Capítulo 4

Incorporação de Modelos de Linguagem em Larga Escala em Dispositivos Móveis: Otimização, Personalização e Desafios

Nicollas R. de Oliveira (UFF), João Vitor V. Silva (UFF), Guilherme N. N. Barbosa (UFF), Dianne S. V. de Medeiros (UFF), Diogo M. F. Mattos (UFF)

Abstract

The chapter discusses techniques and tools for optimizing and customizing Large Language Models (LLMs) for resource-constrained devices. The high computational cost and power consumption challenges of these models limit their applicability in environments with diverse capabilities. Thus, the chapter introduces models such as GPT, Llama, and Falcon, based on architectures such as AutoEncoders and Transformers, which have transformed natural language processing with impressive text understanding and generation capabilities. Quantization is presented as a crucial technique for improving the usability of models on resource-constrained devices. Finally, the practical applications of adapting and embedding quantized models on resource-constrained devices and the risks associated with the use of LLMs are considered.

Resumo

O capítulo discute as técnicas e ferramentas para otimizar e personalizar modelos de linguagem em larga escala (Large Language Models - LLMs) para dispositivos com recursos limitados. Os desafios impostos pelo alto custo computacional e consumo de energia desses modelos limitam sua aplicabilidade em ambientes com capacidades variadas. Assim, o capítulo apresenta modelos como GPT, Llama e Falcon, baseados em arquiteturas como AutoEncoders e Transformers, que transformaram o processamento de linguagem natural com impressionantes capacidades de compreensão e geração de texto. A quantização é apresentada como uma técnica crucial para aprimorar a usabilidade dos modelos em dispositivos limitados. Por fim, as aplicações práticas da adaptação e incorporação de modelos quantizados em dispositivos com recursos limitados e os riscos associados ao uso de LLMs são elencados.

Este capítulo foi realizado com recursos do CNPq, CAPES, RNP e FAPERJ. Ferramentas de Inteligência Artificial Generativa, incluindo ChatGPT, Grammarly e Llama3.1, foram empregadas na revisão textual deste trabalho.

4.1. Introdução

A popularização de Modelos de Linguagem em Larga Escala (*Large Language Models* - LLMs) implica uma profunda transformação no campo do processamento de linguagem natural. Inspirados pelo GPT-3, diversos modelos, como OPT, PaLM, BLOOM, Chinchilla e LLaMA, demonstraram que o aumento na escala do modelo resulta em capacidades de compreensão e geração aprimoradas. Consequentemente, modelos com dezenas, ou até mesmo centenas, de bilhões de parâmetros se tornaram um padrão no cenário atual da inteligência artificial. Yang *et al.* argumentam que a proficiência linguística de LLMs está correlacionada à presença de quatro características principais [Yang et al., 2024]. A primeira delas é a compreensão contextual profunda, que envolve a capacidade de resolver ambiguidades semânticas bem como identificar relações discursivas. A segunda característica é a geração textual fluente, que implica a habilidade de produzir textos gramaticalmente corretos, com vocabulário diverso e não repetitivo, e adequados ao contexto pragmático exigido, considerando diferentes estilos e tons. A terceira característica é a consciência de domínio do conhecimento e abrange a capacidade de realizar inferências a partir de informações implícitas e adaptar o discurso a diferentes níveis de especialização. Por fim, destaca-se a execução direcionada, característica especialmente útil na resolução de problemas complexos e na tomada de decisões a partir de diretrizes formuladas em linguagem natural pelo usuário. Essas habilidades estão intimamente atrelada à engenharia de *prompt*, que se dedica a formular instruções, em linguagem natural, eficazes para elicitar as respostas desejadas do modelo. Essas competências fazem dos LLMs ferramentas fundamentais em diversos campos, incluindo análise de texto, geração de conteúdo e suporte à decisão.

A implementação dos LLMs, entretanto, enfrenta desafios significativos quando realizada em dispositivos móveis, de computação de borda ou com recursos computacionais limitados. Além de questões tradicionais relacionadas à implantação em servidores na nuvem, como latência, segurança e conectividade contínua, a execução de LLMs em dispositivos restritos em recursos computacionais exige soluções que minimizem os altos custos computacionais e a pegada ambiental associada [Xu et al., 2024]. Para abordar esses desafios, técnicas de otimização de uso de memória e técnicas avançadas de quantização têm sido desenvolvidas. Métodos como a quantização pós-treinamento em 8 bits de pesos e ativações [Yao et al., 2024a] permitem reduzir significativamente o consumo de memória e energia sem comprometer a precisão. No entanto, modelos como o LLaMA, com 65 bilhões de parâmetros, continuam a exigir recursos significativos de memória, consumindo até 65 GB de VRAM apenas para armazenar os pesos. Em aplicações de sequência longa, os caches de chave-valor podem alcançar dezenas de *gigabytes*, excedendo os recursos de memória e processamento disponíveis [Zhao et al., 2024].

Diante desse contexto, este capítulo tem como objetivo aprofundar o conhecimento sobre modelos de linguagem em larga escala e suas aplicações em dispositivos com recursos computacionais limitados, como *smartphones* e dispositivos embarcados. O capítulo busca apresentar os fundamentos dos LLMs, incluindo suas arquiteturas baseadas em *Transformers* e *AutoEncoders*, e também explorar os desafios práticos associados à sua implementação em ambientes restritos. Entre esses desafios, destacam-se o alto consumo de memória e energia, que tornam inviável a execução direta de modelos de grande porte em dispositivos móveis ou embarcados. Dessa forma, o capítulo adota uma abor-

dagem que combina teoria e prática, capacitando os leitores a entenderem os principais aspectos técnicos e metodológicos necessários para a adaptação eficiente desses modelos. Um dos focos centrais do capítulo é a otimização de LLMs para execução eficiente em dispositivos móveis. Para tanto, são exploradas diversas técnicas, como quantização, poda e destilação de conhecimento, com ênfase especial na quantização.O capítulo discute o impacto dessas técnicas na eficiência computacional e na viabilidade do uso de LLMs em cenários em que o processamento local é essencial para reduzir dependências de conectividade com a nuvem e melhorar a privacidade dos dados.

O capítulo também aborda estratégias para personalização e adaptação de modelos de linguagem em cenários específicos. Isso inclui a configuração de fluxos de processamento, o ajuste fino (*fine-tuning*) para domínios específicos e a incorporação de técnicas que permitam a execução desses modelos em plataformas como Open WebUI. A atividade prática descrita proporciona aos leitores a oportunidade de experimentar diretamente a quantização e personalização de LLMs, avaliando seu impacto no consumo de recursos e na qualidade das respostas geradas.

O restante do capítulo está organizado da seguinte forma. A Seção 4.2 apresenta os fundamentos do processamento de linguagem natural. A Seção 4.3 discute a arquitetura de modelos de linguagem em larga escala. O processo de treinamento dos modelos e as técnicas de otimização do uso de recursos computacionais são apresentados na Seção 4.4. Classes e exemplos de modelos são elencados na Seção 4.5. As métricas para avaliação de modelos de linguagem em larga escala são apresentadas na Seção 4.6. A atividade prática é descrita na Seção 4.7 e a Seção 4.8 discute desafios e tendências de pesquisa. Por fim, as considerações finais são expostas na Seção 4.9.

4.2. Fundamentos e Evolução da Linguística Computacional

O Processamento de Linguagem Natural (*Natural Language Processing* - NLP), ou linguística computacional, dedica-se ao desenvolvimento de sistemas computacionais para compreender e manipular a linguagem humana. Esse processo requer o uso de *corpora* robustos, coleções estruturadas de textos que sustentam análises em níveis morfológicos, sintáticos, semânticos e pragmáticos. A adequação dos *corpora* à tarefa e o equilíbrio entre a diversidade e especificidade do seu vocabulário influenciam diretamente o aprendizado do modelo, afetando sua capacidade de interpretar ocorrências inéditas ou combinações de padrões gramaticais previamente conhecidos. Independentemente do objetivo, a condução do processamento textual inicia com o pré-processamento dos dados brutos, que podem ser estruturados ou não.

O *pipeline* genérico de pré-processamento dos dados utilizados no treinamento de modelos de linguagem é ilustrado na Figura 4.1. Esse *pipeline* organiza o fluxo de transformação dos dados brutos até sua conversão em representações tokenizadas, apropriadas para uso em etapas seguintes como vetorização e treinamento do modelo de linguagem. Inicialmente, os dados brutos são coletados de diferentes fontes, podendo ser estruturados, semi-estruturados ou não estruturados. Em seguida, ocorre a etapa de **filtragem e seleção**, na qual são aplicadas abordagens que visam garantir a qualidade, relevância e consistência dos dados utilizados. Essa etapa pode incluir (i) uma *Filtragem por idioma*, por meio de modelos de detecção automática, como LangID ou FastText; (ii) uma *Fil-*

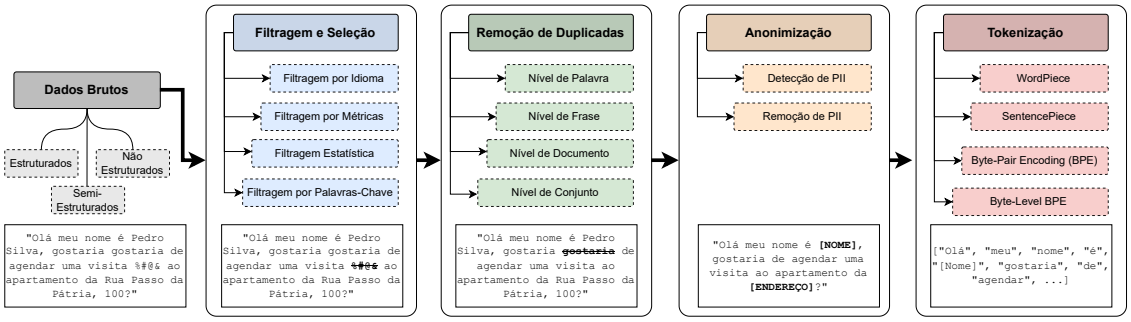


Figura 4.1. *Pipeline* genérico de pré-processamento de dados usados no treinamento de modelos de linguagem em larga escala. Adaptado de [Zhao et al., 2023].

tragem por métricas, com base em legibilidade, coerência ou comprimentos típicos de sequência; (iii) uma *Filtragem estatística*, para remoção de *outliers*, dados ruidosos ou altamente redundantes; ou até (iv) uma *Filtragem por palavras-chave*, útil para garantir que os dados estejam alinhados a domínios de interesse. Na sequência, realiza-se a **remoção de duplicatas**, etapa essencial para mitigar o risco de sobreajuste durante o treinamento dos modelos. A detecção de duplicatas pode ser conduzida em diferentes níveis de granularidade, começando pelo nível de palavra, em que repetições internas são identificadas por comprometerem a naturalidade e diversidade linguística do *corpus*. Em seguida, no nível de frase, são aplicadas técnicas de comparação exata ou medidas de similaridade semântica para detectar redundâncias parciais. No nível de documento, busca-se identificar cópias completas de textos, garantindo que o modelo não aprenda padrões repetidos de maneira artificial. Por fim, a análise no nível de conjunto permite identificar sobreposição entre diferentes subconjuntos de dados, como entre múltiplas fontes de coleta, prevenindo assim a contaminação cruzada e garantindo a diversidade real do conjunto final de treinamento. A etapa seguinte corresponde à **anonimização**, com foco na identificação e eliminação de informações pessoais identificáveis (*Personally Identifiable Information* – PII). Esta tarefa compreende duas subetapas: (i) *detecção de PII*, que pode ser realizada por meio de expressões regulares, modelos de Reconhecimento de Entidades Nomeadas (NER), ou classificadores supervisionados; e (ii) *remoção ou substituição*, geralmente através de máscaras genéricas, de modo a garantir a privacidade e conformidade com regulamentos como a LGPD ou GDPR. Por fim, o pipeline realiza a **tokenização**, processo de segmentação do texto em unidades menores chamadas *tokens*. Essa segmentação pode ocorrer em diferentes níveis, dependendo do modelo ou da tarefa-alvo. Dentre os métodos clássicos incluem: (i) *WordPiece*, utilizado por modelos como BERT, que combina palavras frequentes com subpalavras raras; (ii) *SentencePiece*, amplamente utilizado em modelos treinados em múltiplos idiomas, por sua independência de pré-tokenização; (iii) *Byte-Pair Encoding* (BPE), que substitui pares de caracteres ou subpalavras mais frequentes; (iv) *Byte-Level BPE*, uma extensão do BPE que opera diretamente no nível de bytes, mantendo compatibilidade com textos não normalizados. Esses algoritmos permitem representar texto de forma compacta e eficiente, facilitando o aprendizado de representações semânticas. A preservação de contexto e a modelagem de dependências entre os *tokens* são posteriormente tratadas pelos modelos de linguagem, que capturam essas relações em seus mecanismos internos de atenção ou recorrência.

Modelos de linguagem em larga escala podem ser compreendidos como teorias

Tabela 4.1. Comparativo detalhado dos formatos de precisão numérica nas diferentes gerações de modelos de linguagem.

	SLMs	NLMs	PLMs	LLMs
Abordagem Principal	Regras e/ou Contagem Estatística	Redes Neurais	Transferência de Aprendizado	Transformers em Escala Massiva
Dados de Treinamento	Corpora específicos de tamanho limitado	Corpora de tamanho moderado	Grandes corpora diversificados	Corpora massivos
Representação Semântica	N-grams, Markov Chains	Embeddings densos	Embeddings contextuais	Embeddings contextuais profundos e ricos
Transferência de Aprendizado	Limitada/Inexistente	Limitada	Alta	Muito Alta
Capacidade de Generalização	Baixa	Moderada	Boa	Excelente
Complexidade do Modelo	Baixa	Moderada	Alta	Muito Alta
Requisitos de Hardware	Baixos	Moderados	Altos	Muito Altos
Capacidade de Raciocínio	Praticamente Inexistente	Limitada	Moderada	Avançada
Aplicação Típica	Reconhecimento de Fala, N-grams	Tradução, Geração Simples	Resposta a Perguntas, Análise de Sentimento	Geração de Texto, Sumarização, Chatbots, Geração de Código

linguísticas computacionais, capazes de codificar princípios estruturais não triviais que facilitam tanto a aquisição quanto o processamento da linguagem [Baroni, 2021]. Antes de serem treinados com dados específicos de uma língua, as arquiteturas dessas redes profundas atuam como teorias gerais, definindo um espaço de gramáticas possíveis. Após o treinamento com dados de uma língua específica, os LLMs se comportam como uma gramática funcional, isto é, sistemas computacionais capazes de, dado um enunciado em uma língua, prever se a sequência é aceitável para um falante idealizado dessa língua.

A evolução dos modelos de linguagem demonstra uma sucessão de paradigmas, representados por quatro categorias principais, cada qual com abordagens, técnicas e capacidades específicas para o processamento de dados linguísticos. A Tabela 4.1 compara e resume as características das diversas gerações de modelos de linguagem. Sendo a primeira geração de modelos de linguagem, os **Modelos Estatísticos de Linguagem** (*Statistical Language Models* - SLMs) basearam seu desenvolvimento em métodos de aprendizado estatístico. Estes modelos utilizam a suposição de Markov para prever palavras em uma sequência, conhecida como abordagem de modelos *n-gram*. Dessa forma, os SLMs preveem uma palavra com base nas n palavras imediatamente anteriores, mas enfrentam desafios significativos devido à escassez de dados, especialmente ao tentar estimar modelos de alta ordem. Para mitigar esse problema, técnicas de suavização, como a suavização aditiva e as estimativas de *Good-Turing* e *backoff*, são largamente empregadas. Tais estratégias focam na redistribuição das probabilidades calculadas diretamente das frequências observadas no *corpus*, garantindo que nenhuma probabilidade seja zero e ajustando as probabilidades baixas para valores mais realistas. Embora eficazes em tarefas básicas, como correção ortográfica, os SLMs enfrentam dificuldades ao lidar com dependências de longo alcance e são limitados pelo crescimento exponencial das probabilidades de transição necessárias para modelos de ordem superior.

Os **Modelos de Linguagem Neural** (*Neural Language Models* - NLMs) representam um avanço significativo em relação à primeira geração de modelos, sobretudo na superação das limitações na captura de relações semânticas e dependências de longo

alcance. Esse progresso é amplamente atribuído ao uso de técnicas de incorporação de palavras (*word embeddings*), que são representações distribuídas em que palavras são mapeadas para vetores densos em um espaço contínuo. Essas representações permitem que os NLMs identifiquem semelhanças entre palavras com base em seu contexto de uso, ampliando sua capacidade de generalização e a compreensão semântica. Arquiteturas de NLMs como Redes Neurais Recorrentes (*Recurrent Neural Networks* - RNNs) e suas variantes mais avançadas, como LSTMs (*Long Short-Term Memory*) e GRUs (*Gated Recurrent Unit*), são projetadas para lidar com dados sequenciais, o que permite que o modelo retenha informações ao longo de uma sequência de palavras. No entanto, os NLMs demandam grande volume de dados para treinamento eficaz e apresentam alto custo computacional, tanto em treinamento quanto em inferência.

Posteriormente, os **Modelos de Linguagem Pré-treinados** (*Pre-trained Language Models* - PLMs) introduziram uma abordagem inovadora baseada no pré-treinamento em *corpora* extensos não rotulados, seguido pelo ajuste fino (*fine-tuning*) para tarefas específicas. Sendo o mais proeminente da geração, o modelo BERT (*Bidirectional Encoder Representations from Transformers*) adaptou a então recente arquitetura *Transformer* e a combinou com um mecanismo de autoatenção para oferecer representações contextuais bidirecionais robustas. Esse modelo elevou significativamente o desempenho em tarefas como geração de diálogo e detecção de emoções em textos. A eficácia do BERT, juntamente com modelos subsequentes, catalisou o desenvolvimento de PLMs com arquiteturas otimizadas e estratégias de pré-treinamento mais sofisticadas, como o GPT e o RoBERTa.

O surgimento dos **Modelos de Linguagem em Larga Escala** (*Large Language Models* - LLMs) resulta da aplicação prática do ganho de escala, aprimorando arquiteturas e métodos de treinamento herdados dos PLMs. Explora-se a correlação entre o aumento do tamanho do modelo, dado pelo número de parâmetros, e o volume de dados de treinamento, refletindo em melhorias abruptas de desempenho em tarefas generativas e compreensivas. Comparativamente, enquanto PLMs detêm um treinamento pautado em *corpora* compostos por centenas de milhões até poucos bilhões de *tokens*, os LLMs como o GPT-3 alcançam centenas de bilhões de *tokens*. Além das aplicações tradicionais de PLN, como análise de sentimento, tradução e sumarização automática multilíngue, os LLMs têm atuado de forma semiautomática em tarefas que antes exigiam alta supervisão humana. Entre essas tarefas, destacam-se: (i) a didática, auxiliando na prática conversacional em ambientes de aprendizado de idiomas; (ii) a extração e correlação de conhecimentos complexos estruturados em textos acadêmicos [Dagdelen et al., 2024]; (iii) a segurança digital, detectando indícios de desinformação em textos ou *links* compartilhados; e (iv) o gerenciamento de redes, atuando como assistentes conversacionais inteligentes na interpretação de intenções do operador [de Oliveira et al., 2025].

4.3. Estrutura e Funcionamento de Modelos de Linguagem em Larga Escala

Os LLMs utilizam uma arquitetura que processa dados textuais provenientes de múltiplas fontes. Após a etapa inicial de pré-processamento, o treinamento envolve estágios como inicialização aleatória de parâmetros, cálculo da função de perda, otimização iterativa e ajustes dos parâmetros. Esses modelos oferecem serviços como tradução de texto, sumarização, análise de sentimentos e aplicações específicas em domínios como

ciências médicas e política. Com o surgimento de modelos avançados como GPT, LLaMa e Bard, além de variantes como Alpaca e GPT-Huggingface, os LLMs consolidaram-se como ferramentas essenciais para tarefas de NLP [Raiaan et al., 2024].

