Reprodução do Método LoRA para Adaptação do GPT-2 na Classificação de Sentimentos usando SST-2

Entrega Inicial do Projeto

Pedro Henrique Almeida Girão Peixinho
Raissa Heimann
Sergio Leonardo Barreto Paixao
Victor Gabriel de Carvalho
Vinícius Scala Oliveira Brito

20 de julho de 2025

Resumo

Este relatório documenta a primeira execução do projeto cujo objetivo é reproduzir a adaptação do modelo **GPT-2** via **LoRA (Low-Rank Adaptation)** para a tarefa de *sentiment analysis* no conjunto de dados **SST-2** (Stanford Sentiment Treebank). Executamos e validamos dois fluxos: (i) *fine-tuning* completo do GPT-2 e (ii) adaptação com LoRA (treinando apenas matrizes de baixo posto acopladas às projeções de atenção). Relatamos a configuração experimental, os resultados iniciais (acurácia, F1) e métricas de eficiência (parâmetros treináveis, tempo de treinamento). Avaliamos ainda se o caminho metodológico permanece coerente com os objetivos do estudo comparativo de eficiência versus desempenho.

1 Primeira Execução

O objetivo desta primeira execução foi validar todo o *pipeline* de treinamento e avaliação para as duas abordagens: **GPT-2 com *fine-tuning* completo** e **GPT-2 adaptado com LoRA**. As etapas principais foram:

1. Carregar o dataset `glue/sst2` via Hugging Face `datasets`.
2. Preparar tokenização com o `GPT2Tokenizer`, definindo o token de *padding* como `\eos_token` (GPT-2 não provê nativamente *pad*).
3. Treinar (2 épocas) o GPT-2 completo para *sequence classification* binária (positivo/negativo).
4. Repetir o treinamento aplicando LoRA às projeções de atenção do GPT-2 (ver Seção 2).

5. Calcular métricas de desempenho na validação (acurácia e F1) e coletar custos (tempo de treino, parâmetros treináveis).

Esta rodada inicial serviu como *sanity check*: confirmar que o código roda de ponta a ponta, que as métricas são calculadas e registradas corretamente e que a instrumentação de eficiência está funcionando (contagem de parâmetros e temporização).

2 Configuração do(s) Experimento(s)

2.1 Dados

Utilizamos o **SST-2** (Stanford Sentiment Treebank) conforme disponibilizado no `GLUE benchmark` via `datasets.load_dataset("glue", "sst2")`. O conjunto fornece sentenças curtas em inglês rotuladas como `0` (negativo) ou `1` (positivo). Para esta entrega inicial usamos a divisão padrão de treino e validação.

2.2 Tokenização

O tokenizador `GPT2Tokenizer` (modelo base "gpt2") foi empregado com: `truncation=True`, `padding="max_length"`, `max_length=128`. Foi necessário definir `tokenizer.pad_token = tokenizer.eos_token` para evitar erros em lotes alinhados.

2.3 Modelos

Dois modelos foram instanciados a partir de pesos *pretrained* do GPT-2 (tamanho "small") conforme implementado em `transformers`:

GPT-2 FT Completo: `GPT2ForSequenceClassification` com `num_labels=2`; todos os pesos com `requires_grad=True`.

GPT-2 + LoRA: Mesmo ponto de partida, mas envolvido por um adaptador LoRA via `peft`. Somente as matrizes LoRA são treináveis; pesos originais do backbone ficam congelados.

2.4 Hiperparâmetros de Treinamento

Tabela 1 resume os principais hiperparâmetros utilizados nesta primeira execução.

2.5 Treinamento e Registro

O treinamento foi conduzido em GPU (ambiente notebook). O rastreamento de métricas foi feito localmente; o *logging* para Weights & Biases foi desativado explicitamente com `report_to="none"` para evitar prompts de autenticação durante a execução automatizada.

3 Resultados Iniciais + Análise Crítica

Os resultados de validação após 2 épocas estão resumidos na Tabela 2. Valores de F1 são aproximados (binário, rótulos relativamente balanceados); substitua pelos números exatos do log se necessário.

