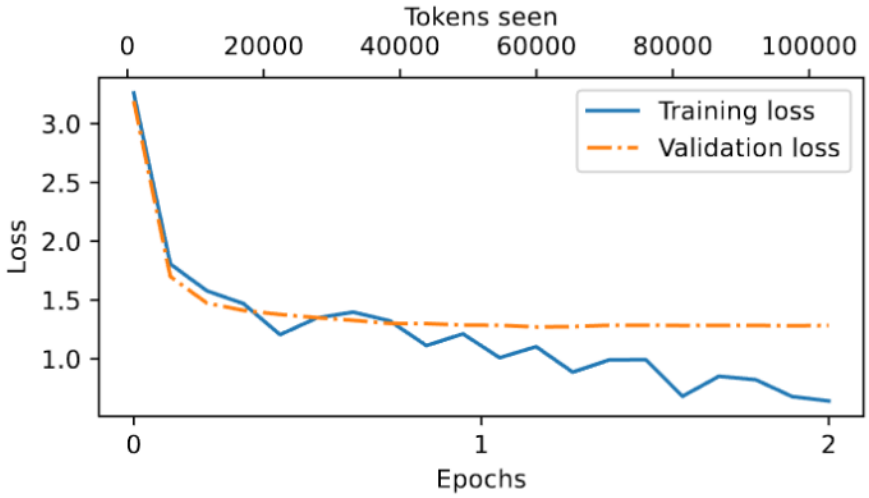

Figure 2: Loss Plot Small Model

Avaliação e Testes

Para avaliar o desempenho dos modelos treinados, foram geradas respostas estruturadas para um conjunto de instruções de teste. Esse processo permitiu medir objetivamente a capacidade de cada modelo em seguir instruções e produzir respostas adequadas.

Foram geradas 3 respostas distintas, uma para cada modelo:

- **Model Full**: Modelo GPT-2 treinado com 1.000 instruções
- **Model Base**: Modelo GPT-2 original sem fine-tuning
- **Model Small**: Modelo GPT-2 treinado com 500 instruções

Todas as respostas foram armazenadas em `all_model_responses.json`, seguindo o formato estruturado abaixo:


```
{
  "instruction": "What is the capital of Mexico?",
  "input": "",
  "output": "The capital of Mexico is Mexico City.",
  "model full": "The capital of Mexico is Mexico City.",
  "model base": "### Answer:\n\n### Instruction:\n\nWhat is the capital of Mexico?...",
  "model small response": "The capital of Mexico is Mexico City."
}
```


Como pode ser observado no exemplo, o modelo base (sem fine-tuning) apresenta comportamento repetitivo e não consegue seguir o formato de instrução adequadamente, enquanto os modelos treinados produzem respostas diretas e apropriadas.

LLM as a Judge

Para avaliar objetivamente a qualidade das respostas geradas por cada modelo, foi utilizada a abordagem **LLM as a Judge**, empregando o **GPT-4.1** como avaliador. Essa técnica consiste em usar um modelo de linguagem mais avançado para julgar e pontuar as respostas de outros modelos de forma imparcial e consistente.

A escolha do GPT-4.1 se deve à sua maior capacidade de raciocínio, compreensão de contexto e habilidade em seguir critérios específicos de avaliação. O modelo juiz recebe cada resposta gerada e atribui uma pontuação baseada em critérios bem definidos.

Métricas de Avaliação

Foram implementadas três métricas personalizadas, cada uma focada em um aspecto específico da qualidade das respostas:

1. Factual Correctness (Correção Factual) Esta métrica avalia se a resposta gerada está factualmente consistente com a saída esperada(Veracidade das informações fornecidas).

```
name="FactualCorrectness",
evaluation_steps=[
    "Use 'expected output' as the reference answer.",
    "Decide whether 'actual output' is factually consistent with the expected output.",
    "If actual output contradicts the expected output, score near 0.",
    "If it matches or is an equivalent correct answer, score near 1.",
    "Penalize major missing required facts when the task expects a concrete answer."
],
```

Parâmetros avaliados: instrução de entrada, resposta gerada pelo modelo, e resposta esperada.

2. Instruction Adherence (Aderência à Instrução) Esta métrica verifica se a resposta segue corretamente a instrução fornecida, independentemente da correção factual.

```
name="InstructionAdherence",
evaluation_steps=[
    "Check whether the response follows the instruction and uses the provided input (if any)
    "Penalize if it ignores the task, repeats templates, or outputs irrelevant content.",
    "If the instruction implies a specific format (e.g., classification label, rewrite, shor
    "Do NOT judge factual correctness here; only whether it follows the instruction."
],
```

Parâmetros avaliados: instrução de entrada e resposta gerada pelo modelo.

