Análise Comparativa de Arquiteturas de Modelos de Linguagem: Um Estudo de Caso entre GPT-2 e Mixture of Experts

Matheus Hensley de Figueiredo e Silva[1]

Abstract

Este relatório apresenta um estudo de caso comparativo entre duas arquiteturas distintas de modelos de linguagem de grande escala (LLMs). O objetivo é analisar os trade-offs de desempenho, eficiência de treinamento e consumo de recursos entre um modelo baseline, baseado na arquitetura original do GPT-2, e um modelo moderno que incorpora otimizações como Mixture of Experts (MoE) e Grouped-Query Attention (GQA). Ambos os modelos foram implementados do zero, treinados e avaliados em um corpus de textos literários em língua portuguesa extraído do Projeto Gutenberg, sob restrições computacionais realistas de uma única GPU. A análise busca fornecer insights práticos sobre os benefícios e custos associados a cada abordagem.

1 Introdução

O campo dos Modelos de Linguagem de Grande Escala (LLMs) tem evoluído a uma velocidade notável, com novas arquiteturas e técnicas sendo propostas constantemente para aprimorar a capacidade, eficiência e escalabilidade desses modelos. Embora modelos mais antigos como o GPT-2 estabeleceram as fundações da geração de texto baseada em Transformers, arquiteturas mais recentes introduziram inovações que otimizam o processo de treinamento e inferência.

Este relatório documenta um estudo de caso prático focado na comparação de duas abordagens distintas para a construção de LLMs. A principal motivação deste trabalho é investigar, em um ambiente controlado e com recursos computacionais limitados, as diferenças empíricas entre uma arquitetura clássica e uma moderna, buscando entender não apenas as métricas de performance, como a perplexidade, mas também os aspectos de engenharia, como throughput de treinamento e uso de memória. Para este experimento, foi selecionado um corpus de textos em português e português do Brasil, extraído do Projeto Gutenberg. A escolha se deu pela livre disponibilidade e pela riqueza linguística das obras literárias, ainda que representem um dialeto mais antigo da língua. Este subconjunto de dados, apesar de modesto em tamanho (50 MB), oferece um desafio interessante e suficiente para observar as dinâmicas de aprendizado dos modelos de linguagem.

Ao longo deste relatório, serão apresentadas as etapas metodológicas de implementação e treinamento, os resultados quantitativos e qualitativos obtidos, bem como uma discussão crítica sobre os trade-offs entre qualidade, eficiência e custo computacional.

2 Metodologia

Esta seção detalha os componentes do experimento, incluindo as arquiteturas dos modelos implementados, o processo de preparação dos dados e a configuração utilizada para o treinamento. O objetivo não foi treinar modelos de alta performance, mas observar empiricamente diferenças estruturais entre as duas arquiteturas sob um cenário de treinamento realista.

2.1 Arquiteturas dos Modelos

Dois modelos foram implementados do zero em Python utilizando a biblioteca PyTorch, garantindo controle total sobre a arquitetura e os parâmetros. Ambos foram projetados para operar dentro das restrições de uma GPU NVIDIA P100 com 16GB de VRAM.

2.1.1 Modelo Baseline: GPT-2

O primeiro modelo segue fielmente a arquitetura GPT-2 original, servindo como baseline para comparação. Ele consiste em uma pilha de blocos decodificadores Transformer, com as seguintes características:

1. Universidade Federal de Campina Grande, Brazil, matheus.figueiredo.silva@ccc.ufcg.edu.br

- Um mecanismo de **Multi-Head Self-Attention** mascarado, onde cada token atende apenas a tokens anteriores, respeitando a causalidade do modelo autoregressivo.
- **Feed-Forward Network** (FFN) em que cada bloco contém uma rede feed-forward com duas camadas lineares separadas por uma função de ativação GELU, responsável por aumentar a expressividade não linear.
- **Layer Normalization** (pré-norm) aplicada antes de cada sub-bloco (attention e FFN), estabilizando o gradiente e melhorando a convergência.
- **Conexões residuais** contornando cada sub-bloco, permitindo preservar informações ao longo das camadas e facilitar o fluxo de gradiente.