A arquitetura *Transformer* é a base dos LLMs devido à sua eficiência no processamento de sequências textuais. Diferentemente de métodos tradicionais baseados em iteração, o *Transformer* emprega mecanismos de atenção para identificar relações contextuais entre diferentes partes do texto. Essa abordagem permite o processamento de sequências de tamanhos variados e garante maior flexibilidade, tornando-se a arquitetura predominante em tarefas de PLN [Vaswani et al., 2017]. Os componentes da arquitetura *Transformer*, elencados a seguir, desempenham papéis específicos e integrados em seu funcionamento. O texto de entrada é dividido em *tokens*, que podem ser palavras ou subpalavras, e transformado em representações numéricas chamadas incorporações ou *embeddings*, que capturam similaridades semânticas entre palavras. Para garantir que a ordem das palavras seja considerada, informações posicionais são incorporadas a esses *embeddings*. O codificador (*encoder*) processa a entrada e gera representações ocultas que capturam o significado contextual do texto, enquanto o decodificador (*decoder*) utiliza essas informações para prever a próxima palavra em uma sequência, baseando-se no contexto anterior. Durante esse processo, os *embeddings* de saída são convertidos novamente em texto, preservando as informações posicionais. Por fim, a camada linear transforma as saídas do decodificador em um espaço de maior dimensão e a função *softmax* gera distribuições probabilísticas, implicando previsões com maior precisão e robustez.

Os mecanismos de atenção desempenham um papel essencial no desempenho dos LLMs permitindo identificar relações contextuais entre *tokens*. O mecanismo de Autoatenção (*Self-Attention*) estabelece conexões dentro do mesmo bloco codificador-decodificador, analisando como diferentes partes do texto se relacionam. O mecanismo de Atenção Total (*Full Attention*) é uma implementação direta e completa da autoatenção, avaliando todas as possíveis interações entre *tokens*. Já a Atenção Cruzada (*Cross Attention*) utiliza as representações geradas pelo codificador como consultas para o decodificador, permitindo integrar informações contextuais mais amplas ao processo de geração de texto [Zheng et al., 2025, Lin et al., 2022]. As melhorias no mecanismo de atenção são organizadas em várias abordagens que visam aprimorar sua eficiência e flexibilidade. A Atenção Esparsa (*Sparse Attention*) reduz a complexidade computacional ao introduzir esparsidade no cálculo das interações entre *tokens*, enquanto a Atenção Linearizada (*Linearized Attention*) simplifica os cálculos utilizando mapas de características de *kernel*. Métodos de compressão de Protótipo e Memória (*Prototype and Memory Compression*) otimizam o uso de memória ao reduzir o número de pares de consulta chave-valor. A Autoatenção de Baixa-Patente (*Low-rank Self-Attention*) explora propriedades de baixa patente (*rank*) para tornar o processamento mais eficiente. Já a Atenção com Precedência (*Attention with Prior*) complementa ou substitui a atenção padrão ao incorporar distribuições prévias. Por fim, o mecanismo Multi-Cabeça Aprimorado (*Improved Multi-Head Mechanism*) introduz o conceito de multi-cabeça referindo-se a uma projeção de atenção que foca em diferentes partes da entrada. O uso de múltiplas cabeças permite que o modelo capture diversas representações simultaneamente, enriquecendo a capacidade de aprendizado e aumentando a eficiência computacional.

4.3.1. Aprendizado Baseado em *Transformers*

A arquitetura do modelo *Transformer* é composta por vários componentes interligados que permitem o processamento eficiente de sequências de dados, como textos. Sua principal inovação reside no uso do mecanismo de atenção, que substitui métodos tradicionais sequenciais, possibilitando o processamento paralelo e a captura de relacionamentos de longo alcance no texto. A entrada do modelo começa com a tokenização do texto. Como o Transformer não processa os *tokens* sequencialmente, é necessário fornecer informações sobre a posição de cada *token* na sequência e isso é feito por meio de codificações posicionais adicionadas ao *embedding*, normalmente utilizando funções trigonométricas que representam a posição de cada *token* no texto.

A arquitetura principal se divide em duas partes: o **codificador** (*encoder*) e o **decodificador** (*decoder*), quando a tarefa exige geração de texto. O **codificador** é composto por várias camadas idênticas, geralmente entre seis e doze, em que cada camada contém uma atenção auto-regressiva (auto-atenção) e uma rede neural *feedforward* totalmente conectada. A atenção auto-regressiva permite que o modelo avalie a importância de cada *token* em relação aos demais, dando foco às partes mais relevantes do texto para formar uma compreensão contextualizada. Essa camada de atenção calcula as inter-relações entre todos os *tokens* simultaneamente, o que permite ao modelo captar relacionamentos de longo alcance de forma eficaz. Em seguida, a saída passa por uma camada de processamento *feedforward*, que ajusta e combina as informações extraídas. Cada camada é seguida por operações de normalização e mecanismos de *dropout*, auxiliando na estabilidade do treinamento.

Quando o *Transformer* inclui um decodificador seu papel é gerar a saída, como uma tradução ou uma continuação de uma frase. O **decodificador** também é formado por múltiplas camadas, contendo uma atenção mascarada que garante que, na geração de cada *token*, o modelo só leve em consideração os *tokens* anteriores, preservando a causalidade. Há também uma atenção que integra as representações extraídas pelo codificador, permitindo que o decodificador alinhe a entrada com a saída de forma eficiente durante a geração do texto. Essa arquitetura possibilita ao modelo entender o contexto de toda a sequência ao mesmo tempo, ao contrário das abordagens sequenciais, como redes recorrentes, aumentando muito a velocidade de treinamento e aprimorando o desempenho.

Por fim, após o processamento nas camadas de atenção e *feedforward*, as saídas passam por uma camada linear que ajusta a dimensionalidade, seguida por uma função *softmax* que fornece as probabilidades para os próximos *tokens* ou para tarefas de classificação. Dessa forma, a arquitetura do *Transformer* consegue representar o conteúdo de textos de maneira profunda e contextualizada, o que revolucionou o campo do processamento de linguagem natural ao possibilitar o treinamento de modelos cada vez maiores e mais precisos.

A Figura 4.2 representa a arquitetura do modelo *Transformer*, mostrando o fluxo de dados desde a entrada textual até a geração da saída. O texto é inicialmente convertido em vetores por meio de incorporações e codificações posicionais. Esses vetores passam por blocos repetidos de codificador, compostos por mecanismos de auto-atenção e redes *feedforward*, que capturam relações contextuais entre as palavras. Em seguida, o decodificador, também com camadas repetidas, utiliza atenção mascarada e cruzada para gerar

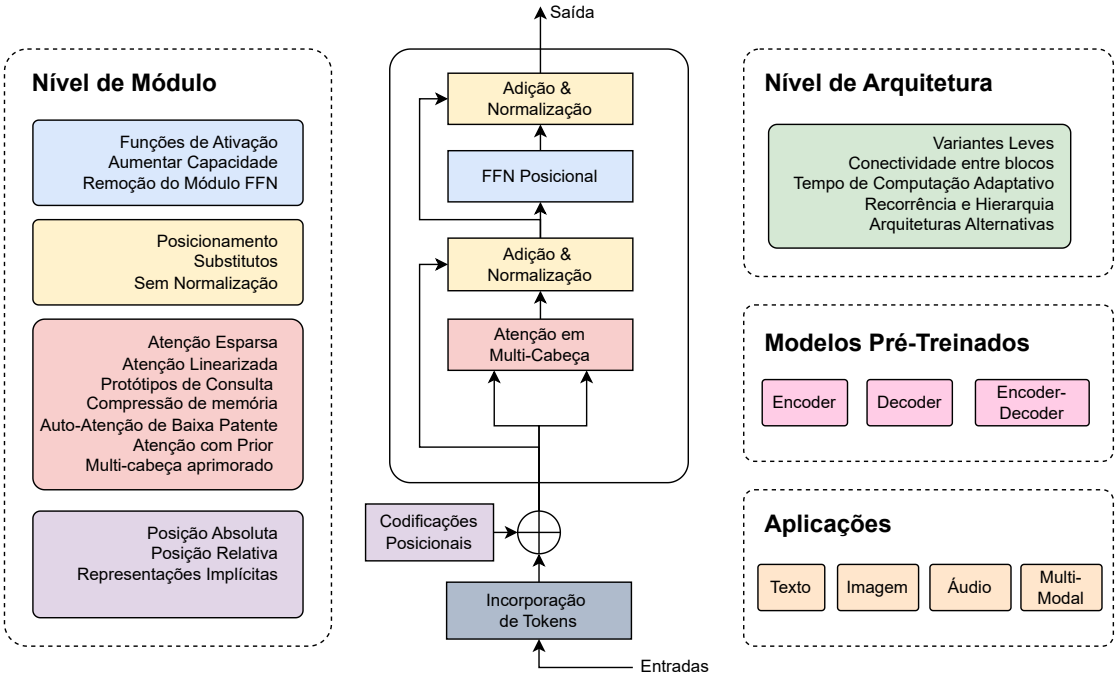


Figura 4.2. Arquitetura básica de um modelo *Transformer* com variações estruturais em nível de módulo e arquitetura, para diferentes aplicações e com tipos de modelos pré-treinados distintos.

a sequência de saída. Por fim, os vetores passam por uma camada linear e uma função *softmax* para prever a próxima palavra com base em probabilidades.

4.3.2. Variações da Arquitetura Principal do Modelo *Transformer*

As variantes principais do modelo *Transformer* são os modelos *Encoder*, modelos *Decoder* e modelos *Encoder-Decoder*, conforme mostra a Figura 4.2 nos modelos pré-treinados. Modelos do tipo ***Encoder*** utilizam apenas a parte codificadora do *Transformer*, permitindo que as camadas de atenção acessem todas as palavras da frase simultaneamente, o que confere uma atenção bidirecional. Eles são pré-treinados corrompendo sentenças, por exemplo, mascarando palavras aleatórias, e ensinados a reconstruí-las. Esses modelos são ideais para tarefas que exigem compreensão completa da sentença, como classificação de sentenças, reconhecimento de entidades nomeadas e respostas a perguntas de forma extrativa. Representantes dessa família de modelos são o BERT [Aftan e Shah, 2023] e o DistilBERT [Sanh et al., 2020].

Modelos do tipo ***Decoder*** utilizam apenas a parte decodificadora do *Transformer*, em que a atenção é unidirecional, permitindo que cada palavra acesse apenas as anteriores na sentença. Chamados de modelos auto-regressivos, são pré-treinados para prever a próxima palavra em uma sequência. Eles são mais indicados para tarefas de geração de texto. Representantes dessa classe são os modelos LLaMa [Dubey et al., 2024], Gemma 3 [Jadouli e Amrani, 2025] e DeepSeek V3 [Guo et al., 2025].

Modelos **Encoder-Decoder**, ou de sequência para sequência, utilizam tanto o codificador quanto o decodificador do *Transformer*. O codificador acessa toda a sentença de entrada, enquanto o decodificador só acessa as palavras anteriores a cada etapa. Seu

pré-treinamento varia, mas geralmente envolve a reconstrução de sentenças corrompidas, como no modelo T5 [Raffel et al., 2020], que substitui trechos por um *token* especial e prevê o texto correspondente. Esses modelos são ideais para tarefas que exigem gerar novas sentenças a partir de uma entrada, como sumarização, tradução e respostas generativas a perguntas. Os representantes dessa família são os modelos BART [Lewis et al., 2019] e T5 [Raffel et al., 2020].

4.3.3. Mecanismos de Auto-Atenção

O mecanismo de auto-atenção é fundamental no modelo *Transformer* porque permite que as representações de entrada e saída sejam altamente contextuais e dinâmicas, sem depender de estruturas sequenciais rígidas como redes neurais recorrentes (*Recurrent Neural Network* - RNNs) ou redes neurais convolucionais (*Convolutional Neural Network* - CNNs). Ao usar atenção, o modelo consegue identificar quais partes da sequência são mais relevantes para cada elemento, facilitando a captura de dependências de longo alcance e relações globais entre palavras ou *tokens*. Essa habilidade de focar diferentes partes da entrada simultaneamente torna o processamento mais eficiente e paralelo, além de melhorar a capacidade de modelar relações complexas em tarefas como tradução, resumo e compreensão de texto, alcançando resultados de ponta em desempenho. Ademais, o mecanismo de atenção melhora a escalabilidade do modelo e permite maior paralelização durante o treinamento, o que reduz consideravelmente o tempo necessário para ajustar os parâmetros. Isso, combinado com sua capacidade de modelar dependências de longo alcance de forma mais direta, faz do mecanismo de atenção o componente central que viabiliza a eficácia do *Transformer* e a sua superioridade em muitas tarefas de processamento de linguagem natural. Dentre os mecanismos de atenção utilizados nesse modelo, destacam-se a atenção por produto escalar e a atenção em multi-cabeças.

A **Atenção por Produto Escalar** (*Scaled Dot-Product Attention*) é um mecanismo central no modelo *Transformer* que calcula a atenção com base na similaridade entre vetores de consulta e vetores de chave, associando-os a vetores de valor [Vaswani et al., 2017]. O processo começa com o cálculo do *produto escalar* entre cada consulta e todas as chaves para medir a compatibilidade. Esse valor é então normalizado dividindo por $\sqrt{d_k}$, em que d_k é a dimensão dos vetores de chave. Essa normalização evita que valores muito grandes empurrem a função *softmax* para regiões com gradientes muito pequenos. Após a normalização, aplica-se a função *softmax* para gerar uma distribuição de probabilidade, os pesos, que é usada para calcular uma soma ponderada dos valores. A equação da atenção escalada por produto escalar é dada por

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

em que Q representa as consultas, K as chaves e V os valores. O termo $QK^\top$ mede a compatibilidade entre consultas e chaves, sendo escalado por $\sqrt{d_k}$ para estabilizar os valores. A função softmax transforma esses escores em pesos, e a multiplicação final por V gera uma saída ponderada com base nas relações entre os elementos da sequência.

Na prática, as consultas, chaves e valores são organizadas em matrizes $Q \in \mathbb{R}^{n\times d_k}$, $K \in \mathbb{R}^{m\times d_k}$ e $V \in \mathbb{R}^{m\times d_v}$, permitindo computações em paralelo e de forma eficiente. Em

comparação, a *atenção aditiva*, proposta por [Bahdanau et al., 2014], calcula a função de compatibilidade com uma rede neural com uma camada oculta dada por

$$\text{AdditiveAttention}(q,k) = v^\top \tanh(W_1 q + W_2 k).$$

Embora ambas as abordagens tenham complexidade teórica semelhante, a atenção por produto escalar é mais eficiente na prática, pois aproveita multiplicações de matrizes otimizadas. No entanto, sem a escala por $\sqrt{d_k}$, seu desempenho tende a se degradar quando d_k é grande, pois os valores do produto escalar se tornam grandes em magnitude, o que empurra a função *softmax* para regiões com gradientes extremamente pequenos, dificultando o treinamento.

O mecanismo de **Atenção em Multi-Cabeças** (*Multi-Head Attention*) estende a atenção tradicional ao projetar linearmente as consultas, chaves e valores, múltiplas vezes com diferentes matrizes aprendidas [Vaswani et al., 2017]. Em vez de calcular a atenção diretamente com vetores na dimensão total d_{model}, o modelo cria h cabeças de atenção, cada uma operando com vetores de dimensão reduzida d_k e d_v. Para cada cabeça i, aplica-se a atenção escalada por produto escalar da seguinte forma:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) = \text{softmax}\left(\frac{QW_i^Q (KW_i^K)^\top}{\sqrt{d_k}}\right) VW_i^V,$$

em que Q, K e V são projetados por matrizes aprendidas W_i^Q, W_i^K e W_i^V. As pontuações de atenção são calculadas via produto escalar, escaladas por $\sqrt{d_k}$, normalizadas com *softmax* e usadas para ponderar os valores projetados VW_i^V, produzindo a saída da cabeça i. Cada saída $\text{head}_i \in \mathbb{R}^{T \times d_v}$, em que T é o número de posições da sequência, é então concatenada com as demais e projetada de volta para a dimensão original, da seguinte forma:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O.$$

Assim, permite-se que o modelo atenda simultaneamente a diferentes subespaços de representação em diferentes posições da entrada, algo que uma única cabeça de atenção não conseguiria capturar de forma eficaz. No *Transformer* original, utilizam-se $h = 8$ cabeças, com $d_k = d_v = \frac{d_{\text{model}}}{h} = 64$, o que mantém o custo computacional semelhante ao da atenção com dimensão total. O uso de múltiplas cabeças melhora significativamente a capacidade do modelo de capturar padrões diversos e complexos, sendo um dos pilares do sucesso dos *Transformers*.

4.3.4. Modelos *Transformers* Eficientes

Os *Transformers* tradicionais apresentam uma complexidade computacional e de memória quadrática ($O(N^2)$) [Tay et al., 2022]. Essa complexidade se origina no mecanismo de autoatenção, que exige que cada *token* calcule sua relação com todos os demais *tokens* em uma sequência. Essa característica torna o uso desses modelos extremamente custosos em cenários que envolvem sequências longas, como grandes documentos, vídeos extensos e imagens de alta resolução.

Para mitigar esses problemas, surgem os chamados *Transformers eficientes*, que utilizam técnicas que reduzem significativamente a complexidade, tornando-os mais escaláveis para diversas aplicações práticas. Exemplos de Transformers Eficientes são citados a seguir [Tay et al., 2022]. O **Linformer** reduz a complexidade quadrática da atenção para uma complexidade linear ($O(N)$) ao utilizar aproximações de baixa patente (*low-rank*). Essa técnica consiste em projetar as matrizes chave e valor em dimensões menores, reduzindo significativamente os custos computacionais, sem comprometer excessivamente a performance. O *Linformer* é especialmente útil em contextos que demandam o processamento de textos extensos. O **Performer** utiliza a técnica de *kernelização* para reformular o mecanismo de autoatenção, eliminando a necessidade explícita de calcular a matriz completa de atenção. Assim, o *Performer* reduz a complexidade para linear ($O(N)$), ideal para aplicações que requerem inferência rápida e uso eficiente de recursos computacionais limitados. O **Longformer** introduz a combinação de atenção local e *tokens* globais específicos, permitindo uma atenção eficiente e quase-linear ($O(N)$). Essa abordagem é particularmente eficaz em aplicações que envolvem processamento e análise de documentos muito longos, como relatórios técnicos, textos jurídicos e artigos científicos. Os **Sparse Transformers** são modelos que utilizam padrões esparsos pré-definidos, como combinações de atenção local e atenção dilatada (*strided attention*). O resultado é uma redução da complexidade para aproximadamente $O(N\sqrt{N})$. Os *Sparse Transformers* são eficazes especialmente na modelagem de imagens e vídeos, capturando simultaneamente detalhes locais e informações contextuais de longo alcance.

Embora muitos mecanismos eficientes de atenção utilizem padrões fixos para reduzir a complexidade computacional, algumas abordagens inovadoras introduzem **padrões aprendidos**. Esses modelos definem, de maneira dinâmica e adaptativa, quais *tokens* irão interagir no cálculo da atenção, oferecendo flexibilidade significativa ao modelo. Dois exemplos notáveis dessa categoria são o *Routing Transformer* e o *Reformer*. O *Routing Transformer* utiliza técnicas de **clusterização dinâmica**, como o algoritmo *k-means*, para organizar *tokens* em grupos similares durante o processamento da sequência. Em cada camada do modelo, os *tokens* são agrupados dinamicamente, permitindo que a atenção seja calculada somente entre *tokens* pertencentes ao mesmo *cluster*. Dessa forma, o *Routing Transformer* consegue reduzir significativamente a complexidade computacional, pois cada *token* não precisa interagir com todos os outros, mas somente com *tokens* semanticamente relacionados. Essa abordagem reduz a complexidade computacional para aproximadamente $O(N\sqrt{N})$, ao mesmo tempo em que oferece capacidade adaptativa de capturar relações contextuais relevantes. Além disso, contribui para a melhoria na eficiência do modelo em sequências muito longas, onde as relações mais importantes são agrupadas automaticamente.

O *Reformer* introduz um mecanismo conhecido como **Locality-Sensitive Hashing (LSH)**, ou *hashing sensível à localidade*. O LSH é uma técnica que permite agrupar *tokens* semelhantes rapidamente em *buckets* (grupos) com alta probabilidade. A atenção é, então, calculada apenas dentro desses grupos, limitando o alcance da atenção e reduzindo drasticamente o custo computacional e o consumo de memória. Especificamente, o mecanismo do *Reformer* consiste em aplicar funções de *hashing* que preservam a proximidade (*localidade*), agrupando automaticamente *tokens* próximos ou similares em representações. Em seguida, a atenção é computada somente dentro desses grupos,

o que reduz a complexidade para $O(N \log N)$. Essa técnica é utilizada de forma eficiente tanto durante o treinamento quanto na inferência, aproveitando a capacidade adaptativa da abordagem. O *Reformer* é particularmente adequado para contextos que possuem alta redundância, como documentos textuais longos ou registros de *logs*.