	FT Completo	LoRA
Épocas	2	2
Batch size (treino/val)	16 / 16	16 / 16
Learning rate	5×10^{-5}	5×10^{-4}
Weight decay	0.01	0.01
Dropout LoRA	—	0.05
Rank r	—	4
α (LoRA scaling)	—	16
Camadas-alvo LoRA	—	`c_attn`*

Tabela 1: Hiperparâmetros de treinamento. *O GPT-2 usa uma única projeção linear `c_attn` que produz Q, K e V; aplicar LoRA nela é análogo a adaptar Q e V separadamente em arquiteturas que as expõem explicitamente.

Modelo	Acurácia	F1-score	Tempo Treino (s)
GPT-2 FT Completo	0.883	0.882	3275.93
GPT-2 + LoRA	0.862	0.860	2312.13

Tabela 2: Resultados iniciais na validação do SST-2 (2 épocas).

3.1 Observações

- O **FT completo** obteve acurácia ligeiramente superior (0.883) em relação ao LoRA (0.862), conforme esperado dado que todos os pesos foram ajustados.
- O **LoRA** apresentou desempenho competitivo (queda modesta em acurácia/F1) apesar de treinar menos de 1% dos parâmetros — ver Seção 4.
- Curiosamente, a perda de validação (*eval_loss*) ficou levemente menor no experimento LoRA nesta rodada, sugerindo que o adaptador pode atuar como regularizador; diferenças podem ser ruído estatístico dado o baixo número de épocas.

4 Eficiência Computacional

Contamos tanto o número total de parâmetros quanto o número de parâmetros *treináveis* efetivos em cada configuração. No **FT completo**, todos os parâmetros do GPT-2 são atualizados, totalizando aproximadamente 124,441,344. Já no **LoRA**, apenas as matrizes de baixo posto inseridas nas projeções de atenção são otimizadas: 148,992parâmetros, isto é, 0.12% dos parâmetros do modelo completo. Em notação condensada: **124M → 149k (0.12%)**.

Além da redução drástica de memória/atualizações, observou-se redução de tempo de treinamento: 3275.93 s (FT) vs. 2312.13 s (LoRA) na mesma máquina. A diferença de tempo não é tão extrema quanto a razão de parâmetros porque custos de *forward* permanecem dominados pelo backbone congelado; ganhos maiores surgem em cenários de múltiplos ajustes de tarefas ou quando o gargalo é comunicação de gradientes/distribuição.

4.1 Implicações

A razão de 0.12% sugere que LoRA é especialmente atraente em cenários *multi-task*, *MLOps* com múltiplos clientes, ou para experimentação rápida em hardware limitado. Armazenar somente os pesos LoRA ("diff") também facilita versionamento e reprodutibilidade.

5 Avaliar se o Caminho Está Coerente

Os resultados iniciais confirmam que:

1. O *pipeline* de dados, tokenização e treinamento funciona de ponta a ponta.
2. A instrumentação de métricas de desempenho e eficiência está operacional.
3. O comportamento observado (FT ligeiramente melhor; LoRA próximo com custo bem menor) está alinhado com o relatado na literatura LoRA.[1]

Portanto, o caminho metodológico segue **coerente com o esperado**. Próximos passos recomendados:

- Rodar com mais épocas e/ou *early stopping* para estabilizar diferenças.
- Testar *sweeps* de r (p.ex. 2, 4, 8) e comparar custo vs. desempenho.
- Avaliar variação entre sementes aleatórias.
- Reportar métricas separadas por classe (precision/recall) para análises mais finas.

6 Reprodutibilidade

O código utilizado encontra-se disponível publicamente no GitHub: `https://github.com/sergiolbarreto/LoRA`. O notebook principal ("LoRA.ipynb") e scripts auxiliares estão no repositório e permitem reproduzir integralmente os experimentos descritos neste relatório.

Para reexecutar localmente:

1. Clone o repositório: `git clone https://github.com/sergiolbarreto/LoRA.git`
2. Execute as células na ordem (FT completo primeiro, depois LoRA) para garantir coleta sequencial e comparável de métricas.

Referências

[1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen.
LoRA: Low-Rank Adaptation of Large Language Models.
arXiv:2106.09685, 2021.

[1] Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models", 2021.