Os parâmetros do modelo, como dimensão de embedding e número de camadas, foram significativamente reduzidos para se adequarem ao escopo do experimento e às limitações de hardware.

2.1.2 Modelo Moderno: MoE-GQA

O segundo modelo segue princípios de arquiteturas mais recentes, como Qwen2/3 e Llama 3, priorizando eficiência computacional e escalabilidade modular. O foco deste trabalho foi inspirado principalmente nas otimizações da família Qwen, por sua integração prática de técnicas modernas com bom custo-benefício de treinamento. As principais diferenças em relação ao modelo baseline são:

- **Grouped-Query Attention (GQA):** Em vez de atribuir conjuntos independentes de chaves (K) e valores (V) para cada cabeça de atenção, o GQA agrupa múltiplas cabeças de consulta (Q) para compartilharem um mesmo conjunto de K/V. Isso reduz significativamente a quantidade de parâmetros e o custo de memória da atenção, sem perda substancial de desempenho.
- **Mixture of Experts (MoE):** As camadas de rede feed-forward (FFN) são substituídas por blocos MoE. Uma rede roteadora seleciona um subconjunto de "experts" (FFNs independentes) para processar cada token, aumentando a capacidade do modelo com custo computacional constante.
- **RMSNorm:** A LayerNorm é substituída pela RMSNorm (Root Mean Square Normalization), uma alternativa computacionalmente mais eficiente que remove a necessidade de subtrair a média. Essa simplificação reduz custo computacional e melhora a estabilidade numérica em hardware de precisão reduzida, como bfloat16.
- **Rotary Position Embeddings (RoPE):** Técnica de posicionamento relativo que rotaciona vetores de consulta e chave no espaço, permitindo que o modelo capture relações posicionais de forma mais natural e generalizável.

2.2 Dados e Pré-processamento

O corpus utilizado consiste em 50 MB de textos de obras literárias em português, obtidos do Projeto Gutenberg. O conjunto de dados foi dividido em 80% para treino, 10% para validação e 10% para teste. O pré-processamento envolveu a tokenização dos textos com o tokenizer do GPT-2 (via tiktoken) e a segmentação em amostras de tamanho fixo, posteriormente realizando a conversão para tensores PyTorch para uso direto nos DataLoaders de treino, validação e teste. A tabela 1 apresenta informações sobre o conjunto de dados após a tokenização.

Table 1: Estatísticas do conjunto de dados após tokenização.

Métrica	**Treino**	**Validação**	**Teste**
Palavras (aprox.)	5.596.830	692.017	692.017
Total de Amostras	101.482	12.843	13.471
Tokens por Amostra	256	256	256
Tamanho do Batch	8	8	8
Número de Batches	12.685	1.605	1.683

2.3 Configuração de Treinamento

Ambos os modelos foram treinados em ambiente Kaggle Notebooks, utilizando GPU NVIDIA Tesla P100 (16 GB). O otimizador adotado foi o AdamW, amplamente utilizado em LLMs pela estabilidade na atualização dos pesos e controle mais robusto de regularização via decaimento de peso. A precisão numérica utilizada foi bfloat16, por oferecer melhor desempenho e menor consumo de memória sem a necessidade de técnicas de mixed precision mais complexas. Abaixo estão os hiperparâmetros compartilhados.

```
CONFIG = {
    "vocab_size": 50257,
    "emb_dim": 512,
    "context_length": 256,
    "n_layers": 8,
    "n_heads": 8,
    "head_dim": 64,
    "hidden_dim": 2048,
    "n_kv_groups": 4,            # GQA: usado apenas no modelo moderno
    "qk_norm": True,
    "rope_base": 10000.0,
    "num_experts": 4,            # MoE: quantidade de experts
    "num_experts_per_token": 2,  # MoE: experts ativos por token
    "moe_intermediate_size": 1024,
    "bias": False,

    "batch_size": 8,
    "max_epochs": 2,
    "num_workers": 0,
    "stride": 128,               # Stride = context_length // 2
    "dtype": "torch.bfloat16",
    "device": "cuda",

    "eval_freq": 100,            # Frequencia de validacao (em batches)
    "eval_iter": 16,             # Batches para rodar na validacao
}
```