4.4. Treinamento e Otimização dos Modelos de Linguagem em Larga Escala

O treinamento dos Modelos de Linguagem em Larga Escala pode ser organizado nas etapas de pré-treinamento, ajuste fino e refinamento através do aprendizado por reforço com *feedback* humano. Na etapa de pré-treinamento, o modelo é treinado de forma auto-supervisionada em um grande *corpus* para prever os próximos *tokens* a partir da entrada. As escolhas de arquitetura dos LLMs variam entre modelos *Encoder-Decoder* e modelos somente com *Eecoder*, utilizando diferentes blocos de construção e funções de perda [Naveed et al., 2023]. Na etapa de ajuste fino (*fine-tuning*), existem diferentes métodos para LLMs e cada um visa adaptar o modelo pré-treinado a objetivos específicos. O aprendizado por transferência (*transfer learning*) consiste em refinar o modelo utilizando dados específicos de uma tarefa-alvo. Apesar de LLMs já apresentarem bom desempenho geral, o ajuste com dados de tarefas específicas melhora ainda mais seus resultados em aplicações concretas. O **instruction-tuning** tem como objetivo ensinar o modelo a seguir instruções escritas em linguagem natural. Isso é feito ajustando o modelo com dados compostos por uma instrução e um par entrada-saída correspondente [Chung et al., 2024]. Esse processo é eficaz para melhorar a capacidade de generalização do modelo em tarefas não vistas anteriormente (*zero-shot*), além de torná-lo mais responsivo a *prompts* fornecidos por usuários. Por fim, o ajuste por alinhamento (**alignment-tuning**) trata da segurança e ética no comportamento do modelo. Como LLMs podem gerar respostas incorretas, tendenciosas ou prejudiciais, o ajuste por alinhamento usa *feedback* humano para ensinar o modelo a evitar essas saídas problemáticas [Ouyang et al., 2022]. Isso é feito expondo o modelo a exemplos de respostas inadequadas e ajustando seus parâmetros para torná-lo mais útil, honesto e inofensivo.

O alinhamento de LLMs para o comportamento desejado é frequentemente feito por meio de aprendizado por reforço com *feedback* humano (*Reinforcement Learning with Human Feedback* - RLHF) [Ouyang et al., 2022]. Nesse processo, o modelo é inicialmente ajustado com demonstrações humanas e, em seguida, passa por etapas adicionais de treinamento com modelagem de recompensas e aprendizado por reforço (*Reinforcement Learning* - RL). Na etapa de modelagem de recompensas (*Reward Modeling* - RM), um modelo é treinado para classificar ou ranquear respostas geradas de acordo com preferências humanas. Para isso, pessoas avaliam as respostas com base nos critérios Útil, Honesto e Inofensivo (*Helpful, Honest, Harmless* - HHH), e essas anotações servem de base para ensinar o modelo a reconhecer respostas melhores ou piores. Por fim, na etapa de aprendizado por reforço, o modelo usa as classificações fornecidas pelo modelo de recompensa para ajustar seu comportamento. Utilizando algoritmos como a otimização de política proximal (*Proximal Policy Optimization* - PPO), o modelo é otimizado para gerar respostas preferidas. Esse processo é iterativo e continua até que o modelo atinja desempenho estável e alinhado com os objetivos desejados.

Contudo, a inferência e o treinamento desses modelos exigem alta capacidade de processamento, grande volume de memória e um consumo energético considerável,

Tabela 4.2. Comparativo detalhado dos formatos de precisão numérica em LLMs.

	FP32	**FP16**	**FP8** (E4M3)	**FP8** (E5M2)	**INT8**	**INT4**
Bits Totais	32	16	8	8	8	4
Bits Sinal	1	1	1	1	N/A	N/A
Bits Expoente	8	5	4	5	N/A	N/A
Bits Mantissa	23	10	3	2	N/A	N/A
Tipo Principal	Ponto Flutuante	Ponto Flutuante	Ponto Flutuante	Ponto Flutuante	Inteiro	Inteiro
Alcance Dinâmico	Muito Amplo	Amplo	Moderado	Amplo	Limitado	Muito Limitado
Precisão	Alta	Moderada	Baixa	Muito Baixa	Baixa	Muito Baixa
Uso de Memória	Muito Alto	Alto	Médio	Médio	Baixo	Muito Baixo

tornando sua implementação direta inviável em ambientes restritos. Para mitigar essas barreiras, estratégias são desenvolvidas em duas frentes [Loukas et al., 2023, Xu et al., 2024]: a primeira visa a otimização da arquitetura e da distribuição da carga, por meio de modelos colaborativos e hierárquicos que fragmentam a demanda computacional entre diferentes componentes do sistema; a segunda foca a minimização do custo operacional intrínseco aos LLMs, através da redução da complexidade de suas matrizes internas, mantendo seu desempenho. Nesse contexto, quantização, poda e destilação de conhecimento são três técnicas principais de compressão de modelos [Ouyang et al., 2022]. Essas técnicas e suas variantes buscam aprimorar a eficiência operacional dos LLMs, garantindo sua viabilidade para uma gama expandida de aplicações ao equilibrar desempenho, consumo de memória e velocidade de inferência.

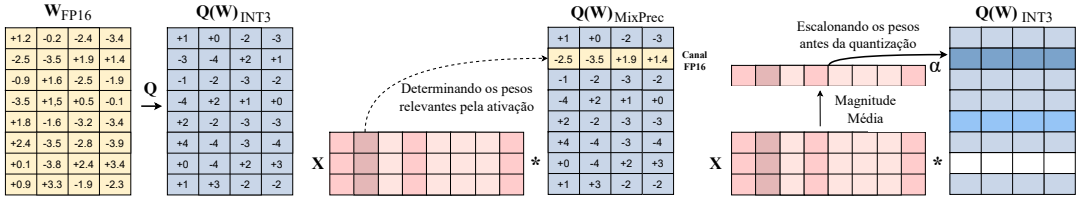
4.4.1. Quantização

Essencialmente, a quantização é um procedimento de mapeamento que visa reduzir a granularidade da representação numérica dos dados. No contexto de redes neurais, essa técnica se traduz na transformação de pesos e ativações, tipicamente representados em formato de ponto flutuante de alta precisão, para representações com menor largura de *bits*. No sistema binário, a representação de um número de ponto flutuante, fundamentada na base 2, é formalmente expressa como $N = \text{Sinal} \times \text{Mantissa} \times 2^{\text{Expoente}}$. Nesta formalização, o *bit* de **Sinal** é convencionalmente alocado como 1 *bit*, indicando a polaridade do número, sendo 0 para valores positivos e 1 para negativos. O **Expoente** é o componente que estabelece a ordem de magnitude do valor, determinando o deslocamento do ponto binário e, consequentemente, ampliando o alcance dinâmico. A alocação de um número maior de *bits* para o expoente resulta diretamente na capacidade de representar um intervalo significativamente mais amplo de valores. Por fim, a **Mantissa** encapsula os dígitos significativos do número, determinando intrinsecamente a precisão ou granularidade da representação numérica. Uma maior quantidade de bits dedicados à mantissa possibilita a representação de números com uma fidelidade superior ao valor real, permitindo a inclusão de um número elevado de casas decimais. A Tabela 4.2 sumariza as características dos principais tipos de formatos de precisão utilizados em LLMs.

Consequentemente, a quantização provê uma redução expressiva no tamanho do

modelo e nas demandas computacionais, resultando em uma inferência significativamente mais rápida e menor consumo de memória [Xu et al., 2024]. Três estratégias comumente utilizadas para quantização são mostradas na Figura 4.3: *Round-to-Nearest*, Quantização de Precisão Mista e Quantização de Pesos e Ativações. Na primeira estratégia, os pesos são arredondados uniformemente para INT3, reduzindo a precisão e resultando em alta perplexidade. Na segunda, os pesos mais relevantes são mantidos em FP16, definidos com base nas ativações de entrada, melhorando a perplexidade, mas introduzindo ineficiências de hardware. Na terceira, aplica-se um escalonamento adaptativo antes da quantização, determinado pela magnitude média das ativações por coluna, permitindo preservar a acurácia do modelo sem comprometer a eficiência computacional. Existem duas abordagens principais para implementar a quantização, cada uma com suas características e benefícios específicos.

A primeira abordagem, conhecida como **Treinamento com Reconhecimento de Quantização**[2] (*Quantization-Aware Training* – QAT) incorpora as operações de quantização diretamente no processo de treinamento, permitindo que o modelo ajuste seus parâmetros considerando restrições de baixa precisão numérica previamente estabelecidas. Essa integração tende a resultar em melhor acurácia após a quantização, uma vez que o modelo aprende a mitigar ativamente os efeitos adversos introduzidos pela representação de menor precisão. Todavia, a aplicação de QAT em modelos de linguagem de grande escala impõe dois desafios significativos, a preservação da generalização e a reprodutibilidade do treinamento original [Liu et al., 2023]. O primeiro diz respeito à manutenção da capacidade *zero-shot*, característica central dos LLMs. Para tanto, é aconselhável que o conjunto de dados empregado no ajuste fino reflita a distribuição observada no pré-treinamento, evitando desvios que prejudiquem o desempenho do modelo quantizado. O segundo desafio está na dificuldade de replicar o processo original de treinamento, devido à escala computacional envolvida e à complexidade dos pipelines utilizados.



(A) Quantização *Round-To-Nearest* (B) Quantização de Precisão Mista (C) Quantização de Pesos e Ativações

Figura 4.3. Comparação entre três estratégias de quantização para modelos de linguagem. (*A*) A Quantização *Round-To-Nearest* reduz a precisão e resulta em alta perplexidade. (*B*) A Quantização de Precisão Mista melhora significativamente a perplexidade, mas introduz ineficiências de *hardware*. (*C*) A Quantização de Pesos e Ativaçoes permite preservar a acurácia do modelo sem comprometer a eficiência computacional, operando exclusivamente com pesos no formato INT3.

A segunda abordagem, denominada **Quantização Pós-Treinamento** (*Post-Training Quantization* - PTQ) consiste na aplicação de técnicas de quantização após o término do treinamento do modelo, sem necessidade de ajuste fino. Por não exigir retropropagação, essa abordagem é computacionalmente mais eficiente que o QAT e apresenta

[2]Disponível em `https://github.com/facebookresearch/LLM-QAT`.

menor custo de implementação, sendo particularmente atrativa em cenários com recursos limitados. No contexto de PTQ, destacam-se duas variantes principais:

- **Quantização Apenas de Pesos** (*Weight-only Quantization* – WoQ). Nessa configuração, apenas os pesos da rede neural são quantizados, enquanto as ativações permanecem em precisão total, geralmente FP32. Trata-se de uma técnica mais simples e com menor impacto sobre o desempenho do modelo quando as ativações apresentam distribuição estável e baixo dinamismo. Essa abordagem é apropriada para reduzir o uso de memória e simplificar a execução, especialmente em dispositivos com suporte limitado a quantização de ativação.
- **Quantização de Pesos e Ativações** (*Activation-aware Weight Quantization* – AWQ). Introduzida por Lin *et al.*, essa variante otimiza o processo de quantização ao estender sua aplicação tanto a pesos quanto a ativações, elevando a redução na complexidade computacional [Lin et al., 2024]. Como visto na Figura 4.3, a premissa central do AWQ reside na observação empírica de que a importância dos pesos para o desempenho de LLMs é heterogênea. Uma fração mínima, tipicamente entre 0,1% e 1%, dos pesos é considerada relevante. Ao desconsiderar seletivamente a quantização desses pesos críticos, o AWQ mitiga substancialmente a perda de precisão induzida pela quantização. A identificação desses canais de peso salientes é guiada pela distribuição das ativações, em vez da distribuição dos próprios pesos. Canais de peso que correspondem a ativações de maior magnitude são inferidos como mais relevantes, dado seu papel no processamento de características mais representativas. Visando a eficiência em implementações de hardware e para evitar a complexidade da precisão mista, a metodologia AWQ analisa o erro de quantização do peso e propõe que o escalonamento desses canais salientes pode reduzir seu erro de quantização relativo. Esse método é particularmente vantajoso para arquiteturas de hardware, como GPUs e ASICs, devido à otimização da multiplicação de matrizes, operação crucial em computações neurais.

4.4.2. Mecanismos Avançados de Atenção para *Transformers* Eficientes

O mecanismo tradicional de autoatenção nos *Transformers* requer o cálculo das relações entre todos os pares de *tokens* em uma sequência, resultando em complexidade quadrática ($O(N^2)$). Para superar esse desafio, foi proposto o conceito de **atenção esparsa**, que restringe o cálculo da atenção apenas a um subconjunto limitado e predefinido ou aprendido de pares de *tokens*. A atenção esparsa reduz significativamente a complexidade computacional e o consumo de memória. No entanto, envolve relações de compromisso importantes, pois ao limitar a conexão entre *tokens*, pode-se perder parcialmente a capacidade de captar dependências complexas de longo alcance se não implementada cuidadosamente.

Existem diversas variantes de atenção esparsa, sendo as mais conhecidas a Atenção Local, a Atenção Dilatada e a Atenção Global-Local [Tay et al., 2022]. A **Atenção Local** considera apenas um número limitado de *tokens* próximos ao *token* atual, assumindo que informações relevantes estão frequentemente próximas no texto. A **Atenção Dilatada** (*Strided Attention*), em que *tokens* atendem a elementos posicionados em intervalos fixos, permite que o modelo capture informações contextuais mais distantes com

menos cálculos. A **Atenção Global-Local** combina atenção local para contextos próximos e *tokens* específicos que são atendidos globalmente para capturar informações relevantes mais distantes.

Outra classe importante de mecanismos de atenção eficientes é baseada na linearização ou na utilização de *kernels*. Esses métodos evitam o cálculo direto da matriz completa de atenção através de transformações matemáticas inteligentes, tais como o *Linformer* e o *Performer*, discutidos na Seção 4.3.4. Essas abordagens lineares e baseadas em *kernel* garantem melhor desempenho computacional, embora exijam cuidado especial na definição dos *kernels* e das transformações aplicadas, que podem influenciar diretamente a performance final do modelo.

As técnicas de *downsampling* e *pooling* são também estratégias alternativas para lidar com sequências extensas, reduzindo diretamente a complexidade computacional ao diminuir a resolução ou o número de *tokens* processados. São exemplos dessas técnicas o *Funnel Transformer* e o *Perceiver* [Tay et al., 2022]. A técnica **Funnel Transformer** aplica camadas de *pooling* intercaladas com camadas *Transformers* convencionais para reduzir gradualmente o comprimento da sequência ao longo das camadas intermediárias. À medida que a sequência é comprimida, as camadas subsequentes conseguem operar com custos computacionais menores, ao mesmo tempo em que mantêm as informações essenciais. O *Funnel Transformer* é especialmente útil em tarefas de compreensão textual em que o contexto pode ser condensado sem perda significativa de informação. O **Perceiver** reduz a complexidade processando a sequência original através de um conjunto limitado de *tokens* latentes que atuam como uma representação comprimida da entrada. Esses *tokens* latentes agregam informações globais através de um processo iterativo de atenção. Como resultado, o *Perceiver* pode lidar eficientemente com entradas muito grandes, inclusive em aplicações multimodais que incluem texto, imagens e áudio.

Ambas as abordagens proporcionam uma redução considerável nos custos computacionais e de memória, tornando os *Transformers* viáveis em cenários reais com grandes quantidades de dados ou sequências extensas. Contudo, esses métodos dependem da capacidade dos mecanismos de *pooling* ou *tokens* latentes de reter informações cruciais, o que pode envolver desafios no balanceamento entre compressão e preservação da qualidade da informação.

4.4.3. Poda

A técnica de poda (*pruning*) em redes neurais consiste na remoção seletiva de componentes menos relevantes da rede, como pesos, neurônios ou camadas, com o objetivo de reduzir a complexidade e o tamanho do modelo, mantendo seu desempenho funcional. Essa estratégia é particularmente vantajosa para otimizar grandes modelos, tornando-os mais eficientes em termos computacionais e adequados para execução em dispositivos com recursos limitados. Diferentemente de outras abordagens de otimização para LLMs, a poda atua diretamente na eliminação de parâmetros redundantes ou de baixa relevância [Xu et al., 2024]. Essa técnica pode ser classificada segundo dois critérios principais, o momento da aplicação e o nível de granularidade. Quanto ao momento da aplicação, distinguem-se **poda estática**, realizada após o treinamento do modelo; e a **poda dinâmica**, aplicada durante o processo de treinamento. Com relação ao nível de

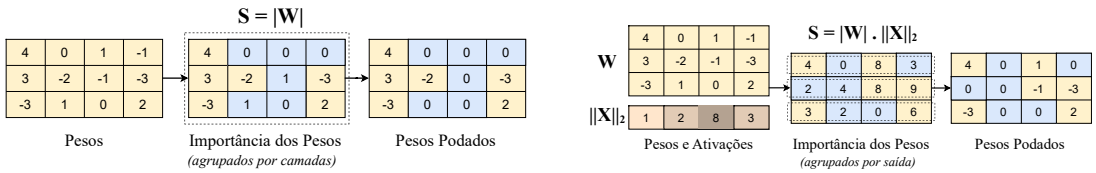
granularidade, a literatura identifica três vertentes principais de poda para LLMs [Kaddour et al., 2023], sendo elas:

- **Poda Estruturada** caracterizada pela remoção de blocos inteiros de parâmetros, como camadas, canais ou filtros, favorecendo a otimização de hardware por promover padrões regulares de acesso à memória e computações mais simples. Nesse contexto, Kaddour *et al.* propuseram o LLM-Pruner[3], um método de compressão que preserva a capacidade multitarefa dos modelos sem dependência excessiva do conjunto de dados original [Ma et al., 2023]. O método é dividido em três etapas principais: (i) descoberta de estruturas acopladas, na qual são identificados grupos de neurônios interdependentes a serem podados conjuntamente para evitar degradação do desempenho; (ii) estimação de importância, que avalia a relevância desses grupos por meio de informações de gradiente e da Hessiana aproximada, priorizando a remoção dos menos críticos; e (iii) ajuste fino leve, que utiliza uma quantidade reduzida de dados e tempo para refinar o modelo, acelerando substancialmente a etapa de pós-treinamento.

- **Poda Semi-Estruturada** também conhecida como poda contextual, opera em um nível intermediário de granularidade, removendo padrões regulares de pesos, como linhas ou colunas inteiras de matrizes (*block pruning*), ou canais específicos em redes convolucionais. Diferentemente da poda não estruturada, essa técnica produz subarquiteturas consistentes, ou seja, estruturas neurais que mantêm padrões de conectividade regulares e previsíveis. Essa característica facilita a aceleração em hardware convencional (GPUs/TPUs) sem requerer suporte à esparsidade. Seguindo essa premissa, métodos recentes propõem abordagens mais eficazes para explorar a esparsidade N:M em LLMs. A técnica MaskLLM [Fang et al., 2024] introduz uma abordagem de aprendizagem baseada em Gumbel Softmax, capaz de gerar máscaras semi-estruturadas otimizadas via treinamento fim-a-fim, com alto grau de transferibilidade para diferentes domínios e tarefas. Por outro lado, o método PBS2P [Yan et al., 2025] combina poda semi-estruturada com binarização progressiva, propondo uma estratégia passo-a-passo de otimização que reduz o erro total de quantização e poda, inclusive abaixo do erro da binarização isolada.

- **Poda Não Estruturada** prevê a remoção de pesos específicos de uma rede neural com base em critérios como magnitude (Poda por Magnitude) ou impacto estimado na função de perda. Essa técnica resulta em uma matriz de pesos esparsa, com valores zero distribuídos de forma irregular. Essa abordagem é altamente granular, permitindo a preservação de conexões críticas enquanto elimina parâmetros redundantes, o que pode reduzir o tamanho do modelo em até 90% sem perda significativa de desempenho. No entanto, sua eficiência prática depende de *hardware* especializado, já que arquiteturas convencionais como GPUs padrão, não processam zeros de forma otimizada. Nesse contexto, o SparseGPT[4] destaca-se como um algoritmo eficiente de poda *one-shot* pós-treinamento, voltado especialmente para modelos da família GPT [Frantar e Alistarh, 2023]. Ele alcança entre 50% e 60%

[3]Disponível em `https://github.com/horseee/LLM-Pruner`.
[4]Disponível em `https://github.com/IST-DASLab/sparsegpt`.