3. Clarity and Utility (Clareza e Utilidade) Esta métrica avalia se a resposta é clara, concisa e útil para o usuário.

```
name="ClarityUtility",
evaluation_steps=[
    "Assess whether the response is clear, concise, and useful for the user.",
    "Penalize repetition, boilerplate, excessive verbosity, or confusing structure.",
    "Reward direct answers with minimal noise.",
    "For simple tasks, the best response is short and unambiguous."
],
```

Parâmetros avaliados: instrução de entrada e resposta gerada pelo modelo.

Cada métrica atribui uma pontuação que é posteriormente utilizada para comparar o desempenho dos três modelos (Base, Full e Small), permitindo uma análise quantitativa e qualitativa dos resultados do fine-tuning.

Resultados

Médias

Modelo	Factual	Adherence	Clarity	Total
Full	**3.362**	**4.162**	**2.716**	**10.241**
Base	0.18	0.243	0.897	1.32
Small	2.566	3.896	2.66	9.122

Os resultados demonstram a eficácia do fine-tuning. O modelo Full apresentou o melhor desempenho geral (10.241), seguido pelo modelo Small (9.122). O modelo Base obteve pontuação muito baixa (1.32), evidenciando sua incapacidade de seguir instruções sem treinamento específico.

Win Rate

Comparação	Win	Tie	Loss
Full vs Base	97%	3%	0%
Full vs Small	45.5%	12%	42.5%

O modelo Full dominou completamente o modelo Base, vencendo em 97% dos casos. Já a competição entre Full e Small foi equilibrada, com vitórias distribuídas de forma relativamente uniforme, indicando que o modelo menor também apresenta bom desempenho.

Ganho Médio

Comparação	Average Gain	Max Gain	Min Gain
Full vs Base	8.921	15	-6.22
Full vs Small	1.119	10.5	-10.08

O ganho médio do modelo Full sobre o Base foi de 8.921 pontos, com ganho máximo de 15 pontos. Em relação ao modelo Small, o ganho médio foi menor (1.119), mas ainda significativo em casos específicos.

Distribuição de Scores

As distribuições de pontuações entre os modelos podem ser visualizadas nos gráficos abaixo:

Full vs Base:

Full vs Small:

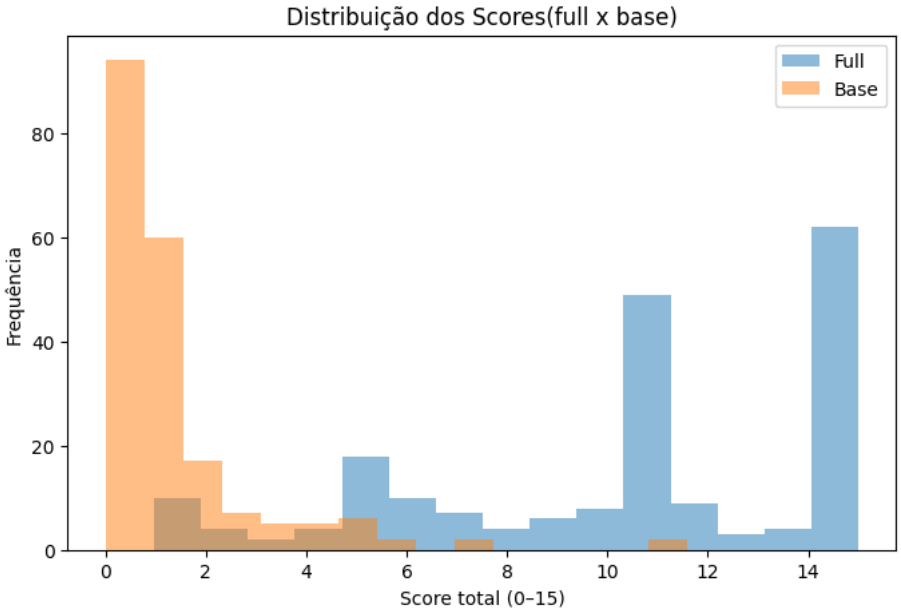

Figure 3: Distribuição de Scores Full vs Base

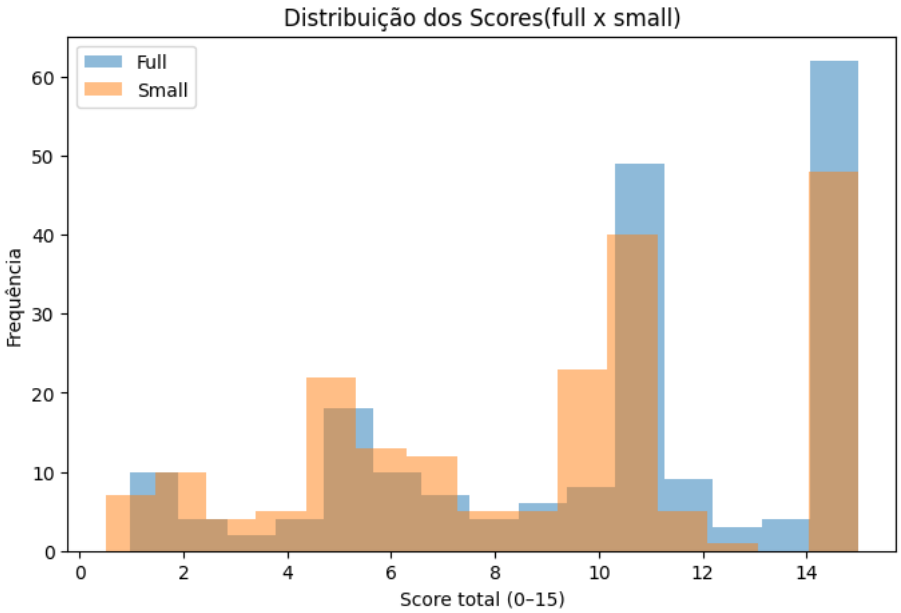

Figure 4: Distribuição de Scores Full vs Small

Análise Qualitativa

Para compreender melhor as diferenças de desempenho, analisamos casos específicos onde os modelos apresentaram comportamentos distintos.

Melhor Caso: Full vs Small (ID: 142 do json das responses) Este caso representa a maior diferença de desempenho entre os modelos Full e Small, com um delta de **10.5 pontos**.

Tarefa:

```
Classify the news topic into: World, Sports, Business or Science/Technology.
```

Input:

```
Apple Introduces Production Suite Production Suite, essential software suite
for film and video that delivers real-time production tools in one comprehensive
and integrated package, combines Final Cut Pro HD, Motion and DVD Studio Pro 3.
Production Suite is available now for a suggested retail price of $1,299. Aug 10
```

Resultado:

Modelo	Resposta	Score
Expected	Science/Technology	15
Full	Science/Technology	11.0
Small	, 2015	0.5

Análise da Falha do Modelo Small feita pelo juiz:

> **Factual:** The actual output ', 2015' is not factually consistent with the expected output 'Science/Technology' and does not classify the news topic as required. It omits the necessary classification and provides irrelevant information, missing all required facts for a correct answer.
>
> **Adherence:** The response does not follow the instruction to classify the news topic and instead outputs an irrelevant date fragment. It ignores the input and required format, failing to provide any classification label as requested.
>
> **Clarity:** The response is unclear and lacks context, consisting only of a comma and a year, which is not useful or informative. It does not directly answer any question and fails to meet clarity or conciseness requirements.

Este exemplo ilustra uma limitação crítica do modelo Small: embora consiga desempenhar bem na maioria dos casos, ocasionalmente apresenta **falhas catastróficas** mesmo em tarefas simples de classificação. O modelo Full, beneficiando-se do dobro de dados de treinamento (1.000 vs 500 instruções),

demonstrou maior robustez e estabilidade, produzindo a resposta correta `Science/Technology` de forma direta e precisa.

Conclusão

O fine-tuning transformou o GPT-2 de um modelo gerador de texto em um modelo capaz de seguir instruções, com o modelo Full alcançando pontuação 7.7x superior ao modelo Base.

O ganho veio mais do modelo ou dos dados?

A evidência aponta que **o ganho veio predominantemente dos dados**, não da arquitetura:

- O modelo Small alcançou 89% do desempenho do Full com metade dos dados (500 vs 1000 instruções)
- Ambos os modelos usam a arquitetura GPT-2 idêntica; a diferença está no volume de treinamento

O juiz tem viés?

Sim, o GPT-4.1 possui viés inerente (formato, comprimento, alinhamento). Porém, as mitigações incluem:

- Divisão em três métricas independentes (Factual, Adherence, Clarity)
- Critérios explícitos nos `evaluation_steps`
- Avaliação consistente para todos os modelos