Durante o treinamento, foi utilizado o **Weights & Biases** (W&B), uma plataforma de experimentação e monitoramento amplamente usada em pesquisa com deep learning, para armazenar checkpoints automáticos. Os pesos do modelo foram salvos a cada 5000 batches e uma última vez ao final de cada época, garantindo reprodutibilidade, recuperação em caso de falhas e rastreabilidade dos experimentos.

3 Resultados e Análise Qualitativa

Nesta seção, apresenta-se os resultados quantitativos e qualitativos obtidos ao final do treinamento de ambos os modelos.

3.1 Métricas de Desempenho e Eficiência

As métricas de desempenho no conjunto de teste, juntamente com as de eficiência computacional, estão compiladas na Tabela 2.

3.2 Curvas de Aprendizado

As curvas de perda (loss) de treinamento e validação para ambos os modelos são mostradas abaixo. Visualmente, ambos os modelos apresentam uma curva de aprendizado estável, com a perda diminuindo rapidamente no início e estabilizando ao longo do tempo, como apresentado nas figuras.

Table 2: Resultados consolidados do treinamento e avaliação.

Métrica	Modelo Baseline (GPT-2)	Modelo Moderno (MoE-GQA)	Variação (%)
Loss (Teste)	3.2699	**3.2127**	−1.75
Perplexidade (Teste)	26.31	**24.85**	−5.55
Tempo Total de Treino (s)	3882.26	**3623.42**	−6.67
Memória VRAM Máxima (MB)	3243.45	**2823.18**	−12.96
Throughput (tokens/s)	**13346.97**	8648.30	−35.21

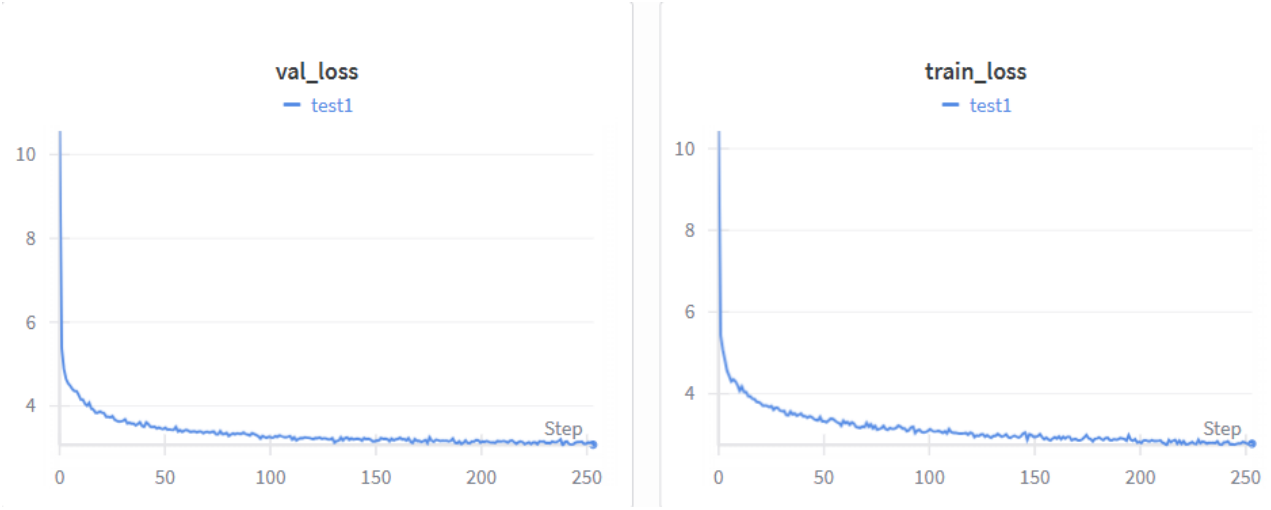

Figure 1: **Curvas de Loss (Validação e Treino) - Modelo Baseline (GPT-2)**

3.3 Análise Qualitativa da Geração de Texto

Para uma avaliação qualitativa simples, os dois modelos foram instruídos a gerar texto a partir do mesmo contexto inicial: "lisboa". Os resultados, embora limitados pela curta duração do treinamento, oferecem um vislumbre de suas capacidades.