(a) Poda por Magnitude. (b) Método Wanda.

Figura 4.4. Comparação entre duas estratégias de poda não estruturada. Em (*a*) a Poda por Magnitude remove os pesos com menor valor absoluto, considerando a importância apenas com base na magnitude dos pesos, agrupada por camada. Em (*b*), o método Wanda considera tanto os pesos quanto a magnitude das ativações de entrada, calculando a importância por saída. Essa abordagem leva em conta a contribuição real dos pesos durante a inferência, resultando em uma poda mais informada e seletiva.

de esparsidade sem necessidade de retreinamento, por meio de uma otimização local baseada em Hessiana aproximada. Na prática, o algoritmo reformula a poda como um problema de regressão esparsa em larga escala, resolvido iterativamente com atualizações parciais nos pesos para compensar os erros introduzidos, permitindo compressão rápida e eficiente. Similarmente, o método Wanda[5] (*Pruning by **W**eights **and** **a**ctivations*) aplica a poda com base na magnitude dos pesos combinada à norma das ativações de entrada correspondentes [Sun et al., 2023]. Matematicamente esse método é formulado por $S_{ij} = |W_{ij}| \cdot \|X_j\|_2$ onde ($|W_{ij}|$) representa o valor absoluto do peso na matriz e ($\|X_j\|_2$) corresponde à norma ℓ_2 (Euclidiana) das ativações associadas, calculada a partir de um pequeno conjunto de dados de calibração. Tal método alcança desempenho semelhante ao do SparseGPT, porém com maior simplicidade uma vez que realiza a poda em uma única passagem direta pelo modelo, sem necessidade de iterações adicionais ou retropropagação. A Figura 4.4 ilustra essas duas estratégias de poda não estruturada.

4.4.4. Destilação de Conhecimento

A destilação de conhecimento (*Knowledge Distillation* - KD) é uma técnica de compressão e transferência de conhecimento em que um modelo de menor porte, denominado aluno, é treinado para reproduzir o comportamento de um modelo maior e mais expressivo, conhecido como professor. Tipicamente, esse processo envolve o uso das distribuições de saída do professor, que representam a probabilidade atribuída a cada classe ou *token*. Essas distribuições são geradas a partir dos *logits*, valores brutos produzidos pelo modelo antes da aplicação da função *softmax*, e refletem o grau de confiança nas predições. O aluno, então, é otimizado para minimizar a divergência entre suas próprias distribuições e as do professor, aproximando-se do seu comportamento probabilístico [Gou et al., 2021]. Para uma tarefa de classificação multiclasse, em que o modelo estudante geralmente utiliza uma função *softmax* para suas ativações, a função de perda em um processo de KD é composta por duas partes, a perda de destilação e a perda de estudante [Li et al., 2023]. Essa função de perda, representada por $\mathscr{L}_{KD}$, é definida por

[5]Disponível em `https://github.com/locuslab/wanda`.

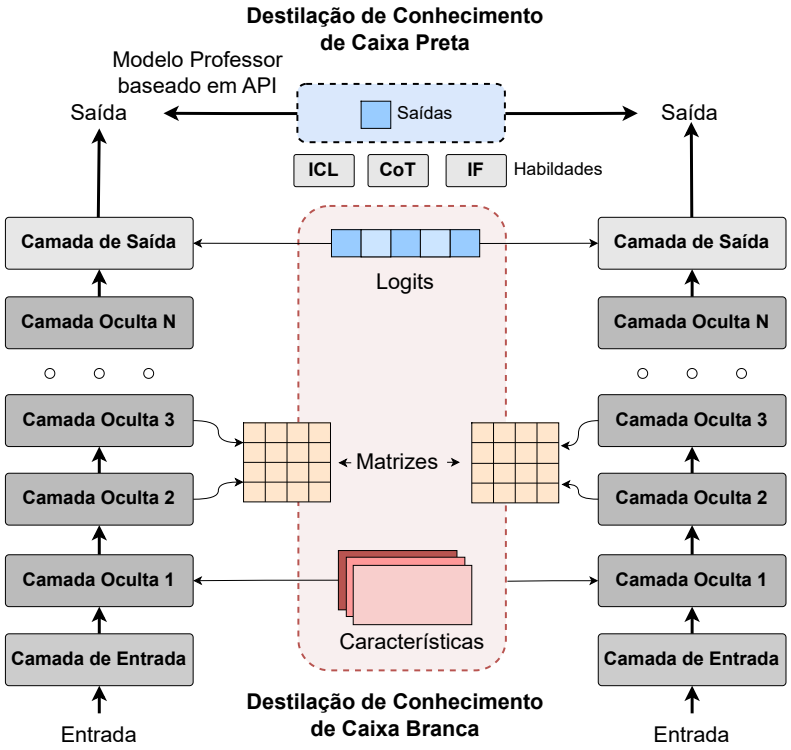


Figura 4.5. Diagrama geral de técnica de destilação de conhecimento, com um modelo professor à esquerda e um modelo estudante à direita. No topo, a destilação de conhecimento de caixa preta utiliza apenas as saídas do professor (*logits*), obtidas por tarefas como *In-context Learning* (ICL), *Chain-of-Thought* (CoT) e *Instruction Following* (IF). Na base, a destilação de caixa branca envolve o alinhamento de representações internas, como características latentes ou matrizes de atenção, entre os modelos, permitindo uma transferência de conhecimento mais detalhada.

$$\mathscr{L}_{KD} = H(\mathbf{y}, \mathbf{y}_s) + \lambda H(\mathbf{y}_t, \mathbf{y}_s),$$

em que $\mathbf{y}$ representa a verdade fundamental (*ground truth*), os rótulos reais do conjunto de dados de treinamento. As saídas do modelo estudante e do modelo professor são denotadas por $\mathbf{y}_s$ e $\mathbf{y}_t$ respectivamente. H é a função de custo, tipicamente uma entropia cruzada, aplicada tanto à comparação com a verdade fundamental quanto à comparação entre as saídas do professor e do aluno. O parâmetro λ atua como um fator de equilíbrio, ajustando a contribuição de cada uma dessas duas parcelas na perda total. Como ilustrado na Figura 4.5, essa técnica detém duas principais abordagens conforme o nível de acesso ao modelo professor:

- **Destilação de Conhecimento de Caixa Preta** (*Black-box Knowledge Distillation*), abordagem que se fundamenta na ausência de acesso aos parâmetros internos ou representações intermediárias do modelo professor, restringindo o aprendizado do modelo aluno exclusivamente às saídas observáveis, tipicamente distribuições de probabilidade sobre o espaço de saída. Tal configuração é particularmente útil quando o modelo professor é disponibilizado apenas por meio de interfaces restritas, como APIs comerciais, ou quando há incompatibilidade estrutural entre as

arquiteturas de professor e aluno. Nesses casos, o processo de destilação consiste em minimizar a divergência entre as predições do aluno e as respostas do professor, com base em pares entrada-saída.

- **Destilação de Conhecimento de Caixa Branca** (*White-box Knowledge Distillation*), abordagem que pressupõe total acesso ao funcionamento interno do modelo professor, incluindo seus estados latentes, ativação de camadas intermediárias e pesos. Tal disponibilidade permite que o modelo aluno seja treinado para alinhar não apenas suas predições finais, mas também suas representações internas às do professor, promovendo uma transferência de conhecimento mais profunda e semântica. Tal alinhamento pode envolver a minimização de distâncias entre *embeddings*, mapas de atenção ou distribuições de ativação, aumentando significativamente a fidelidade do modelo destilado. No entanto, a efetividade dessa abordagem depende de uma compatibilidade arquitetural entre os modelos envolvidos, e sua implementação tende a ser mais complexa e computacionalmente intensiva, exigindo maior controle sobre o *pipeline* de treinamento e acesso integral ao modelo fonte.

Combinando características de ambas as abordagens, o MiniLLM[6], proposto por Gu *et al.* [Gu et al., 2023], é um método eficaz para destilação de LLMs proprietárias, utilizando a divergência de Kullback-Leibler reversa. Embora formalmente classificado como um método de caixa branca, já que acessa a distribuição de probabilidade do professor, o MiniLLM também incorpora elementos típicos da caixa preta, como a priorização da qualidade das saídas do aluno e estratégias de otimização baseadas em *feedback*. Essa abordagem híbrida permite maior flexibilidade na destilação de modelos restritos, mantendo alto desempenho mesmo com acesso limitado.

4.4.5. Técnicas Avançadas para Otimização de Modelos Embarcados

A otimização eficiente dos Modelos de Linguagem Compactos (*Small Language Models* - SLMs), especialmente aqueles com menos de um bilhão de parâmetros, requer técnicas avançadas para maximizar desempenho sem comprometer a eficiência computacional. Entre as técnicas mais eficazes destacam-se a utilização da atenção multi-consulta (*Multi-Query Attention* - MQA), a atenção por consultas agrupadas (*Grouped-Query Attention* - GQA), compartilhamento de camada e de representações, bem como o uso de arquiteturas profundas e estreitas (*Deep-and-Thin*). Essas abordagens permitem significativa redução no consumo de memória e no tempo de inferência, preservando a precisão e estabilidade dos modelos, fatores críticos especialmente em aplicações com recursos computacionais restritos ou dispositivos móveis [Liu et al., 2024b, Ainslie et al., 2023].

As técnicas de compartilhamento de camada e de representação, ***Layer Sharing*** e ***Embedding Sharing***, são estratégias eficazes para melhorar a eficiência dos modelos de linguagem sem aumentar o custo computacional ou o número total de parâmetros. O *Layer Sharing* consiste em reutilizar os mesmos pesos em múltiplas camadas adjacentes do modelo, permitindo aumentar a profundidade efetiva sem aumentar o tamanho da memória requerida. Isso é especialmente vantajoso em ambientes com restrições severas de memória, pois mantém os pesos na memória cache durante a inferência, minimizando

[6]Disponível em `https://github.com/microsoft/LMOps/tree/main/minillm`.

movimentos custosos de memória. Por outro lado, a técnica de *Embedding Sharing* reutiliza os pesos da camada de representação (*embedding*) de entrada como pesos da camada de projeção de saída, reduzindo consideravelmente o número total de parâmetros dedicados às representações vetoriais das palavras. Essa estratégia maximiza a eficiência no uso dos pesos, sendo particularmente útil em modelos menores, em que as camadas de representação são uma parcela significativa dos parâmetros totais. Ambas as abordagens garantem alto desempenho com menores custos computacionais e de armazenamento [Liu et al., 2024b].

As técnicas de **atenção por múltiplas consultas** (*Multi-Query Attention* - MQA) e **atenção por consultas agrupadas** (*Grouped-Query Attention* - GQA) são estratégias avançadas utilizadas em modelos baseados em *Transformers* para reduzir custos computacionais e o uso de memória durante a inferência, tornando-os ideais para aplicação em dispositivos móveis. A técnica MQA funciona simplificando o mecanismo de atenção tradicional (*Multi-Head Attention* - MHA), que normalmente emprega múltiplas cabeças para consultas, chaves e valores [Shazeer, 2019]. Ao contrário do MHA, o MQA utiliza múltiplas cabeças apenas para as consultas, enquanto as chaves (*keys*) e valores (*values*) são representados por apenas uma única cabeça compartilhada entre todas as consultas. Essa simplificação reduz drasticamente a quantidade de memória necessária para armazenar as representações intermediárias de chaves e valores durante a inferência. Contudo, devido a essa redução significativa, pode ocorrer degradação na qualidade final das respostas, especialmente se não houver adaptação adicional através de um processo conhecido como *uptraining*. O processo de *uptraining* consiste em adaptar um modelo previamente treinado para uma nova configuração arquitetural, realizando um treinamento adicional breve com um subconjunto dos dados originais para que o modelo se ajuste às mudanças introduzidas e recupere ou supere sua capacidade preditiva inicial.

Para contornar a possível perda de qualidade do MQA, surgiu a técnica GQA [Ainslie et al., 2023], que representa um compromisso intermediário entre o MHA e o MQA. No GQA, as cabeças de consulta são divididas em grupos e cada grupo compartilha uma única cabeça de chave e valor. Assim, ao invés de uma única cabeça de chave-valor, como no MQA, ou várias cabeças individuais, como no MHA, o GQA estabelece uma configuração intermediária com mais de uma, mas menos cabeças de chave-valor do que o número total de cabeças de consulta. Essa configuração intermediária mantém boa parte da eficiência computacional e economia de memória do MQA, reduzindo significativamente a necessidade de armazenamento das representações intermediárias, enquanto mantém uma qualidade próxima ao MHA.

As técnicas MQA e GQA se diferenciam essencialmente pelo grau de compartilhamento das representações intermediárias, já que, enquanto o MQA utiliza um compartilhamento completo, sendo uma única cabeça de chave-valor para todas as consultas, o GQA utiliza um compartilhamento parcial em que cabeça de chave-valor é compartilhadas por um grupo de consultas, equilibrando eficiência e qualidade. Para modelos embarcados em dispositivos móveis, tanto o MQA quanto o GQA são altamente vantajosos, pois permitem reduzir a quantidade de dados que precisam ser carregados repetidamente da memória, melhorando o desempenho da inferência e minimizando o consumo energético. O uso dessas técnicas possibilita a implementação prática de modelos de alto desempenho e baixo custo computacional em dispositivos móveis com recursos limitados,

contribuindo para experiências de usuário mais rápidas, eficientes e sustentáveis.

A técnica **Deep-and-Thin** envolve a adoção de modelos com maior profundidade e menor largura (menos parâmetros por camada), contrariando a ideia prevalente de que apenas o número absoluto de parâmetros define o desempenho dos modelos linguísticos. Essa abordagem privilegia um número maior de camadas, porém com menos neurônios em cada camada, aumentando a capacidade do modelo de capturar abstrações complexas e hierárquicas sem aumentar significativamente a quantidade total de parâmetros. Na prática, isso resulta em modelos mais eficientes, com menor custo computacional durante a inferência e maior capacidade de generalização em comparação a modelos tradicionais mais largos e superficiais. Essa estratégia mostrou ganhos significativos em tarefas de raciocínio e compreensão semântico-textual, especialmente em cenários de uso limitado de recursos [Liu et al., 2024b].

4.5. Classes e Características de Modelos de Linguagem em Larga Escala

Um LLM eficaz deve ter quatro características principais: (i) capacidade de captar e modelar o contexto da linguagem natural; (ii) habilidade de gerar texto que semelhante a linguagem humana; (iii) manutenção de contexto e aplicação de conhecimento em domínios específicos; (iv) forte capacidade de seguir instruções, o que é útil para a resolução de problemas e apoio na tomada de decisões [Yao et al., 2024b]. Outro aspecto importante é o gerenciamento de viés presente nos dados. Como os LLMs são treinados em grandes quantidades de texto retiradas da Internet e de outras fontes amplamente disponíveis, há um risco de que esses modelos apresentem vieses presentes nesses dados, o que pode levar a resultados indesejados e até mesmo discriminatórias [Bai et al., 2024].

O **GPT** (*Generative Pre-trained Transformer*) é um modelo de aprendizado profundo baseado na arquitetura *Transformer*, pré-treinado em extensos corpora de texto. Ele pode ser ajustado para uma ampla variedade de tarefas de Processamento de Linguagem Natural, como geração de texto, análise de sentimentos, modelagem de linguagem, tradução automática e classificação. Sua arquitetura representa um avanço significativo em relação a abordagens anteriores baseadas em RNNs e CNNs, ao permitir o processamento eficiente de dependências de longo alcance no texto [Yenduri et al., 2024]. Esse modelo ficou amplamente conhecido em função do ChatGPT desenvolvido pela empresa OpenAI[7]. Esse modelo é capaz de executar diversas tarefas, entre elas *Natural Language Understanding* (NLU), permitindo analisar e interpretar textos e reconhecer entidades e relacionamentos dentro de frases [Mahmud et al., 2025]. A versão inicial, GPT-1, lançada em 2018, empregou uma arquitetura de decodificador *Transformer* com 12 camadas e aproximadamente 110 milhões de parâmetros [Zheng et al., 2021]. Seu treinamento não supervisionado em vastos volumes de texto possibilitou a geração de respostas contextualmente relevantes. Subsequentemente, o GPT-2, introduzido em 2019, expandiu significativamente essa capacidade, com variantes que atingiram até 48 blocos e 1,5 bilhões de parâmetros [Jawahar et al., 2019], resultando em uma notável melhoria na coerência e diversidade da geração textual. O marco seguinte, GPT-3, lançado em 2020, elevou o padrão com 175 bilhões de parâmetros. Essa escala permitiu avanços substanciais em cenários de *few-shot* e *zero-shot learning*, capacitando o modelo a executar tarefas complexas

[7]Disponível em `https://openai.com/`.

com poucos ou nenhum exemplo de treinamento. Em 2023, a OpenAI lançou o GPT-4, um modelo ainda mais avançado e eficiente, com a capacidade de analisar não apenas texto, mas também imagens, caracterizando-o como um modelo multimodal. O GPT-4 tem demonstrado desempenho em níveis humanos em diversas avaliações e exames padronizados, além de gerar textos com maior naturalidade e completude, sublinhando sua escalabilidade e versatilidade multimodal.

O modelo **T5** (*Text-to-Text Transfer Transformer*) é baseado na arquitetura *Transformer encoder-decoder* que reformula todas as tarefas de linguagem natural como problemas de entrada e saída de texto, permitindo uma abordagem unificada para tradução, resumo, classificação, resposta a perguntas e outras tarefas de PLN [Raffel et al., 2020]. A principal inovação desse modelo, consiste em uma abordagem unificada, na qual toda tarefa de linguagem é reformulada como um problema de "texto para texto", permitindo que funções diversas como responder a perguntas, resumir, classificar e traduzir sejam processadas de forma consistente [Mahmud et al., 2025]. Sua versatilidade o torna uma ferramenta poderosa para uma ampla variedade de aplicações em PLN, consolidando-se como uma das arquiteturas mais abrangentes.

O **BERT** (*Bidirectional Encoder Representations from Transformers*) é um modelo de *Deep Learning* bidirecional baseado na arquitetura *Transformer* que analisa todo o contexto de uma sentença, ou seja, na hora de prever um *token*, o BERT avalia tanto palavras anteriores quanto palavras posteriores, ao contrário de modelos unidirecionais como GPT, que utilizam apenas o histórico [Zheng et al., 2021]. O modelo opera em duas fases sequenciais, pré-treinamento e ajuste fino (*fine-tuning*) [Aftan e Shah, 2023]. A fase de pré-treinamento visa a compreensão da linguagem, enquanto o ajuste fino adapta o modelo para tarefas específicas, tais como análise de sentimentos, resposta a perguntas e sumarização de textos. No pré-treinamento, o BERT emprega duas estratégias, Modelagem de Linguagem Mascarada (*Masked Language Modeling* - MLM) e Previsão da Próxima Sentença (*Next Sentence Prediction* - NSP). A MLM consiste em predizer palavras mascaradas em uma sentença com base em seu contexto bidirecional. Por sua vez, a NSP treina o modelo para discernir a relação de coerência entre pares de sentenças, prevendo se uma sentença sucede logicamente a outra [Djeffal et al., 2023]. Essa combinação o torna uma representação extremamente poderosa, capaz de compreender nuances e contextos complexos na linguagem natural.

Um dos principais modelos de código aberto, é o **LLaMA** (*Large Language Model Meta AI*)[8] desenvolvido pela empresa Meta. LLaMa é uma família de modelos baseados na arquitetura *Transformer*, projetados para serem uma alternativa capaz de competir com outros modelos comerciais, como o ChatGPT. O LLaMA se destaca por sua arquitetura otimizada, permitindo sua execução em dispositivos com menor capacidade de recursos disponíveis, sem comprometer o desempenho [Touvron et al., 2023]. Essa eficiência faz com que o LLaMA seja uma opção viável para pesquisas que buscam soluções de processamento de linguagem natural, mas que não têm acesso a grandes infraestruturas de computação. Uma das técnicas utilizadas pelo LLaMA, é o uso *embeddings positional* que codificam a posição relativa ou absoluta dos *tokens* na sequência, garantindo que a ordem seja considerada durante o processamento. Já as camadas *feed-forward* subsequentes

[8]Disponivel em `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`.

Tabela 4.3. Comparação de diferentes modelos de LLM de código aberto, em termos de provedores e número de parâmetros disponíveis.