- **Modelo Baseline (GPT-2):** "lisboa de noite ao seu hospede. Apenas achou a esperana, porque as pessoas que iam, como estavam muito bruscas e muito linda saborosas, podiam fazer-se"...
- **Modelo Moderno (MoE-GQA):**"lisboa, se faziam já nos prazos, e nada havia que a ajudasse, e tornasse o fim ás tres horas da manhã. Pouco depois, chegou o tempo e viuva, e"...

Ambos os modelos geram textos com falhas de coerência, o que é esperado dado o escopo do treinamento. No entanto, a geração do modelo moderno parece ter uma estrutura sintática ligeiramente mais plausível, com referências temporais ("tres horas da manhã", "Pouco depois") que sugerem uma tentativa de fluxo narrativo. A saída do modelo baseline parece mais desconexa e semanticamente confusa ("muito bruscas e muito linda saborosas").

4 Discussão

A análise dos resultados revela um claro trade-off entre qualidade e custo computacional, com vantagens significativas para a arquitetura moderna.

O modelo MoE-GQA alcançou uma perplexidade 5.55% menor no conjunto de teste, indicando uma capacidade superior de prever a sequência de tokens e, consequentemente, um melhor entendimento da estrutura da linguagem presente no corpus. Essa melhoria na qualidade foi obtida com maior eficiência: o treinamento foi 6.67% mais rápido e consumiu 12.96% menos memória VRAM.

A redução no uso de memória é um benefício direto da Grouped-Query Attention (GQA), que diminui o tamanho do cache de chaves e valores, um dos principais gargalos de memória em modelos baseados em Transformer. A arquitetura Mixture of Experts (MoE), por sua vez, permite que o modelo tenha uma capacidade teórica maior (mais parâmetros nos experts) sem aumentar o custo computacional por token, o que provavelmente contribuiu para a convergência mais rápida e a menor loss final.