Modelo	Versão	Parâmetros	Provedor	Data
GPT [OpenAI et al., 2024]	4	Não divulgado	OpenAI	Março 2024
T5 [Raffel et al., 2020]	1.1	11 B	Google	Janeiro 2023
Gemma [Gemma Team et al., 2025]	3	27 B	Google	Março 2025
LLaMA	4	402 B	Meta AI	Abril 2025
Falcon [Falcon-LLM Team, 2024]	3	10 B	TII	Dezembro 2024
PaLM [Anil et al., 2023]	2	540 B	Google	Maio 2023

utilizam funções de ativação não lineares, como GELU, seguidas por projeções lineares, formando uma estrutura altamente modular e eficiente na captura de relações de longo alcance [Dubey et al., 2024]. Além disso, o modelo incorpora otimizações, como técnicas de *checkpointing* de ativação e atenção eficiente, que reduzem o consumo de memória durante o treinamento, permitindo que modelos de até 400 bilhões de parâmetros sejam treinados de forma mais viável. Essa combinação de melhorias arquiteturais e técnicas de treinamento garante que o LLaMA seja um modelo altamente escalável e versátil.

O modelo **PaLM** *(Pathways Language Model)*, desenvolvido pela empresa Google, também utiliza a arquitetura *Transformer*. Esse modelo utiliza a arquitetura **Pathways**, que permite treinar um único modelo em várias tarefas ao mesmo tempo, ao invés de treinar modelos separados para cada tarefa. Esse paralelismo oferece uma grande vantagem em termos de flexibilidade e economia de recursos computacionais, uma vez que reduz o tempo de treinamento. O PaLM foi treinado com um corpus de 780 bilhões de *tokens*, cobrindo uma vasta gama de tarefas linguísticas e de raciocínio, o que contribui para seu desempenho de ponta em tarefas de linguagem natural, geração de código e raciocínio matemático [Chowdhery et al., 2023].

O modelo **Falcon** é uma família de modelos desenvolvida pelo Technology Innovation Institute (TII)[9], com destaque para o uso de dados exclusivamente extraídos da *web*. Diferente de abordagens anteriores que combinavam *corpora* selecionados manualmente e dados da *web*, o Falcon foi treinado com o conjunto RefinedWeb [Penedo et al., 2023]. Sua arquitetura é baseada no PaLM [Chowdhery et al., 2023], sendo um *transformer* autoregressivo com atenção causal, utilizando ALiBi [Press et al., 2022] para codificação posicional e FlashAttention [Dao et al., 2022] para otimização da atenção.

A Tabela 4.3 apresenta um comparativo entre os diversos modelos LLM disponíveis. Já a Tabela 4.4 mostra a comparação entre os tipos de arquitetura, treinamento e capacidades dos modelos LLM de código aberto.

Os modelos compactos de linguagem são uma classe de modelos projetados especificamente para oferecer desempenho eficiente no processamento local de linguagem natural em ambientes com recursos computacionais limitados, como dispositivos móveis, sistemas embarcados ou aplicações em tempo real [V. et al., 2025]. Em comparação aos LLMs, o número reduzido de parâmetros característico dos SLMs resulta em maior celeridade tanto para a análise de consultas quanto para a geração de respostas [Zhang et al., 2025], o que reduz significativamente seu consumo de memória, processamento e ener-

[9]Disponível em `https://tii.ae/`.

Tabela 4.4. Comparativo entre os tipos de arquitetura, treinamento e capacidades dos modelos LLM.

Modelo	Arquitetura	Treinamento	Capacidades
GPT	*Transformer Decoder-only* com atenção causal e camadas de *self-attention*	Pré-treinado via *Causal Language Modeling* em grandes corpora textuais. Ajuste fino opcional	Geração de texto, sumarização, tradução, QA, diálogo contextual
T5	Arquitetura *Encoder-Decoder* com abordagem texto-para-texto.	Pré-treinado com *span corruption* e ajustado supervisionadamente para tarefas específicas.	Tradução, resumo, classificação, QA, geração condicional de texto.
BERT	*Transformer Encoder-only* com atenção bidirecional.	Pré-treinado com *Masked Language Modeling* e *Next Sentence Prediction*.	Classificação, reconhecimento de entidades, QA tipo *extractive*, análise de sentimento.
LLaMA	*Transformer Decoder-only* com atenção causal, RoPE, SwiGLU.	Pré-treinado com *Causal Language Modeling* sobre grande volume de dados públicos.	Geração de texto, resposta a perguntas, diálogo, base para LLMs especializados.
Falcon	*Transformer Decoder-only* com atenção causal, RoPE e *Multiquery Attention*.	Pré-treinado com *Causal Language Modeling* em grandes conjuntos de dados filtrados da *web*.	Geração de texto eficiente, diálogo, resumo e tarefas gerais de NLP.
PaLM	*Transformer Decoder-only* com SwiGLU, paralelismo de caminho (Pathways).	Pré-treinado com *Causal Language Modeling* usando grandes datasets multilíngues e multitarefa.	Raciocínio, QA complexo, geração de código, tradução, generalização multitarefa.

gia, mantendo ainda assim um desempenho satisfatório para tarefas específicas. Modelos compactos são frequentemente otimizados por técnicas como quantização, poda de parâmetros e destilação de conhecimento, permitindo aplicações práticas em situações em que o uso de grandes modelos é inviável devido a restrições técnicas ou econômicas.

O modelo **Phi-3** é uma família de modelos desenvolvida pela Microsoft, cuja versão principal, o phi-3-mini, possui 3,8 bilhões de parâmetros e é baseada na arquitetura de *Transformer* do tipo *decoder*, similar à do LLaMA-2. O treinamento prioriza a qualidade dos dados em vez do aumento de escala. O phi-3 é treinado com 3,3 trilhões de *tokens* compostos por dados da Internet e dados sintéticos gerados por outros LLMs. Adicionalmente, variantes com arquiteturas maiores e capacidades multimodais e multilinguísticas foram desenvolvidas, como o phi-3.5-MoE, que adota uma estrutura de *Mixture-of-Experts* com 6,6 bilhões de parâmetros ativos, promovendo ganhos significativos em tarefas de raciocínio, matemática e programação [Abdin et al., 2024].

O modelo **DistilBERT** representa uma versão compactada do modelo BERT, desenvolvido com o objetivo de otimizar a eficiência computacional e a velocidade de inferência, sem comprometer a capacidade de compreensão de linguagem natural. Sua arquitetura baseia-se na técnica de destilação de conhecimento, discutida na Seção 4.4.4. DistilBERT preserva aproximadamente 97% do desempenho do BERT original, mas com uma redução de 40% no número de parâmetros e um aumento de 60% na velocidade de inferência [Sanh et al., 2020]. O processo de treinamento integra três funções de perda,

Tabela 4.5. Comparativo entre SLMs em termos de provedores e número de parâmetros.

Modelo	Versão	Parâmetros	Provedor	Data
Phi-3 [Abdin et al., 2024]	mini	3.8 B	Microsoft	Junho 2024
TinyLlama [Zhang et al., 2024]	1.1	1.1 B	StatNLP	Janeiro 2023
DistilBERT [Sanh et al., 2020]	-	66 M	Google	2020
ALBERT [Lan et al., 2020]	large	18 M	Google	2019
Qwen [Yang et al., 2025]	2.5	0.5 B	Alibaba	2025

MLM, destilação e distância do cosseno. Esse modelo é particularmente adequado para uso em dispositivos móveis e ambientes com recursos limitados [Sanh et al., 2020].

O **TinyLlama** é um modelo de linguagem compacto de código aberto, com aproximadamente 1,1 bilhão de parâmetros, desenvolvido a partir da arquitetura do Llama 2. Seu objetivo é ser uma alternativa eficiente e leve a modelos maiores, mantendo o desempenho em tarefas de compreensão, raciocínio e geração de texto. Seu treinamento, que utilizou até 3 trilhões de *tokens* de conjuntos de dados limpos como SlimPajama e StarCoder, incorpora técnicas avançadas como FlashAttention [Zhang et al., 2024]. Além da versão padrão, existem outras especializadas para matemática e programação. Essa combinação de tamanho reduzido e treinamento otimizado torna o TinyLlama ideal para aplicações com recursos computacionais limitados, como em dispositivos móveis e sistemas embarcados [Zhang et al., 2024].

O **ALBERT** é um modelo de linguagem baseado no BERT, desenvolvido para ser mais eficiente em termos de memória e tempo de treinamento, mantendo ou melhorando o desempenho em tarefas de PLN. Esse modelo introduz duas técnicas de redução de parâmetros: (i) Parametrização de incorporação fatorada (*Factorized embedding parameterization*), ao dividir a matriz de vocabulário, pode-se aumentar o tamanho das camadas ocultas sem expandir o número de parâmetros dos *embeddings* de vocabulário; e (ii) o compartilhamento de parâmetros entre camadas (*cross-layer parameter sharing*), que reduz significativamente o número total de parâmetros. O ALBERT substitui a tarefa de predição da próxima sentença (NSP) pela tarefa de predição da ordem das sentenças (SOP), que melhora a modelagem da coerência entre sentenças [Lan et al., 2020].

O **Qwen** é um modelo baseado na arquitetura *Transformer*, desenvolvido para processar sequências de texto com até 1 milhão de *tokens*. Ele foi treinado com dados naturais e sintéticos, utilizando estratégias de extensão progressiva de contexto e ajustes supervisionados em múltiplas etapas, incluindo aprendizado por reforço. Seu funcionamento é otimizado para tarefas de longo contexto, como recuperação de informações, leitura extensiva de documentos e geração de código, por meio de técnicas como atenção esparsa, extrapolação de comprimento e paralelismo de execução [Yang et al., 2025]. As Tabelas 4.5 e 4.6 apresentam um comparativo entre SLMs.

4.6. Avaliação de Modelos de Linguagem em Larga Escala

A avaliação eficaz de modelos de linguagem em larga escala, particularmente no que tange ao impacto de sua compressão, é tratada na literatura por meio de duas abordagens metodológicas principais. A Figura 4.6 ilustra uma taxonomia de métricas e critérios, enquadrados nas duas abordagens distintas: (i) qualidade cognitiva, que reflete a

Tabela 4.6. Comparação entre os tipos de arquitetura, treinamento e capacidades de modelos SLM.

Modelo	Arquitetura	Treinamento	Capacidades
Phi-3	*Transformer Decoder*	Treinamento supervisionado com dados sintéticos cuidadosamente filtrados (*curriculum learning*)	Geração de texto, raciocínio lógico, tarefas *few-shot* com alta performance
TinyLlama	*Transformer Decoder* (baseado no LLaMA)	Pré-treinamento autoregressivo com *corpus* diverso e otimizações em memória e velocidade	Geração de texto, compreensão de linguagem, tarefas de NLP em dispositivos de borda
DistilBERT	*Transformer Encoder* (baseado no BERT)	Distilação do BERT com redução de camadas, mantendo 97% da performance original	Classificação de texto, QA, análise de sentimentos, *embeddings*
ALBERT	*Transformer Encoder* com compartilhamento de pesos e *embeddings* fatorados	Treinamento com *autoencoder* com predição de frases	Tarefas de NLP como QA, NLI, classificação com alta eficiência em memória
Qwen	*Transformer Decoder* com *Rotary Positional Embeddings* e Tokenização baseada em *WordPiece*	Pré-treinamento autoregressivo em múltiplos domínios, com foco em multilinguismo e código	Geração de texto, respostas em linguagem natural, geração de código, multilinguismo

precisão e a capacidade de generalização do modelo em tarefas não vistas, abrangendo tanto saídas textuais generativas quanto extrativas; e (ii) desempenho operacional, que engloba aspectos fundamentais do custo e da eficiência computacional.

Os paradigmas de *zero-shot*, *few-shot* e suas variantes descrevem diferentes formas de aprendizado em contexto (*in-context learning*) em LLMs, caracterizando como esses modelos podem resolver tarefas sem ajustes nos seus pesos (isto é, sem *fine-tuning*), apenas manipulando a estrutura e o conteúdo do *prompt* de entrada. No aprendizado *zero-shot*, o modelo é condicionado a executar uma tarefa sem receber qualquer exemplo explícito de entrada-saída relacionado à tarefa no *prompt*. A inferência é baseada inteiramente no conhecimento adquirido durante o pré-treinamento em larga escala, explorando a capacidade de generalização do LLM. Esse tipo de raciocínio demonstra como o modelo pode mapear instruções linguísticas abstratas para comportamentos coerentes, mesmo em tarefas inéditas. Por sua vez, o aprendizado *few-shot* envolve a apresentação de um número limitado de exemplos dentro do próprio *prompt*, seguido por uma nova instância de entrada para a qual se espera uma saída correspondente. Essa abordagem permite que o modelo induza padrões latentes a partir dos exemplos e adapte temporariamente seu comportamento, dentro da janela de contexto da inferência. O *one-shot learning* é um caso particular, com um único exemplo disponível. Variantes como o *multi-shot learning* ou *many-shot prompting* fornecem dezenas ou centenas de exemplos no *prompt*, limitados apenas pela capacidade da janela de contexto do modelo. Essas variantes são eficazes em tarefas mais complexas ou sensíveis ao estilo e formato da resposta, beneficiando-se de uma amostragem mais representativa da tarefa-alvo.

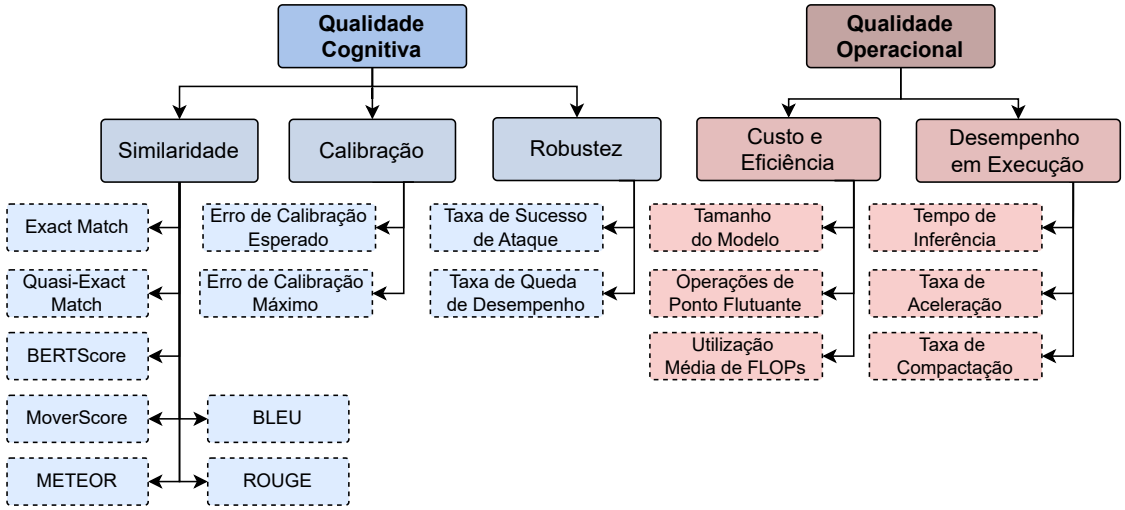


Figura 4.6. Taxonomia de avaliação computacional de modelos de linguaguem. Os criterios e métricas de avaliação focam tando no viés cognitivo, considerando a qualidade da exatidão e capacidade linguística, quanto no viés operacional, considerando a eficiência e desempenho durante a execução.

4.6.1. Critérios e Métricas de Avaliação da Qualidade Cognitiva

A avaliação da qualidade cognitiva de modelos de linguagem representa um desafio multifacetado, impulsionando o desenvolvimento de métricas automatizadas que transcendem a mera correspondência textual. Tradicionalmente, a aferição da qualidade de respostas em PLN tem se baseado extensivamente na análise humana com anotadores, que, embora forneça a verdade fundamental (*ground truth*) de alta fidelidade, é inerentemente subjetiva, demorada e escalável de forma limitada [Chang et al., 2024]. Ademais, a complexidade crescente das saídas de LLMs e a vasta gama de cenários de aplicação tornam a avaliação manual impraticável e suscetível a vieses humanos inconsistentes. Para abordar essa complexidade e mitigar as limitações das abordagens tradicionais, há a proposta de uma estrutura avaliativa automatizada baseada em três critérios chave: a similaridade léxica, a calibração e a robustez. A combinação desses critérios permite uma análise mais abrangente, eficiente e objetiva da capacidade de raciocínio, coerência e adaptabilidade dos LLMs.

O critério de **similaridade léxica** quantifica o grau de sobreposição de termos e sequências de palavras entre o texto gerado por um modelo e um ou mais textos de referência. Essa métrica avalia a correspondência direta de vocabulário e a estrutura frasal, servindo como um indicador da fidelidade ou da relevância do conteúdo gerado em relação a uma base esperada. Sua avaliação pode ser realizada através de diversas métricas, categorizadas tanto por saídas binárias, como *Exact Match* e *Quasi-Exact Match*, quanto por saídas reais, como o *F1-score* e *ROUGE score* [Chang et al., 2024, Yang et al., 2024].

Exact Match (EM) é uma métrica binária que verifica a correspondência literal entre a saída de um modelo e a resposta de referência em tarefas de geração de texto. Em cenários de resposta a perguntas, o EM assume valor 1 se o texto gerado for idêntico à resposta esperada, e 0 caso contrário. **Quasi-Exact Match** (QEM) é uma métrica que estende o conceito de correspondência exata, flexibilizando a exigência de uma equiva-

lência perfeita entre os textos comparados. A QEM considera uma *string* correta mesmo diante de pequenas variações, mediante a aplicação de um pós-processamento leve à saída do modelo. Este processo inclui a normalização para minúsculas, a remoção de espaços em branco e pontuações excessivas, além da exclusão de artigos definidos ou indefinidos. Essa abordagem reconhece que variações triviais em formatação ou na escolha de *stop words* frequentemente não alteram o significado semântico essencial da resposta, conferindo maior robustez à métrica e reduzindo a sensibilidade a detalhes estilísticos irrelevantes para a correção semântica.

F1-score é uma das métricas para avaliação de modelos de classificação, particularmente em problemas binários e cenários desbalanceados em que a classe positiva é rara. Expressa por

$$F_1 = \frac{2 \times \text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}},$$

a F1-score computa a média harmônica entre outras duas métricas de recuperação de informação, a precisão, dada pela acurácia nas predições positivas, e *recall*, que expressa a capacidade de identificar todos os casos positivos. Matematicamente, a métrica é definida entre $[0,1]$, em que valores próximos ao limite superior (1) indicam um modelo com alto desempenho, pois precisão e *recall* estão bem equilibrados e maximizados, enquanto valores próximos ao limite inferior (0) representam um modelo ineficaz, com falhas graves na identificação de casos positivos ou excesso de falsos positivos. Em cenários com mais de duas classes, o F1-score pode ser calculado de três formas: (i) *micro-F1* que agrega contribuições de todas as classes, e.g. somatório de verdadeiros positivos, falsos positivos e falsos negativos, sendo robusto a desbalanceamentos e adequado para análise global; (ii) *macro-F1* que calcula a média aritmética dos F1-scores individuais de cada classe, tratando todas igualmente; e (iii) *weighted-F1* que representa a média ponderada pelo número de instâncias de cada classe, combinando sensibilidade a desbalanceamentos com a equidade entre classes.

BLEU (*Bilingual Evaluation Understudy*) é uma métrica amplamente utilizada para avaliar a qualidade de textos gerados por máquinas, especialmente em tarefas como tradução automática e sumarização de texto. Sua metodologia consiste em comparar o texto gerado com um ou mais textos de referência, quantificando a similaridade com base na sobreposição de *n*-grams, *i.e.* sequências de palavras. A métrica é calculada por

$$BLEU_{\text{weight}} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right),$$

que combina a precisão de *n*-grams, representada por p_n e ponderada por pesos w_n, com uma penalidade por brevidade (*BP*) aplicada para evitar traduções excessivamente curtas. O termo exponencial $\exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$ calcula a média geométrica das precisões para diferentes tamanhos de *n*-grams, enquanto o *BP* ajusta o *score* quando a tradução é mais curta que a referência. Quanto mais próximo de 1, mais similar é o texto gerado comparado à tradução humana de referência.

ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) é um conjunto de

métricas utilizado para quantificar a qualidade da sumarização e tradução automática [Lin, 2004]. Ao contrário do BLEU, que se concentra na precisão, o ROUGE foca principalmente o *recall*, tornando-o particularmente útil para medir quanto do conteúdo do texto de referência é capturado pelo texto gerado. Em sua versão genérica, o ROUGE-*N* mede a sobreposição de *n-grams* entre o texto de referência e o texto gerado por um LLM, expresso por

$$ROUGE\text{-}N = \frac{\text{Número de } n\text{-gramas sobrepostos}}{\text{Total de } n\text{-gramas no texto de referência}}.$$

A variação de *N* define diferentes granularidades, como a **ROUGE-**1, que avalia a sobreposição de palavras únicas (*unigrams*) e a **ROUGE-**2, que mede a sobreposição de duas palavras consecutivas (*bigrams*). Já o **ROUGE-***L* foca na maior subsequência comum, avaliando o quão bem o texto gerado captura a sequência mais longa de palavras compartilhadas, preservando a estrutura do texto de referência. Outras variantes da métrica, como a **ROUGE-***W*, atribui maior peso a correspondências contínuas mais longas, ou seja, sequências de palavras mais extensas e correspondentes entre o texto gerado e a referência recebem pontuações mais altas. Paralelamente, a **ROUGE-***S* mede a sobreposição de *bigrams* que não são necessariamente consecutivos, capturando correspondências mais flexíveis entre os dois textos.

METEOR (*Metric for Evaluation of Translation with Explicit ORdering*) é uma métrica desenvolvida para avaliar a qualidade de tradução automática e geração de linguagem natural. A métrica melhora o BLEU ao considerar tanto a precisão quanto o *recall* [Banerjee e Lavie, 2005], buscando uma correlação mais fiel ao julgamento humano. Para isso, o METEOR incorpora fatores como sinonímia, *stemming* e a ordem das palavras, aplicando uma penalidade quando esta está incorreta. Essencialmente, é uma métrica baseada em alinhamento de *unigrams* que recompensa traduções que preservam o significado e a fluidez, reconhecendo correspondências de sinônimos e radicais. A métrica é dada por

$$\text{METEOR} = (1 - P) \cdot F_{\text{média}},$$

em que $F_{\text{média}}$ é a média harmônica ponderada da precisão e do *recall*, calculada como $F_{\text{média}} = \frac{10 \cdot \text{Precisão} \cdot \text{Recall}}{9 \cdot \text{Precisão} + \text{Recall}}$. O termo P é uma penalidade que varia entre $[0, 1]$, projetada para ajustar o *score* de acordo com a desordem na sequência das palavras. Essa penalidade é calculada com base no número de correspondências e de blocos de palavras não consecutivas entre a tradução gerada e a referência, penalizando traduções com ordem de palavras significativamente diferente da esperada.

Além de métricas baseadas em sobreposição léxica, como BLEU e ROUGE, o **MoverScore**[10] [Zhao et al., 2019] e o **BERTScore**[11] [Zhang et al., 2020] representam métricas avançadas que utilizam *embeddings* contextuais para avaliar a similaridade semântica entre textos gerados por LLMs e suas referências. Ambas iniciam com a geração

[10]Disponível em `https://github.com/AIPHES/emnlp19-moverscore`.
[11]Disponível em `https://github.com/Tiiiger/bert_score`.

de *embeddings* para cada palavra dos textos candidato e de referência, geralmente empregando modelos pré-treinados como o BERT. O MoverScore então se diferencia ao calcular distâncias ponderadas entre pares de *embeddings* e resolver um problema de transporte ótimo (*Earth Mover's Distance* - EMD). Esse problema busca o custo mínimo para mover a massa semântica das palavras do texto candidato para a distribuição de palavras do texto de referência, quantificando o esforço necessário para que os significados do texto gerado se alinhem com os da referência. Em contraste, o BERTScore calcula a similaridade de cosseno entre cada *token* do candidato e seu *token* mais similar na referência, agregando esses *scores* em métricas de precisão, *recall* e *F1-score*. Enquanto o MoverScore se concentra no custo de transporte semântico global, o BERTScore avalia a similaridade contextual de *tokens* individuais e sua agregação.

O critério de **calibração** busca medir a correspondência entre a confiança preditiva do modelo e a acurácia observada de suas saídas. Um modelo é bem calibrado quando a confiança atribuída reflete com precisão a proporção de acertos. Assim, ao atribuir 80% de confiança a um conjunto de predições, espera-se que 80% estejam corretas. A ausência de calibração, manifestada por superconfiança ou subconfiança, compromete a confiabilidade do modelo, especialmente em aplicações críticas.

Há ainda métricas de erro que podem ser utilizadas na avaliação de LLMs. O **Erro de Calibração Esperado** (*Expected Calibration Error* – ECE) é uma das métricas comumente empregadas para mensurar o desempenho de calibração de modelos preditivos. Seu cálculo envolve a discretização das predições do modelo em *M bins* de confiança (B_m), em que cada *bin* agrupa previsões com níveis de confiança semelhantes. Para cada *bin*, computa-se a precisão média (acc) e a confiança média ($conf$). A métrica é definida como a média ponderada das diferenças absolutas entre a precisão e a confiança em cada *bin*, dada por

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |acc(B_m) - conf(B_m)|,$$

em que N é o número total de previsões e $|B_m|$ é o número de previsões no *bin* B_m. Tian *et al*. aplicaram o ECE para analisar a calibração de LLMs otimizados via *Reinforcement Learning from Human Feedback* (RLHF), incluindo modelos como ChatGPT, GPT-4, Claude 1, Claude 2 e Llama2 [Tian et al., 2023].

Complementando a ECE, o **Erro de Calibração Máximo** (*Maximum Calibration Error* – MCE) foca o pior erro de calibração observado. O MCE identifica o maior desvio absoluto entre a precisão e a confiança em qualquer um dos *bins* de confiança. O MCE é particularmente indicado para identificar cenários em que o modelo exibe a maior descalibração, seja por excesso ou falta de confiança, fornecendo *insights* sobre os pontos mais críticos da robustez do modelo. A métrica é dada por

$$\mathrm{MCE} = \max_{m \in \{1,\ldots,M\}} |acc(B_m) - conf(B_m)|.$$

O critério de **robustez** avalia a capacidade de um modelo manter seu desempenho e funcionalidade sob condições de entrada desafiadoras ou perturbadas. Tal avaliação en-

globa a resiliência do modelo a ataques adversariais, eventuais mudanças na distribuição dos dados e a presença de ruído ou variações na entrada. Sendo assim, a robustez é um critério fundamental para garantir a confiabilidade e a segurança dos modelos em ambientes do mundo real, nos quais os dados de entrada podem não ser idênticos aos dados de treinamento ou podem ser manipulados intencionalmente. Dentre as métricas fundamentais baseadas nesse critério, destacam-se a taxa de sucesso de ataque e a taxa de queda de desempenho.

A **Taxa de Sucesso de Ataque** (*Attack Success Rate* – ASR) serve como uma métrica primordial para quantificar a robustez adversarial em LLMs [Wang et al., 2023]. Especificamente, para um conjunto de dados de validação $D = \{(x_i, y_i)\}_{i=1}^{N}$, onde x_i é a amostra de entrada e y_i é sua verdade fundamental (*ground truth*) correspondente, e dado um método de ataque adversarial A, o ASR mede a proporção de ataques bem-sucedidos. Um ataque é considerado bem-sucedido quando, para uma entrada original x que o modelo substituto f classifica corretamente ($f(x) = y$), o exemplo adversarial gerado $A(x)$ induz o modelo a produzir uma predição incorreta ($f(A(x)) \neq y$). A taxa de sucesso é calculada como

$$ASR = \frac{\sum_{(x,y)\in D} I[f(A(x)) \neq y \text{ e } f(x) = y]}{\sum_{(x,y)\in D} I[f(x) = y]},$$

em que $I[\cdot]$ é a função indicadora, que retorna 1 se a condição dentro dos colchetes for verdadeira e 0 caso contrário. Vale ressaltar que o numerador deve considerar apenas os casos onde o modelo f inicialmente acerta a previsão para a entrada original x, e o ataque é bem-sucedido ao induzir um erro.

A **Taxa de Queda de Desempenho** (*Performance Drop Rate* – PDR) é uma métrica unificada e eficaz para avaliar a robustez do *prompt* em LLMs [Zhu et al., 2024a]. Essa métrica quantifica a degradação relativa do desempenho do modelo após a aplicação de um ataque adversarial ao *prompt*. A degradação é medida comparando o desempenho do modelo com o *prompt* original e com o *prompt* atacado. A PDR é calculada por

$$PDR = 1 - \frac{\sum_{(x,y)\in D} M[f([A(P), x]), y]}{\sum_{(x,y)\in D} M[f([P, x]), y]},$$

em que A representa a função de ataque adversarial aplicada ao *prompt*, P denota a concatenação do *prompt* com a entrada x e M é a função de avaliação de desempenho que varia de acordo com a tarefa específica. Uma PDR elevada indica uma baixa robustez do modelo a alterações no *prompt*.

4.6.2. Critérios e Métricas de Avaliação da Qualidade Operacional

Paralelamente à avaliação da qualidade cognitiva, o desempenho operacional de modelos de linguagem é avaliado com base em métricas que refletem custo computacional, eficiência de execução e viabilidade de implantação. Tais métricas podem ser agrupadas em dois conjuntos principais, (i) aquelas dedicadas a quantificar o custo e eficiência computacional e (ii) aquelas voltadas à medição do desempenho em execução [Chang et al., 2024, Zhu et al., 2024b]. Dentre as métricas focadas em custo e eficiência, a mais

elementar é o **Tamanho do Modelo**, que é expresso pelo número total de parâmetros treináveis e influencia diretamente os requisitos de memória e o custo de armazenamento. Logo, modelos maiores não apenas demandam maior capacidade de armazenamento, mas também exigem mais recursos computacionais tanto para treinamento quanto para inferência. Essa relação entre tamanho e custo computacional pode ser quantificada por meio das **Operações de Ponto Flutuante** (*Floating Point Operations* – FLOPs), que medem a complexidade computacional do modelo, representando o número estimado de operações aritméticas necessárias para processar uma entrada. A redução dos FLOPs é um indicador de maior eficiência computacional, especialmente em modelos comprimidos, pois implica menor carga de processamento. No entanto, a mera contagem de FLOPs não captura totalmente a eficiência prática do modelo. Para avaliar quão bem o hardware disponível é aproveitado, calcula-se a **Utilização Média de FLOPs** (*Mean FLOPs Utilization* - MFU). Essa métrica compara os FLOPs efetivamente utilizados com a capacidade máxima teórica do hardware por ciclo de *clock*, oferecendo uma medida direta da eficiência computacional. Valores elevados de MFU indicam que o modelo está fazendo uso otimizado dos recursos disponíveis, enquanto valores baixos sugerem gargalos ou subutilização.

No segundo grupo de métricas, o **Tempo de Inferência** destaca-se por medir a latência entre a entrada do dado e a geração da resposta pelo modelo. Essa métrica é essencial em aplicações em tempo real e é influenciada pela arquitetura do modelo, paralelismo e otimizações no pipeline de execução. A **Taxa de Aceleração** (T_A) quantifica o ganho de desempenho entre dois modelos distintos, normalmente entre um modelo original e sua versão comprimida. É definida como $T_A = \frac{T_{\text{original}}}{T_{\text{compactado}}}$ em que T_{original} e $T_{\text{compactado}}$ são os tempos médios de inferência de cada modelo. Em contrapartida, a **Taxa de Compactação** (T_C) expressa a economia de espaço após compressão do modelo. Tal métrica é matematicamente definida como $T_C = \frac{S_{\text{original}}}{S_{\text{compactado}}}$, em que S representa o tamanho do modelo, em número de parâmetros ou bytes. Altas taxas de compactação são desejáveis para viabilizar o uso de modelos em dispositivos com recursos limitados.

4.6.3. Arcabouços Públicos de Avaliação

Ao adotar a abordagem baseada no desempenho operacional, é comum a aplicação de *benchmarks* e protocolos de avaliação padronizados. Ferramentas como MMLU, BIG-bench e HELM estabelecem metodologias rigorosas para mensurar o desempenho desses modelos em diferentes cenários, analisando sua capacidade de gerar respostas coerentes, seguras e alinhadas a diretrizes éticas. Esses *benchmarks* estruturam conjuntos de testes abrangentes, permitindo comparações objetivas entre diferentes abordagens e refinamentos sucessivos nos modelos.

O **MMLU**[12] (*Massive Multitask Language Understanding*), proposto por Hendrycks *et. al*, compreende um *benchmark* robusto para a avaliação da capacidade de modelos de linguagem na assimilação e aplicação de conhecimento em múltiplos domínios durante o pré-treinamento dos modelos [Hendrycks et al., 2020]. A abordagem central do MMLU reside na avaliação exclusiva em cenários *zero-shot* e *few-shot*, simulando de forma mais fidedigna a avaliação cognitiva humana, que dispensa treinamento

[12]Disponível em `https://github.com/hendrycks/test`.

explícito para cada tarefa específica. Diferentemente dos demais *benchmarks*, o MMLU distingue-se por dois fatores: (i) abrangência multidisciplinar, englobando 57 áreas de conhecimento e (ii) variedade de dificuldade das questões, transitando do nível elementar ao profissional avançado e aferindo tanto o conhecimento geral quanto a habilidade de resolução de problemas. Metodologicamente, o MMLU é construído como um teste massivo e multitarefa, compreendendo questões de múltipla escolha curadas manualmente a partir de fontes abertas, como exames de pós-graduação, licenciamento profissional e materiais de cursos universitários. O *dataset* totaliza 15.908 questões, distribuídas entre um conjunto de desenvolvimento para validação *few-shot* e um conjunto de teste, assegurando um mínimo de 100 exemplos por assunto.

O **BIG-bench**[13] constitui um *benchmark* colaborativo concebido para uma investigação multifacetada de LLMs existentes [Suzgun et al., 2023]. Composto por 204 tarefas variadas, cada qual contendo múltiplas questões, o BIG-bench permite avaliar aspectos diversos das capacidades dos modelos. Devido ao elevado custo computacional associado à avaliação do *benchmark* completo, uma versão mais leve foi desenvolvida, o **BIG-bench-Lite**, reduzindo o escopo a apenas 24 tarefas. Adicionalmente, o **BIG-bench Hard**[14] (BBH) foi proposto com o objetivo de investigar as tarefas que permanecem insolúveis para os LLMs, selecionando especificamente aquelas nas quais os modelos exibem desempenho inferior ao humano. Dada a elevada dificuldade do BBH, modelos de menor escala frequentemente apresentam desempenho próximo ao aleatório.

O **HELM**[15] (Holistic Evaluation of Language Models) destaca-se como um *benchmark* abrangente por adotar uma abordagem holística e padronizada na avaliação de modelos de linguagem [Liang et al., 2023]. Diferentemente de *benchmarks* tradicionais, que frequentemente se restringem a métricas isoladas como acurácia, o HELM propõe uma estrutura taxonômica que abrange 16 cenários centrais, organizados por tarefa, domínio e idioma, e 7 categorias de métricas, incluindo robustez, justiça, viés e estereótipos, toxicidade, calibração e eficiência. Seu principal diferencial está na capacidade de identificar limitações críticas dos modelos, como quedas acentuadas de desempenho em testes de robustez ou a sub-representação de dialetos não padrão, viabilizando pela primeira vez comparações diretas entre modelos que anteriormente eram avaliados de forma fragmentada.

4.7. Atividade Prática

A atividade prática proposta tem como objetivo principal explorar o processo de quantização de modelos de linguagem e realizar avaliações comparativas entre modelos quantizados e modelos compactos previamente gerados. A atividade é conduzida ao longo de três etapas complementares, descritas e acessíveis em um repositório público no GitHub, disponível em `https://gitlab.com/LabGen/minicurso`. No repositório estão os tutoriais e os códigos para a execução da atividade prática. A **primeira etapa** consiste em aplicar a técnica de Quantização Pós-Treinamento (PTQ) sobre modelos de linguagem já treinados e realizar avaliações comparativas com modelos compactos

[13]Disponível em `https://github.com/google/BIG-bench`.
[14]Disponível em `https://github.com/suzgunmirac/BIG-Bench-Hard`.
[15]Disponível em `https://crfm.stanford.edu/helm/v0.1.0`.

previamente gerados. Para isso, utiliza-se a biblioteca `PyTorch`[16], que oferece suporte robusto para a quantização dinâmica de modelos. Em particular, o modelo adotado nesta etapa é o DistilBERT, uma versão mais leve do BERT, já ajustada para tarefas de análise de sentimento, disponível via `Hugging Face Transformers`[17]. Esse modelo foi treinado sobre o conjunto de dados SST-2 (*Stanford Sentiment Treebank*), amplamente utilizado em *benchmarks* de classificação para análise de sentimentos. A técnica de quantização consiste na conversão dos pesos do modelo de precisão `FP32` para o formato `INT8`, utilizando o `Torch Quantization Toolkit`[18], que permite aplicar quantização dinâmica sobre as camadas lineares do modelo. Durante essa etapa, explora-se a redução de tamanho do modelo, o impacto sobre o tempo de inferência e os efeitos na precisão dos resultados. Embora outras abordagens como o Treinamento com Reconhecimento de Quantização (QAT) possam ser aplicadas, nesta primeira fase o foco está na praticidade e acessibilidade da PTQ. A avaliação é então conduzida com base em tarefas pré-definidas de classificação de texto, utilizando como métrica principal a acurácia, além de indicadores adicionais como o tempo médio de inferência, tamanho final do modelo em disco e a diferença absoluta nas probabilidades preditas entre os modelos original e quantizado. Para o cálculo dessas métricas, utiliza-se a biblioteca `Scikit-learn`[19], bem como ferramentas auxiliares do `PyTorch`.

A **segunda etapa** foca a incorporação dos modelos de linguagem personalizados em dispositivos com restrições de recursos. Tal procedimento é demonstrado utilizando um Raspberry Pi 4[20], com ênfase na execução local e eficiente dos modelos utilizando apenas a CPU. Para isso, adota-se a combinação do ambiente Ollama com a interface interativa Open WebUI, ambos executados de forma integrada em um único contêiner Docker. A abordagem adotada consiste na utilização da imagem integrada disponibilizada pelo projeto Open WebUI [21], que encapsula tanto o servidor Ollama quanto a interface Open WebUI em um único contêiner. Isso permite uma instalação simplificada e automatizada, sem necessidade de configurar múltiplos serviços ou dependências. Como o Raspberry Pi 4 não possui unidade de processamento gráfico dedicada compatível com os requisitos do Ollama, é utilizada a opção de instalação voltada exclusivamente para execução em CPU. Após a instalação, é possível carregar modelos compactos, como o TinyOllama [22], disponibilizados pelo próprio repositório do Ollama ou gerados na etapa anterior, e utilizá-los de forma responsiva e acessível mesmo em um ambiente de hardware limitado.

A **terceira etapa** prevê a personalização e utilização dos modelos otimizados nas etapas anteriores, via OpenWebUI no Raspeberry Pi. Essa plataforma oferece uma interface amigável para editar e testar fluxo de processamento dos modelos quantizados e não quantizados inseridos na plataforma. Internamente, o Open WebUI oferece recursos para o desenvolvimento de elementos de processamento, o que permite a personalização

[16]Disponível em `https://pytorch.org/`.

[17]Disponível em `https://huggingface.co/docs/transformers/index`.

[18]Disponível em `https://docs.nvidia.com/deeplearning/tensorrt/archives/tensorrt-861/pytorch-quantization-toolkit/docs/index.html`.

[19]Disponível em `https://scikit-learn.org/stable/`.

[20]Disponível em `https://www.raspberrypi.com/`.

[21]Disponível em `https://openwebui.com/`.

[22]Disponível em `https://github.com/jzhang38/TinyLlama`.

e a extensão das funcionalidades da plataforma. Esses elementos incluem: (i) *filter*, elemento que permite filtrar e pré-processar dados de entrada para os modelos; (ii) *pipeline*, elemento que facilita a criação de fluxos de trabalho complexos, combinando diferentes elementos de processamento como *filters*. Quando o *pipeline* de filtro é ativado, a mensagem de entrada (*inlet*) é processada antes do LLM e a saída (*outlet*) após, personalizando a interação; (iii) *tool*, elemento que possibilita a integração de ferramentas externas para enriquecer as capacidades dos modelos; (iv) *function*, elemento que permite a definição de funções personalizadas para manipulação de dados e execução de tarefas específicas.

Com essa sequência de etapas, a atividade prática permite compreender os fundamentos da quantização e sua aplicação em modelos de linguagem e também demonstra, de forma concreta, a viabilidade da personalização e execução eficiente desses modelos em plataformas embarcadas com recursos limitados.

4.8. Desafios e Tendências de Pesquisa

Os LLMs transformaram significativamente o processamento de linguagem natural e a inteligência artificial generativa. No entanto, essa transformação trouxe uma série de desafios técnicos e éticos. Um dos principais desafios técnicos está relacionado ao elevado custo computacional e energético envolvido tanto no treinamento quanto na inferência dos modelos. LLMs exigem grande quantidade de memória e capacidade de processamento, o que restringe sua adoção ampla. A escalabilidade de LLMs tem sido acompanhada de custos ambientais e operacionais crescentes, com grandes modelos consumindo megawatts-hora [Argerich e Patiño-Martínez, 2024, de Vries, 2023]. A otimização desses modelos é essencial para reduzir o custo operacional e permitir sua escalabilidade. Nesse contexto, técnicas de quantização, modelos esparsos, mistura de especialistas (*Mixture of Experts* - MoE), poda, destilação, e uso de hardware especializado (e.g., FPGAs, TPUs, ACAPs), vêm ganhando destaque como abordagens promissoras para melhorar a eficiência computacional de LLMs e consequente escalabilidade.

A inferência de LLMs é limitada por sua natureza autoregressiva e pelo elevado consumo computacional. Reduzir a latência e aumentar a eficiência é essencial para aplicações em tempo real [Huang et al., 2024]. Estratégias como a utilização eficiente de parâmetros, por meio da ativação seletiva de subconjuntos do modelo, permitem acelerar a geração de respostas com menor custo computacional. Para tal, é necessário estabelecer critérios bem definidos que permitam encerrar antecipadamente o processo de decodificação, mantendo a qualidade da saída ao mesmo tempo em que economiza recursos computacionais [Huang et al., 2024]. A esparsidade contextual também possibilita ignorar pesos irrelevantes dinamicamente durante a inferência, sem comprometer a acurácia. A Mistura de Especialistas (*Mixture of Experts* - MoE) expande esse princípio, ativando apenas os submodelos especializados mais relevantes para cada entrada, o que permite escalar a capacidade do modelo com eficiência. A otimização da memória cache de chaves e valores em LLMs busca reduzir os custos associados ao armazenamento e gerenciamento de chaves e valores previamente calculados, especialmente em tarefas complexas como cadeia de pensamento e recuperação de informações [Huang et al., 2024].

A quantização de LLMs lida com o desafio adicional de manter a confiabilidade estatística dos resultados, mesmo em ambientes computacionais restritos. Um obstáculo

nesse sentido é o desenvolvimento de algoritmos de inferência escaláveis e confiáveis, superando limitações de métodos como Monte Carlo via Cadeias de Markov (MCMC), que oferecem boa precisão, mas sofrem em escalabilidade. Técnicas recentes como subamostragem de dados e paralelismo assíncrono buscam mitigar esses problemas, garantindo eficiência mesmo em cenários complexos[23].

Outro desafio é integrar computação aproximada para melhorar a eficiência energética, equilibrando precisão e economia de energia, com avanços em modelos de programação, ferramentas de teste, suporte sistêmico e arquiteturas de hardware. Baseado na ideia de que nem todas as tarefas computacionais requerem precisão absoluta, há projetos que exploram o paradigma de computação aproximada para reduzir o consumo de energia e recursos computacionais[24]. O projeto SHF, desenvolvido pela Washington University e apoiado pela NSF/EUA, é inspirado em um modelo cognitivo que afirma que o cérebro humano realiza raciocínio rápido e simples para a maioria das tarefas e executa um processamento detalhado apenas quando necessário. O projeto desenvolveu técnicas avançadas de compilação automática para modelos quantizados, com suporte a tensores esparsos e seleção adaptativa de bibliotecas otimizadas, que reduzem o consumo energético de LLMs em execução em múltiplas plataformas. Essas inovações foram integradas ao Apache TVM, um dos compiladores mais amplamente utilizados atualmente para aprendizado de máquina, o qual originou a *startup* OctoML. A otimização permitiu que modelos de linguagem sejam compilados e otimizados para execução eficiente em dispositivos com restrições energéticas, como FPGAs e NPUs, bem como em nuvens escaláveis e energeticamente conscientes.

No contexto científico, o projeto A3D3 desenvolve algoritmos personalizados de IA integrados a plataformas heterogêneas, como GPUs e FPGAs, para acelerar o processamento em tempo real em áreas como física de altas energias, astrofísica e neurociência[25]. Em física de partículas, técnicas como Redes Neurais de Grafos (*Graph Neural Networks* - GNNs) de baixa latência em FPGAs foram exploradas para rastreamento de trajetórias em experimentos como o ProtoDUNE [Cai et al., 2023]. A implementação de inferência semântica em tempo real com `hls4ml` possibilitou aplicações autônomas de IA embarcada em física e veículos autônomos [Ghielmetti et al., 2022, Khoda et al., 2023]. Na área de ondas gravitacionais, ecossistemas de software e inferência acelerada por hardware permitiram implantar aprendizado profundo com alta eficiência para detectar sinais astrofísicos em tempo real [Gunny et al., 2022]. Em colaboração com o experimento ATLAS, foi demonstrada a viabilidade de avaliação de inferência de aprendizado de máquina em FPGAs no contexto do sistema de gatilho de eventos [Jiang et al., 2024]. O projeto contribuiu também para o avanço de algoritmos interpretáveis e escaláveis de aprendizado em grafos, como os baseados em atenção estocástica e representação temporal de redes, com aplicações em múltiplos domínios científicos [Miao et al., 2022].

Além das técnicas de otimização, surgem desafios na gestão eficiente de cargas de trabalho complexas de aprendizado profundo em ambientes de centros de dados, com foco no agendamento de tarefas dinâmicas e heterogêneas. Nesse cenário, há projetos cujo ob-

[23]Disponível em `https://www.nsf.gov/awardsearch/showAward?AWD_ID=2046760`.
[24]Disponível em `https://www.nsf.gov/awardsearch/showAward?AWD_ID=1518703`.
[25]Disponível em `https://www.nsf.gov/awardsearch/showAward?AWD_ID=2117997`.

jetivo central é o desenvolvimento de um sistema de software intuitivo e eficiente, capaz de realizar, com apenas um clique, a configuração e execução de treinamentos distribuídos de modelos de aprendizado de máquina e aprendizado profundo, inclusive modelos fundacionais de larga escala[26]. A proposta foca no desafio de gerenciar automaticamente recursos computacionais em ambientes multiusuário e heterogêneos, otimizando desempenho, eficiência estatística e uso de infraestrutura. O projeto desenvolveu dois sistemas: Alpa, para gerenciar o treinamento e a inferência de LLMs, e Pollux, para otimizar o agendamento de tarefas em *clusters* de GPUs. O sistema Alpa constrói planos hierárquicos de execução paralela e aplica passes de compilação para gerar estratégias eficientes de execução, tanto em termos de computação quanto de memória. Já o sistema Pollux utiliza o conceito de *goodput* para redistribuir recursos dinamicamente, promovendo uma utilização justa e otimizada do *cluster*. O projeto também desenvolveu o AMP, um arcabouço que automatiza a identificação e a aplicação de estratégias eficazes de paralelismo de modelo, selecionando a estratégia mais adequada utilizando um modelo de custo sensível à diversidade das estruturas dos modelos e às especificações do *cluster* de computação. O arcabouço se destaca na gestão de modelos complexos com camadas estruturalmente irregulares e em *clusters* que combinam diferentes gerações de GPUs. Outra ferramenta desenvolvida no projeto é a Redco, que simplifica a configuração de treinamento distribuído para LLMs, reduzindo a necessidade de conhecimento técnico aprofundado. Entre as principais funcionalidades do Redco estão regras pré-definidas para a distribuição dos modelos e funções personalizáveis que otimizam os fluxos de trabalho em aprendizado de máquina. Também foram desenvolvidos algoritmos que aumentam a velocidade e a eficiência tanto no pré-treinamento quanto no ajuste fino de modelos fundacionais, bem como softwares voltados para desafios específicos, como inferência de modelo privado e problemas complexos de meta-aprendizado.

Questões de privacidade e uso ético de dados emergem como barreiras críticas [Jiao et al., 2025]. Os LLMs frequentemente são treinados com informações sensíveis, muitas vezes sem consentimento explícito. Durante a inferência, o modelo também pode ter acesso a dados sensíveis fornecidos pelo usuário. Contudo, o acesso a esses dados, seja durante o treinamento ou durante a inferência, fere a privacidade do proprietário dos dados. Existe risco de vazamento de dados, uma vez que os modelos podem aprender e repetir informações sensíveis presentes nos dados de treinamento, como nomes, e-mails e números de documentos. Os dados fornecidos durante o uso do modelo para inferência podem ser armazenados pelo proprietário do modelo para usos diversos, sendo possível haver vazamento do histórico de interações com o modelo. Casos como o da plataforma DeepSeek ilustram esses riscos[27]. Em janeiro de 2025, um banco de dados ligado à plataforma DeepSeek estava publicamente acessível, não sendo necessário autenticação para acessar as informações. O banco de dados continha histórico de bate-papo dos usuários, dados de *back-end* e informações confidenciais, incluindo fluxos de registros, segredos de API e detalhes operacionais da plataforma. O uso de conteúdos protegidos por direitos autorais no treinamento dos modelos pode gerar processos judiciais. No início de 2025, uma tendência utilizando o gerador de imagens da OpenAI surgiu nas redes sociais

[26]Disponível em `https://www.nsf.gov/awardsearch/showAward?AWD_ID=2008248`.

[27]Disponível em `https://www.wiz.io/blog/wiz-research-uncovers-exposed-deepseek-database-leak`.

para criar imagens com traços característicos do Studio Ghibli[28], levando a uma discussão intensa sobre questões éticas do uso de propriedade intelectual para treinamento de LLMs. Apesar de não se ter conhecimento sobre a abertura de processo judicial do Studio Ghibli contra a OpenAI, outros casos que infringem direitos de uso já resultarem em processos judiciais[29,30]. Soluções emergentes para proteção de dados sensíveis incluem o treinamento com privacidade diferencial e o uso de técnicas de privacidade computacional como aprendizado federado. A implementação de filtros e guarda-corpos (*guardrail*) durante o refinamento e a geração das respostas são medidas que podem ser tomadas para mitigar o problema da privacidade.

Para além de preocupações com a privacidade e direitos autorais, a implementação maciça dos LLMs pode exacerbar desigualdades preexistentes e gerar impactos sistêmicos indesejados [Weidinger et al., 2022]. A concentração hegemônica do desenvolvimento de LLMs em um número restrito de corporações e nações tende a acentuar a assimetria tecnológica global, resultando em uma dependência crítica de infraestruturas computacionais e cognitivas centralizadas. A adoção acelerada de LLMs por entidades públicas e privadas, frequentemente desprovida de avaliações de impacto abrangentes ou de mecanismos robustos de responsabilização, arrisca comprometer direitos individuais e coletivos, particularmente em contextos caracterizados por arcabouços regulatórios incipientes. Adicionalmente, identificam-se riscos intrínsecos relacionados à obsolescência programada de competências humanas, com potenciais repercussões adversas sobre os mercados de trabalho, processos educacionais e a autonomia dos indivíduos diante de sistemas automatizados opacos [Deranty e Corbin, 2024]. A aplicação indiscriminada de LLMs como substituto do julgamento humano, mormente em domínios sensíveis, pode culminar na desumanização de processos decisórios e na erosão de mecanismos tradicionais de validação, revisão e contestação. Na ausência de estruturas de governança transparentes e explicitamente orientadas pelo interesse público, a incorporação ubíqua desses modelos corre o risco de intensificar dinâmicas de exclusão, vigilância e controle algorítmico, transmutando desafios técnicos em questões críticas de justiça social."

Outro risco à utilização de modelos de linguagem de grande escala e genéricos é injeção de *prompts* (*prompt injection*). A injeção de *prompt* é considerada uma das principais vulnerabilidades de segurança relacionadas a modelos de linguagem de grande escala, sendo classificada pela OWASP como a ameaça mais crítica nesse contexto[31]. Esse tipo de ataque ocorre quando entradas enviadas ao modelo são manipuladas com instruções maliciosas capazes de burlar a intenção original da aplicação. O problema se agrava em sistemas que utilizam técnicas de *retrieval-augmented generation* (RAG), nos quais a separação entre dados e instruções pode se tornar nebulosa. Para reduzir os riscos associados ao uso de LLMs, sobretudo no que se refere à segurança e ao uso indevido, filtros e guarda-corpos (*guardrail*) são essenciais. Entre elas estão os filtros

[28]Disponível em `https://www.forbes.com/sites/danidiplacido/2025/03/27/the-ai-generated-studio-ghibli-trend-explained/`

[29]Disponível em `https://www.theguardian.com/books/2025/apr/04/us-authors-copyright-lawsuits-against-openai-and-microsoft-combined-in-new-york-with-newspaper-actions`

[30]Disponível em `https://originality.ai/blog/openai-chatgpt-lawsuit-list`.

[31]Disponível em `https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf`.

de segurança implementados em modelos como DALL-E 2 e Midjourney, que impedem a geração de conteúdos impróprios (*Not Safe for Work* - NSFW); modelos especializados em segurança, como o LlamaGuard da Meta, capazes de classificar entradas e saídas como seguras ou não; detectores específicos para conteúdos visuais, como os *CLIP-based NSFW detectors*; e técnicas de *watermarking*, voltadas à rastreabilidade e autenticação de mídias geradas por IA. Apesar desses esforços, os mecanismos de proteção ainda apresentam fragilidades, já que muitos ainda são susceptíveis a ataques de *prompt injection* mais elaborados, o que torna a evolução contínua dessas defesas um tema central nas pesquisas atuais sobre segurança em LLMs [Divakaran e Peddinti, 2024].

Paralelamente, na esfera social, os LLMs podem amplificar estereótipos e refletir desigualdades sociais encontradas nos dados utilizados para treinamento. A amplificação de vieses sociais presentes nos dados resulta, por exemplo, na geração de conteúdo discriminatório ou ofensivo, e no reforço de assimetrias de gênero, raça e classe [Gross, 2023, Kotek et al., 2023]. É preocupante a possibilidade de disseminação de desinformação e informações falsas, dada a capacidade dos LLMs de gerar conteúdos convincentes, porém potencialmente falsos ou enganosos. A disseminação desse tipo de informação pode ser feita para manipular a opinião pública ou criar materiais fraudulentos, o que pode ter consequências graves para a saúde pública, a democracia e a harmonia social [Henderson et al., 2018, Cortiz e Zubiaga, 2021]. A mitigação dos efeitos do viés social inserido nos LLMs requer a utilização de técnicas de desenviesamento no treinamento e inferência, avaliações contínuas com métricas de justiça e explicabilidade e uma curadoria ativa dos conjuntos de dados. Nesse sentido, Xue *et al.* criaram um conjunto de dados, denominado OccuQuest [Xue et al., 2023] a fim de refinar LLMs para remover o enviesamento do modelo. Os autores focam no viés social relacionado a profissões, que limita a capacidade dos LLMs ajustados por instruções de gerar respostas úteis para perguntas profissionais específicas. Aplicando esse conjunto de dados ao LLaMa, o modelo obtido superou significativamente variantes do LLaMA de última geração (como Vicuna, Tulu e WizardLM) em perguntas profissionais, segundo avaliações do GPT-4 e avaliações humanas [Xue et al., 2023].

A incorporação de LLMs em sistemas embarcados e infraestruturas críticas traz preocupações adicionais [Jiao et al., 2025]. Diagnósticos médicos, sentenças judiciais, controle e gerenciamento de sistemas em áreas essenciais, como energia, água, gás e telecomunicações, realizados de forma autônoma por LLMs são tarefas potencialmente arriscadas, pois envolvem decisões de alto impacto que exigem precisão, responsabilidade legal e consideração ética, características que os LLMs, por sua natureza estatística e opaca, não são capazes de garantir plenamente. Há ainda o risco de que os usuários desenvolvam uma dependência excessiva ou confiança acrítica nas recomendações geradas automaticamente, o que pode comprometer sua autonomia decisória e contribuir para sobrecarga cognitiva. Modelos interpretáveis e explicáveis, além de interfaces humano-no-circuito (*human-in-the-loop*), são fundamentais para mitigar esses riscos.

A fim de mitigar o uso indiscriminado dos LLMs, a regulação de LLMs está em rápida evolução globalmente, com legislações como: AI Act, na União Europeia, Executive Order 14110, nos Estados Unidos, e o Marco Legal da IA, no Brasil. AI Act[32]

[32]Disponível em `https://artificialintelligenceact.eu/`.

classifica sistemas de IA por risco e impõe obrigações específicas para modelos fundacionais. Executive Order 14110[33] foca segurança, privacidade e inovação responsável. O Marco Legal da IA[34] está em discussão no Brasil e busca equilibrar inovação com direitos fundamentais. Essas regulações pressionam por transparência, rastreabilidade e controle de uso, exigindo que desenvolvedores e operadores se adaptem a requisitos de conformidade, governança e auditoria algorítmica.

Há também iniciativas de auto-regulação desenvolvidas por empresas do setor. O *Frontier Model Forum*[35] e a *Coalition for Content Provenance and Authenticity* (C2PA)[36] são duas iniciativas interempresariais que buscam promover a segurança, a transparência e a confiança no uso de tecnologias baseadas em inteligência artificial e na produção de conteúdos digitais [Divakaran e Peddinti, 2024]. O *Frontier Model Forum* foi criado em 2023 pelas empresas Microsoft, Google, OpenAI e Anthropic com o objetivo de fomentar o desenvolvimento seguro e responsável de modelos de linguagem de fronteira, modelos de IA altamente avançados e com amplo poder de generalização. Entre suas metas estão o avanço da pesquisa em segurança para IA, o estabelecimento de melhores práticas para desenvolvimento e implantação desses modelos e o incentivo à colaboração entre setor privado, governos, academia e sociedade civil. A iniciativa também visa apoiar aplicações de IA voltadas à resolução de grandes desafios sociais, como mudanças climáticas, saúde e segurança digital. Já a C2PA reúne empresas como Adobe, Microsoft, BBC, Intel e Truepic, com foco na criação de padrões técnicos para garantir a rastreabilidade e autenticidade de conteúdos digitais. Essa coalizão atua no desenvolvimento de uma infraestrutura padronizada que incorpora metadados criptograficamente protegidos às imagens, vídeos e documentos digitais, permitindo identificar quem criou o conteúdo, quando, onde e com quais ferramentas. A rastreabilidade é essencial no combate à desinformação e ao uso indevido de *deepfakes*, fornecendo mecanismos confiáveis de verificação de autenticidade para o público, plataformas e autoridades.

O desenvolvimento de LLMs exige recursos computacionais e conhecimentos multidisciplinares que extrapolam as capacidades isoladas de empresas ou instituições acadêmicas. Nesse cenário, parcerias estratégicas têm se mostrado cruciais. Iniciativas como o IBM-Illinois Discovery Accelerator[37] foi criada para ampliar o acesso à educação em tecnologia e ao desenvolvimento de habilidades, a fim de impulsionar avanços em áreas emergentes da tecnologia, incluindo computação em nuvem híbrida e inteligência artificial, computação quântica, descoberta acelerada de novos materiais e sustentabilidade, com o objetivo de acelerar a descoberta de soluções para desafios globais complexos. Iniciativas de *benchmarking* aberto e conjuntos de dados colaborativos promovem transparência e reprodutibilidade [Liu et al., 2024a].

As tendências de pesquisa em LLMs refletem esforços para superar desafios técni-

[33]Disponível em `https://www.govinfo.gov/app/details/CFR-2024-title3-vol1/CFR-2024-title3-vol1-eo14110/summary`.

[34]Disponível em `https://www25.senado.leg.br/web/atividade/materias/-/materia/157233`.

[35]Disponível em `https://www.frontiermodelforum.org/`.

[36]Disponível em `https://c2pa.org/`.