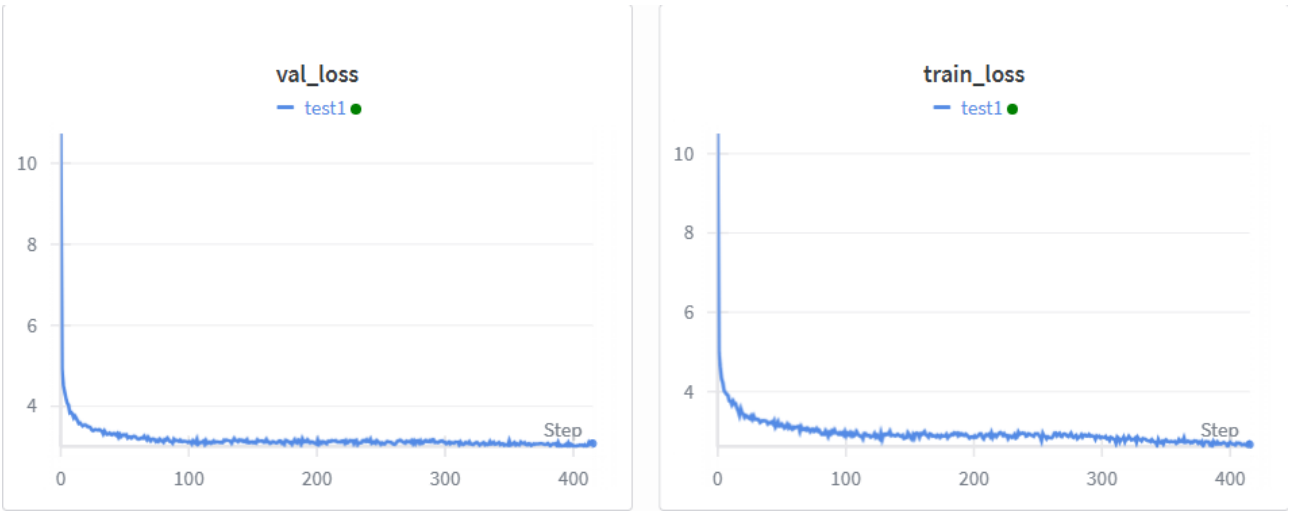

Figure 2: **Curvas de Loss (Validação e Treino) - Modelo Moderno (MoE-GQA)**

Um ponto contraintuitivo é a redução do throughput (tokens/s) no modelo moderno. Embora o tempo total de treino tenha sido menor, a velocidade de processamento instantânea foi mais baixa. Isso sugere que, embora cada passo de treinamento seja computacionalmente mais complexo (devido à lógica de roteamento do MoE), o modelo aprende de forma mais eficiente, necessitando de menos passos ou tempo para atingir um nível de performance superior. Essencialmente, o modelo troca velocidade por amostra por eficiência de aprendizado.

Qualitativamente, a superioridade do modelo moderno, embora sutil, aponta para uma melhor captura de padrões narrativos básicos. Isso reforça que as otimizações arquiteturais não apenas melhoram as métricas, mas também podem se traduzir em uma geração de texto mais coerente.

5 Conclusão e Trabalhos Futuros

Este estudo de caso demonstrou empiricamente os benefícios de arquiteturas modernas de LLMs em comparação com uma abordagem clássica como a do GPT-2. Em um ambiente com recursos computacionais limitados, o modelo que incorpora MoE, GQA, RMSNorm e RoPE não só atingiu uma qualidade superior, medida pela perplexidade, como também o fez de forma mais eficiente, com menor tempo de treinamento e menor consumo de memória.

Os resultados confirmam que essas inovações arquiteturais são cruciais para democratizar o desenvolvimento e o treinamento de LLMs, tornando-os viáveis mesmo fora de grandes centros de pesquisa com vastos recursos computacionais. Como trabalhos futuros, diversas avenidas de exploração podem aprofundar os aprendizados deste experimento:

- **Melhora do Conjunto de Dados:** O uso de um corpus maior, mais diversificado e com textos em português brasileiro mais modernos poderia melhorar drasticamente a coerência e a qualidade da geração de texto. A limpeza e a curadoria dos dados são passos fundamentais.
- **Escalabilidade do Treinamento:** Com acesso a hardware mais robusto (ex: GPUs NVIDIA A100 ou H100), seria possível treinar modelos com mais parâmetros (mais camadas, maior dimensão de embedding) por um número maior de épocas, explorando o verdadeiro potencial de cada arquitetura em larga escala.
- **Ajuste de Hiperparâmetros:** Realizar uma busca mais sistemática por hiperparâmetros, especialmente os relacionados ao MoE (número de experts, experts por token), poderia otimizar ainda mais o trade-off entre performance e custo.
- **Avaliação Abrangente:** Utilizar métricas de avaliação mais sofisticadas, além da perplexidade, como ROUGE, BLEU, ou até mesmo avaliações humanas, para obter uma medida mais precisa da qualidade da geração de texto.

6 Referências Bibliográficas

Lepikhin, Dmitry, Hyouk-Joong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. *GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding*. arXiv, 2020. https://arxiv.org/abs/2006.16668.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. Accessed on October 17, 2025. OpenAI, 2019. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Raschka, Sebastian. "LLMs from Scratch." Accessed on October 17, 2025. https://github.com/rasbt/LLMs-from-scratch.

Su, Jianlin, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. "RoFormer: Enhanced Transformer with Rotary Position Embedding." *CoRR* abs/2104.09864 (2021). https://arxiv.org/abs/2104.09864.

Yang, An, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. "Qwen3 Technical Report." *CoRR* abs/2505.09388 (2025). https://arxiv.org/abs/2505.09388.