[37]Disponível em `https://discoveryacceleratorinstitute.grainger.illinois.edu/`

cos, energéticos e éticos da sua adoção em larga escala. Destacam-se avanços em eficiência computacional e energética com técnicas como quantização, poda, modelos esparsos, MoE e hardware especializado. Métodos de inferência seletiva e paralelismo buscam reduzir a latência e manter a confiabilidade estatística. A integração de computação aproximada e compiladores otimizados também tem possibilitado o uso eficiente de LLMs em ambientes restritos. Paralelamente, crescem as preocupações com privacidade, viés e uso indevido de dados, incentivando técnicas como privacidade diferencial e aprendizado federado. No campo regulatório, iniciativas internacionais visam equilibrar inovação com direitos fundamentais. Parcerias institucionais e práticas de ciência aberta têm se mostrado essenciais para o avanço responsável, ético e sustentável desses modelos. A expansão do uso de LLMs deve ser acompanhada por uma abordagem crítica e proativa quanto aos seus impactos éticos, sociais e ambientais. A combinação de regulação inteligente, arquitetura sustentável e colaboração interdisciplinar é essencial para garantir que esses modelos avancem com responsabilidade e tragam benefícios reais à sociedade.

4.9. Considerações Finais

O capítulo apresentou uma análise detalhada sobre os desafios e estratégias para viabilizar a execução de Modelos de Linguagem em Larga Escala (LLMs) em dispositivos com restrições computacionais, como smartphones, sistemas embarcados e equipamentos de borda. Destacou-se o impacto transformador dos LLMs no processamento de linguagem natural, com ênfase em suas capacidades de compreensão contextual, geração fluente e adaptação a diferentes domínios. Foram discutidos os fundamentos teóricos, as arquiteturas baseadas em *Transformers* e autoatenção, bem como técnicas de otimização voltadas à redução de consumo de memória, energia e tempo de inferência. A atividade prática, centrada em quantização e personalização, evidenciou os ganhos concretos dessas abordagens na implantação local de LLMs. Entre as técnicas abordadas, destacaram-se a quantização, a poda e a destilação de conhecimento, que compõem o núcleo das estratégias de compressão de modelos. A quantização, por exemplo, demonstrou-se eficaz ao reduzir a precisão numérica para otimizar memória e acelerar a inferência, com ênfase em métodos como AWQ e PTQ. A poda permitiu eliminar parâmetros redundantes, promovendo esparsidade e eficiência em hardware restrito. Já a destilação mostrou-se útil para transferir o desempenho de modelos maiores para arquiteturas menores, mantendo a qualidade das respostas. Também foram discutidas variantes arquiteturais, como os *Transformers* eficientes (*Linformer* e *Performer*), e mecanismos de atenção adaptativa, voltados à escalabilidade em tarefas extensas.

Como trabalhos futuros, três direções promissoras de pesquisa são vislumbradas. Primeiramente, há o desenvolvimento de técnicas de quantização e poda que incorporem o contexto semântico das tarefas de aplicação, promovendo uma compactação mais informada. Outra direção de pesquisas futuras é o desenvolvimento de LLMs integrados a sensores embarcados e entradas multimodais, que pode ampliar significativamente sua utilidade em sistemas ciberfísicos. Por fim, destaca-se a necessidade urgente de pesquisas voltadas à segurança e confiabilidade dos LLMs, sobretudo no que se refere ao alinhamento ético, mitigação de vieses, proteção contra usos maliciosos e o desenvolvimento de guarda-corpos (*guardrails*) autônomos e adaptáveis, essenciais para garantir o uso responsável de modelos em larga escala em dispositivos acessíveis.

Por fim, espera-se que as técnicas discutidas nesse capítulo sirvam de base para avanços futuros na área, permitindo que dispositivos móveis se beneficiem cada vez mais de modelos de linguagem robustos. Pesquisas contínuas em modelos mais eficientes e estratégias de alinhamento ético serão cruciais para consolidar a aplicação de Modelos de Linguagem de Larga Escala em dispositivos com hardware restrito.

Agradecimentos

Os autores agradecem aos alunos Gabriel de Almeida Campos, Ricardo Araujo de Souza Filho, Lucca Vieira Chatack e Igor Rodrigues Alves pelo apoio no desenvolvimento dos códigos para a atividade prática.

Referências

[Abdin et al., 2024] Abdin M, Aneja J, Awadalla H, Awadallah A, Awan A A, Bach N, et al. (2024). Phi-3 technical report: A highly capable language model locally on your phone. arXiv:2404.14219 v4 [cs.CL].

[Aftan e Shah, 2023] Aftan S, Shah H (2023). A survey on BERT and its applications. Em *2023 20th Learning and Technology Conference (LT)*, p. 161–166.

[Ainslie et al., 2023] Ainslie J, Lee Thorp J, de Jong M, Zemlyanskiy Y, Lebrón F, Sanghai S (2023). GQA: Training generalized multi-query transformer models from multi-head checkpoints. arXiv:2305.13245 [cs.CL].

[Anil et al., 2023] Anil R, Dai A M, Firat O, Johnson M, Lepikhin D, Passos A, et al. (2023). PaLM 2 technical report. arXiv:2305.10403 [cs.CL].

[Argerich e Patiño-Martínez, 2024] Argerich M F, Patiño Martínez M (2024). Measuring and improving the energy efficiency of large language models inference. *IEEE Access*, 12:80194–80207.

[Bahdanau et al., 2014] Bahdanau D, Cho K, Bengio Y (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bai et al., 2024] Bai X, Wang A, Sucholutsky I, Griffiths T L (2024). Measuring implicit bias in explicitly unbiased large language models. *arXiv preprint arXiv:2402.04105*.

[Banerjee e Lavie, 2005] Banerjee S, Lavie A (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. Em *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, p. 65–72.

[Baroni, 2021] Baroni M (2021). On the proper role of linguistically-oriented deep net analysis in linguistic theorizing. *CoRR*, abs/2106.08694.

[Cai et al., 2023] Cai T, Herner K, Yang T, Wang M, Acosta Flechas M, Harris P, et al. (2023). Accelerating machine learning inference with GPUs in ProtoDUNE data processing. *Computing and Software for Big Science*, 7(1).

[Chang et al., 2024] Chang Y, Wang X, Wang J, Wu Y, Yang L, Zhu K, et al. (2024). A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3).

[Chowdhery et al., 2023] Chowdhery A, Narang S, Devlin J, Bosma M, Mishra G, Roberts A, et al. (2023). PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

[Chung et al., 2024] Chung H W, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, et al. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

[Cortiz e Zubiaga, 2021] Cortiz D, Zubiaga A (2021). Ethical and technical challenges of AI in tackling hate speech. *The International Review of Information Ethics*, 29.

[Dagdelen et al., 2024] Dagdelen J, Dunn A, Lee S, Walker N, Rosen A S, Ceder G, Persson K A, Jain A (2024). Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

[Dao et al., 2022] Dao T, Fu D, Ermon S, Rudra A, Ré C (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. Em Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, editors, *Advances in Neural Information Processing Systems*, volume 35, p. 16344–16359. Curran Associates, Inc.

[de Oliveira et al., 2025] de Oliveira N R, Campos G, Rodrigues I, Watanabe J A C, Couto R S, Moraes I M, Medeiros D, Mattos D M F (2025). Quantization effects on large language models for intent-based network management. Em *Proceedings of the 17th International Conference on Network and Telecommunications Management Systems (NTMS)*, Paris, France. IEEE. A ser apresentado.

[de Vries, 2023] de Vries A (2023). The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194.

[Deranty e Corbin, 2024] Deranty J P, Corbin T (2024). Artificial intelligence and work: a critical review of recent research from the social sciences. *Ai & Society*, 39(2):675–691.

[Divakaran e Peddinti, 2024] Divakaran D M, Peddinti S T (2024). Large language models for cybersecurity: New opportunities. *IEEE Security & Privacy*, p. 2–9.

[Djeffal et al., 2023] Djeffal N, Kheddar H, Addou D, Mazari A C, Himeur Y (2023). Automatic speech recognition with BERT and CTC transformers: A review. Em *2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM)*, volume 1, p. 1–8.

[Dubey et al., 2024] Dubey A, Jauhri A, Pandey A, Kadian A, Al Dahle A, Letman A, et al. (2024). The Llama 3 herd of models.

[Falcon-LLM Team, 2024] Falcon-LLM Team (2024). The Falcon 3 family of open models. Relatório técnico, Technology Innovation Institute (TII).

[Fang et al., 2024] Fang G, Yin H, Muralidharan S, Heinrich G, Pool J, Kautz J, Molchanov P, Wang X (2024). MaskLLM: Learnable semi-structured sparsity for large language models. *arXiv preprint arXiv:2409.17481*.

[Frantar e Alistarh, 2023] Frantar E, Alistarh D (2023). SparseGPT: Massive language models can be accurately pruned in one-shot. Em *International Conference on Machine Learning*, p. 10323–10337. PMLR.

[Gemma Team et al., 2025] Gemma Team, Kamath A, Ferret J, Pathak S, Vieillard N, Merhej R, et al. (2025). Gemma 3 technical report. arXiv:2503.19786 [cs.CL].

[Ghielmetti et al., 2022] Ghielmetti N, Loncar V, Pierini M, Roed M, Summers S, Aarrestad T, et al. (2022). Real-time semantic segmentation on FPGAs for autonomous vehicles with hls4ml. *Machine Learning: Science and Technology*, 3(4):045011.

[Gou et al., 2021] Gou J, Yu B, Maybank S J, Tao D (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

[Gross, 2023] Gross N (2023). What ChatGPT tells us about gender: A cautionary tale about performativity and gender biases in AI. *Social Sciences*, 12(8):435.

[Gu et al., 2023] Gu Y, Dong L, Wei F, Huang M (2023). Minillm: Knowledge distillation of large language models. Em *The Twelfth International Conference on Learning Representations*.

[Gunny et al., 2022] Gunny A, Rankin D, Harris P, Katsavounidis E, Marx E, Saleem M, Coughlin M, Benoit W (2022). A software ecosystem for deploying deep learning in gravitational wave physics. Em *Proceedings of the 12th Workshop on AI and Scientific Computing at Scale using Flexible Computing Infrastructures*, HPDC '22, p. 9–17. ACM.

[Guo et al., 2025] Guo D, Yang D, Zhang H, Song J, Zhang R, Xu R, et al. (2025). DeepSeek-R1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

[Henderson et al., 2018] Henderson P, Sinha K, Angelard Gontier N, Ke N R, Fried G, Lowe R, Pineau J (2018). Ethical challenges in data-driven dialogue systems. Em *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, p. 123–129, New York, NY, USA. Association for Computing Machinery.

[Hendrycks et al., 2020] Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, Steinhardt J (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

[Huang et al., 2024] Huang Y, Wan L J, Ye H, Jha M, Wang J, Li Y, Zhang X, Chen D (2024). Invited: New solutions on LLM acceleration, optimization, and application. Em *Proceedings of the 61st ACM/IEEE Design Automation Conference*, DAC '24, New York, NY, USA. Association for Computing Machinery.

[Jadouli e Amrani, 2025] Jadouli A, Amrani C E (2025). Deep learning with pretrained'internal world'layers: A Gemma 3-based modular architecture for wildfire prediction. *arXiv preprint arXiv:2504.18562*.

[Jawahar et al., 2019] Jawahar G, Sagot B, Seddah D (2019). What does BERT learn about the structure of language? Em *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*.

[Jiang et al., 2024] Jiang Z, Carlson B, Deiana A, Eastlack J, Hauck S, Hsu S C, et al. (2024). Machine learning evaluation in the global event processor FPGA for the ATLAS trigger upgrade. *Journal of Instrumentation*, 19(05):P05031.

[Jiao et al., 2025] Jiao J, Afroogh S, Xu Y, Phillips C (2025). Navigating LLM ethics: Advancements, challenges, and future directions. arXiv:2406.18841v4 [cs.CY].

[Kaddour et al., 2023] Kaddour J, Harris J, Mozes M, Bradley H, Raileanu R, McHardy R (2023). Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*.

[Khoda et al., 2023] Khoda E E, Rankin D, Teixeira de Lima R, Harris P, Hauck S, Hsu S C, et al. (2023). Ultra-low latency recurrent neural network inference on FPGAs for physics applications with hls4ml. *Machine Learning: Science and Technology*, 4(2):025004.

[Kotek et al., 2023] Kotek H, Dockum R, Sun D (2023). Gender bias and stereotypes in large language models. Em *Proceedings of The ACM Collective Intelligence Conference*, CI '23, p. 12–24. ACM.

[Lan et al., 2020] Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, Soricut R (2020). ALBERT: A Lite BERT for self-supervised learning of language representations. arXiv:1909.11942v6 [cs.CL].

[Lewis et al., 2019] Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, Stoyanov V, Zettlemoyer L (2019). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

[Li et al., 2023] Li L J, Zhou S L, Chao F, Chang X, Yang L, Yu X, Shang C, Shen Q (2023). Model compression optimized neural network controller for nonlinear systems. *Knowledge-Based Systems*, 265:110311.

[Liang et al., 2023] Liang P, Bommasani R, Lee T, Tsipras D, Soylu D, Yasunaga M, et al. (2023). Holistic evaluation of language models. *Transactions on Machine Learning Research*. Featured Certification, Expert Certification.

[Lin, 2004] Lin C Y (2004). ROUGE: a package for automatic evaluation of summaries. Em *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004, Barcelona, Spain*, p. 74–81.

[Lin et al., 2024] Lin J, Tang J, Tang H, Yang S, Chen W M, Wang W C, et al. (2024). AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

[Lin et al., 2022] Lin T, Wang Y, Liu X, Qiu X (2022). A survey of transformers. *AI open*, 3:111–132.

[Liu et al., 2024a] Liu S, Lu Y, Fang W, Li M, Xie Z (2024a). OpenLLM-RTL: Open dataset and benchmark for LLM-aided design RTL generation. Em *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '24, p. 1–9. ACM.

[Liu et al., 2023] Liu Z, Oguz B, Zhao C, Chang E, Stock P, Mehdad Y, et al. (2023). Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

[Liu et al., 2024b] Liu Z, Zhao C, Iandola F, Lai C, Tian Y, Fedorov I, et al. (2024b). MobileLLM: optimizing sub-billion parameter language models for on-device use cases. ICML'24. JMLR.org.

[Loukas et al., 2023] Loukas L, Stogiannidis I, Diamantopoulos O, Malakasiotis P, Vassos S (2023). Making LLMs worth every penny: Resource-limited text classification in banking. Em *Proceedings of the Fourth ACM International Conference on AI in Finance*, p. 392–400.

[Ma et al., 2023] Ma X, Fang G, Wang X (2023). LLM-Pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720.

[Mahmud et al., 2025] Mahmud D, Hajmohamed H, Almentheri S, Alqaydi S, Aldhaheri L, Khalil R A, Saeed N (2025). Integrating LLMs with ITS: Recent advances, potentials, challenges, and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 26(5):5674–5709.

[Miao et al., 2022] Miao S, Liu M, Li P (2022). Interpretable and generalizable graph learning via stochastic attention mechanism. Em Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G, Sabato S, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, p. 15524–15543. PMLR.

[Naveed et al., 2023] Naveed H, Khan A U, Qiu S, Saqib M, Anwar S, Usman M, et al. (2023). A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

[OpenAI et al., 2024] OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, et al. (2024). GPT-4 technical report. arXiv:2303.08774 [cs.CL].

[Ouyang et al., 2022] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, et al. (2022). Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

[Penedo et al., 2023] Penedo G, Malartic Q, Hesslow D, Cojocaru R, Cappelli A, Alobeidli H, et al. (2023). The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only. arXiv:2306.01116 [cs.CL].

[Press et al., 2022] Press O, Smith N A, Lewis M (2022). Train short, test long: Attention with linear biases enables input length extrapolation. arXiv:2108.12409 [cs.CL].

[Raffel et al., 2020] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

[Raiaan et al., 2024] Raiaan M A K, Mukta M S H, Fatema K, Fahad N M, Sakib S, Mim M M J, et al. (2024). A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access*, 12:26839–26874.

[Sanh et al., 2020] Sanh V, Debut L, Chaumond J, Wolf T (2020). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108 [cs.CL].

[Shazeer, 2019] Shazeer N (2019). Fast transformer decoding: One write-head is all you need. arXiv:1911.02150 [cs.NE].

[Sun et al., 2023] Sun M, Liu Z, Bair A, Kolter J Z (2023). A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

[Suzgun et al., 2023] Suzgun M, Scales N, Schärli N, Gehrmann S, Tay Y, Chung H W, et al. (2023). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

[Tay et al., 2022] Tay Y, Dehghani M, Bahri D, Metzler D (2022). Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6).

[Tian et al., 2023] Tian K, Mitchell E, Zhou A, Sharma A, Rafailov R, Yao H, Finn C, Manning C D (2023). Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*.

[Touvron et al., 2023] Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M A, Lacroix T, et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

[V. et al., 2025] V. J M N, Giraldo D M, Segura S G, Corchado J M, la Prieta F D (2025). Bioinspired small language models in edge systems for bee colony monitoring and control. *Internet of Things*, 32:101633.

[Vaswani et al., 2017] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I (2017). Attention is all you need. Em *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

[Wang et al., 2023] Wang J, Hu X, Hou W, Chen H, Zheng R, Wang Y, et al. (2023). On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2302.12095*.

[Weidinger et al., 2022] Weidinger L, Uesato J, Rauh M, Griffin C, Huang P S, Mellor J, et al. (2022). Taxonomy of risks posed by language models. Em *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, p. 214–229.

[Xu et al., 2024] Xu J, Li Z, Chen W, Wang Q, Gao X, Cai Q, Ling Z (2024). On-device language models: A comprehensive review. *arXiv preprint arXiv:2409.00088*.

[Xue et al., 2023] Xue M, Liu D, Yang K, Dong G, Lei W, Yuan Z, Zhou C, Zhou J (2023). OccuQuest: Mitigating occupational bias for inclusive large language models. arXiv:2310.16517v1 [cs.CL].

[Yan et al., 2025] Yan X, Zhang T, Li Z, Zhang Y (2025). Progressive binarization with semi-structured pruning for LLMs. *arXiv preprint arXiv:2502.01705*.

[Yang et al., 2025] Yang A, Yu B, Li C, Liu D, Huang F, Huang H, et al. (2025). Qwen2.5-1m technical report. arXiv:2501.15383 [cs.CL].

[Yang et al., 2024] Yang J, Jin H, Tang R, Han X, Feng Q, Jiang H, et al. (2024). Harnessing the power of LLMs in practice: A survey on chatgpt and beyond. *ACM Trans. Knowl. Discov. Data*, 18(6).

[Yao et al., 2024a] Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y (2024a). A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, p. 100211.

[Yao et al., 2024b] Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y (2024b). A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211.

[Yenduri et al., 2024] Yenduri G, Ramalingam M, Selvi G C, Supriya Y, Srivastava G, Maddikunta P K R, et al. (2024). GPT (generative pre-trained transformer)— a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access*, 12:54608–54649.

[Zhang et al., 2024] Zhang P, Zeng G, Wang T, Lu W (2024). TinyLlama: An open-source small language model. arXiv:2401.02385v2 [cs.CL].

[Zhang et al., 2025] Zhang Q, Liu Z, Pan S (2025). The rise of small language models. *IEEE Intelligent Systems*, 40(1):30–37.

[Zhang et al., 2020] Zhang T, Kishore V, Wu F, Weinberger K Q, Artzi Y (2020). BERTScore: Evaluating text generation with BERT. Em *International Conference on Learning Representations*.

[Zhao et al., 2019] Zhao W, Peyrard M, Liu F, Gao Y, Meyer C M, Eger S (2019). MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. Em *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong, China. Association for Computational Linguistics.

[Zhao et al., 2023] Zhao W X, Zhou K, Li J, Tang T, Wang X, Hou Y, et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

[Zhao et al., 2024] Zhao Y, Lin C Y, Zhu K, Ye Z, Chen L, Zheng S, et al. (2024). Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209.

[Zheng et al., 2021] Zheng X, Zhang C, Woodland P C (2021). Adapting GPT, GPT-2 and BERT language models for speech recognition. Em *2021 IEEE Automatic speech recognition and understanding workshop (ASRU)*, p. 162–168. IEEE.

[Zheng et al., 2025] Zheng Z, Ning K, Zhong Q, Chen J, Chen W, Guo L, Wang W, Wang Y (2025). Towards an understanding of large language models in software engineering tasks. *Empirical Software Engineering*, 30(2):50.

[Zhu et al., 2024a] Zhu K, Wang J, Zhou J, Wang Z, Chen H, Wang Y, et al. (2024a). PromptRobust: Towards evaluating the robustness of large language models on adversarial prompts. Em *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, LAMPS '24, p. 57–68, New York, NY, USA. Association for Computing Machinery.

[Zhu et al., 2024b] Zhu X, Li J, Liu Y, Ma C, Wang W (2024b). A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.